

# Jugador Número 12

## Miembros del equipo:

Manuel Artero Anguita  
Javier López Gómez  
Roberto Marín Fernández  
Samuel Méndez Galán  
Marcos Alarcón Rivas  
Marina Bezares Álvarez  
Arturo Pareja García  
Alexandra Martín  
Daniel Escoz Solana  
Pedro Morgado Alarcón

## Proyecto de Ingeniería del Software Curso 2012 – 2013

Universidad Complutense de Madrid,  
Facultad de Informática



# JUGADOR #12

Índice de la documentación

Ingeniería del Software, 2013

## ÍNDICE

### 1.- Información del juego

1.1.- Introducción al proyecto .....	Pág. 4
1.2.- Definición del juego .....	Pág. 9
1.3.- Funcionamiento del partido .....	Pág. 18
1.4.- Estados en el partido .....	Pág. 21
1.5.- Definición del prototipo SimuMatch .....	Pág. 25
1.6.- Diagramas de clases de SimuMatch .....	Pág. 29
1.7.- Diagramas de flujo de SimuMatch .....	Pág. 35
1.8.- Conclusiones de SimuMatch .....	Pág. 38
1.9.- Reglas del juego de cartas .....	Pág. 40
1.10.- Detalles del juego de cartas .....	Pág. 44
1.11.- Simulación de subida de niveles y experiencia.....	Pág. 46
1.12.- Relación de las habilidades del juego.....	Pág. 47

### 2.- Documentación técnica

2.1.- Especificación de Requisitos Software .....	Pág. 48
2.2.- Gestión de Riesgos .....	Pág. 59
2.3.- Previsión de líneas de código para el proyecto.....	Pág. 63
2.4.- Métrica de puntos de función para el proyecto .....	Pág. 68
2.5.- Casos de uso .....	Pág. 75
2.6.- Diagramas de casos de uso .....	Pág. 97
2.7.- Diagramas de flujo de ejecución.....	Pág. 101
2.8.- Diagramas de clases .....	Pág. 122

### 3.- Documentación referencial

3.1.- Guía del framework Yii .....	Pág. 130
3.2.- Instalación y uso de XAMPP .....	Pág. 146
3.3.- Manual de Git .....	Pág. 148
3.4.- Tutorial: cómo conectar PHP y MySQL .....	Pág. 161
3.5.- Tutoriales adicionales y enlaces de interés .....	Pág. 164
3.6.- Guía de estilo de la documentación.....	Pág. 167

### 4.- Documentación de organización

4.1.- Actas de reuniones.....	Pág. 174
4.2.- Actas de entregas del primer cuatrimestre .....	Pág. 185
4.3.- Métodos de organización, planificación y reparto de tareas semanal.....	Pág. 192

### 5.- Documentación adicional

5.1.- Encuestas de feedback para las demos .....	Pág. 231
5.2.- Revisión del 13 de mayo al producto .....	Pág. 238

# JUGADOR #12

## Índice de la documentación

Ingeniería del Software, 2013

5.3.- Presentación inicial del proyecto .....	Pág. 240
5.4.- Revisión del proyecto de diciembre .....	Pág. 246
5.5.- Modelo RUP en el proyecto actual .....	Pág. 250
5.6.- Documentación informal adicional .....	Pág. 256

# JUGADOR #12

Información del juego: introducción al proyecto

Ingeniería del Software, 2013

## Introducción

Este documento da una visión muy básica de la motivación inicial del proyecto así como algunas comparativas con otros juegos de navegador.

## Índice

1. Motivación inicial
2. Comparativa con otros juegos de navegador
  - 2.1 OGame
  - 2.2 Bitefight
  - 2.3 Comunio
  - 2.4 Hattrick
  - 2.5 Otros
3. Brainstorming y consideraciones iniciales

## JUGADOR #12

Información del juego: introducción al proyecto

Ingeniería del Software, 2013

### 1. Motivación inicial

En esta sección daremos unas pequeñas pinceladas de los factores que nos han hecho decantarnos por este tipo de proyecto.

Desde hacía ya más de un año parte de los actuales integrantes del equipo de desarrollo habíamos tratado de crear algún tipo de aplicación web (un foro, juego por navegador, etc.). Por problemas de disponibilidad de horarios y compaginación con las asignaturas de la facultad, la idea no llegó a buen puerto. Fue en la asignatura de Ingeniería del Software donde vimos nuestra gran oportunidad para llevar a cabo esta idea y que, aún por encima, sirviese para la carrera.

Pero, ¿por qué un juego y no otro tipo de aplicación?. Este tipo de sistema nos pareció mucho más interesante de cara al aprendizaje que otros más comunes. Por ejemplo, el diseño de un foro u otra página web, en general, resulta bastante más simple y conocido. El juego por navegador nos permitía dar rienda suelta a nuestra creatividad incluyendo módulos personalizados (como el desarrollo de un partido o las acciones) y esto es útil de cara al desarrollo de nuevas aplicaciones ya que fomenta el uso del “brainstorming” y elimina, en parte, dependencias como la de “voy a ver cómo han hecho esta otra aplicación y basarme en ello”. Sí es cierto que el juego tiene una base característica de todos los de navegador, pero creímos que ese factor de creatividad era motivo suficiente para seleccionarlo como candidato a implementación.

La temática y enfoque del juego surgieron de la mano de Roberto Marín en aquella época en la que tratábamos de hacer algún proyecto web al margen de la facultad. Basándonos en los juegos ya existentes (OGame, Bitfight, Comunio, etc.) decidimos crear uno de temática ya conocida (fútbol) pero aportando un enfoque muy diferente al presente en los juegos actuales (que, en general, se basan en manejar equipos de fútbol, y no aficiones).

En un principio, el proyecto no fue propuesto (ni se nos había pasado por la cabeza) para la asignatura de Ingeniería del Software. El resto de propuestas eran mucho más interesantes y hacían uso de tecnologías más recientes (NFC, Android, etc.). A pesar de ello, decidimos lanzarnos y hacer una presentación del sistema para ver si conseguíamos a algún interesado. Lo cierto es que no teníamos muchas esperanzas, pero tras la primera presentación, y ver que a bastante gente le había parecido interesante, decidimos reforzar nuestra candidatura para que el proyecto fuese seleccionado para implementación.

Así fue que, a partir de entonces, y hasta el comienzo de la fase de inicio, empezamos a refinar y dar ideas para que el proyecto resultase lo más atractivo posible.

Algunas de ellas, como la de añadirle una aplicación móvil, fueron descartadas por las limitaciones de tiempo de este curso.

Al margen de la motivación del tipo y enfoque del proyecto, existe otra subyacente, que es la del aprendizaje de tecnologías web. Muchos de los componentes actuales de grupo nunca habíamos usado este tipo de lenguajes y veíamos muy útil el aprendizaje de los mismos en un sistema “real”, ya que durante la carrera apenas se

# JUGADOR #12

Información del juego: introducción al proyecto

Ingeniería del Software, 2013

tocan. De hecho, el proyecto nos permitiría aprender de golpe varias tecnologías de forma básica (HTML, PHP, CSS, Javascript, etc.) así como el uso de frameworks (en nuestro caso, de PHP) y el modelo MVC.

Todo esto permitió constituir unos alicientes más que suficientes para comenzar el desarrollo del producto con un grupo que demuestra interés e implicación en el mismo.

## 2. Comparativa con otros juegos de navegador

En esta sección se llevará a cabo una descripción y comparación de otros productos parecidos que forman parte del mercado actual.

### 2.1 OGame

<http://www.ogame.com.es/>

Juego de navegador de estrategia con temática espacial muy conocido. Actualmente cuenta con gran cantidad de usuarios y servidores activos. A pesar de no compartir mucho parecido con el producto actual (Jugador Número 12), sí se puede tomar como un buen ejemplo y referencia de juego por navegador de éxito. Ésto nos indica, de entrada, que existe un sector de usuarios consumidores de este tipo de productos, lo cual refuerza la decisión del tipo de proyecto a llevar a cabo.

OGame fue creado por una empresa alemana (Gameforge) empleando, en su versión inicial más básica, tecnologías web como PHP, HTML, CSS, etc. Esto, asimismo, ratifica la decisión de los lenguajes seleccionados para la implementación de nuestro producto.

Por último, y para hacer especial hincapié en el éxito de este tipo de juegos, decir que incluso se han desarrollado herramientas de ayuda a los jugadores (calculadoras de recursos, combates, etc.).

### 2.2 Bitfight

<http://www.bitfight.es/game>

Otro juego de navegador también propiedad de la empresa alemana Gameforge. Se pone de manifiesto en esta sección para dejar patente que, tras el éxito de los primeros juegos de navegador, las compañías se han aventurado a lanzar nuevos modelos de los mismos.

En este caso, el usuario juega el papel de un hombre lobo o vampiro, tiene su propia guarida, lucha contra otros usuarios o npcs, gana experiencia y objetos, etc.

Este juego de rol no cuenta con la misma cantidad de jugadores que Ogame, pero sí ha llegado a tener una cantidad de servidores considerable.

De Bitfight se ha tomado la idea inicial de incluir objetos en Jugador Número 12, dándole cierto toque de juego de rol. Del mismo modo, inicialmente se había propuesto la idea de “peleas” entre aficionados

### 2.3 Comunio

<http://www.comunio.es/>

Pasando ahora a juegos de temática de fútbol, presentamos en primer lugar “Comunio”. De enfoque estratégico, pone al usuario en la piel de un manager de fútbol, permitiéndole configurar su propio equipo libremente. Ésto incluye fichajes,

# JUGADOR #12

## Información del juego: introducción al proyecto

Ingeniería del Software, 2013

alineaciones y tácticas de juego. Asimismo, permite la creación de ligas en las que los equipos serán puestos a prueba.

Este juego comparte con Jugador Numero 12 la temática, pero no el enfoque. Dado que ya existen varios que permiten administrar un equipo hemos decidido seguir una línea diferente permitiendo al usuario jugar el papel del aficionado ( hincha).

### 2.4 Hattrick

[www.hattrick.org](http://www.hattrick.org)

Al igual que ocurre con el anterior, éste presenta también un enfoque estratégico. De igual manera, permite al usuario crear un equipo propio y competir contra otros jugadores.

De nuevo, vemos un ejemplo de juego enfocado en el equipo a gestionar, pero no en la parte de la afición.

### 2.5 Otros

Aparte de los mencionados en los puntos anteriores, existe gran cantidad de juegos de navegador. Algunos de mayor éxito y otros relegados a un segundo plano.

En general, siguiendo la evolución de este tipo de productos, se puede ver muy claramente que se lanzan versiones con diferentes temáticas y enfoques, manteniendo en el mercado sólo aquellas que resultan atractivas para los usuarios y descuidando, en cierto modo, las demás.

Ogame, por ejemplo, crece continuamente ya que el modo de juego resulta apropiado para el mercado actual, mientras que otro tipo de juegos como Bitefight han tenido que cerrar algunos servidores y agrupar a los usuarios en los restantes.

Esto sirve como apoyo y aliciente a la idea de lanzar un producto con temática conocida pero nuevo enfoque para ver la reacción de los jugadores. En caso de tener éxito puede continuar creciendo, y si no, puede ser relegado a un segundo plano e iniciado un nuevo proyecto diferente.

## 3. Brainstorming y consideraciones iniciales

Durante la fase de inicio del proyecto se han llevado a cabo frecuentes reuniones con el fin de detallar lo máximo posible la estructura básica del producto. Para ello se ha hecho un brainstorming de ideas. A continuación mostramos algunos de los principales puntos tratados:

- Recursos: al principio se desconocían cuáles eran los recursos del juego. Se sabía que el dinero figuraría seguro entre la lista de candidatos, pero no cuáles serían los demás. Surgieron gran cantidad de ideas al respecto, pero todas convergían o se podían resumir en 3 recursos principales: dinero, ánimo e influencias. Lo único que cambiaba en las ideas aportadas por cada componente del grupo era la forma de nombrar dicho recurso.  
Otra de las ideas tomadas del brainstorming fue la diferenciación entre las recuperaciones de cada uno de los recursos (el dinero con el tiempo, ánimo principalmente con los partidos, etc.).
- Perfiles: en cuanto a los perfiles de jugador, ocurría algo muy parecido a los recursos. Existían dos tipos de jugador seguros (el ultra y el empresario) pero,

## JUGADOR #12

Información del juego: introducción al proyecto

Ingeniería del Software, 2013

tras el brainstorming asociado, surgió un nuevo tipo de personaje a medio camino entre ambos: la movedora de masas (animadora). La inclusión de este nuevo perfil nos permitió asimismo asociar un recurso principal o característico a cada tipo de personaje.

- Acciones: sin duda el brainstorming más productivo. Hacían falta gran cantidad de acciones, por lo que cada componente del grupo debía aportar nuevas ideas. Se llegó a la siguiente reunión con un buen número de opciones posibles y resultó relativamente sencillo seleccionar varias de ellas para una implementación inicial. El resto han sido guardadas para recurrir a ellas en el momento de ampliar el repertorio de acciones.
- Factores de partido: un tema muy delicado fue la decisión de qué factores influirían en un partido. Surgieron multitud de ideas de las cuales fueron seleccionadas las más apropiadas y ayudaron a diferenciar entre factores que cambiarían antes del partido y aquellos que sólo se modificarían durante el encuentro. En el primer grupo encontramos, por ejemplo, el ambiente o el aforo, y en el segundo, la moral y factores ofensivo/defensivo.
- Peñas de aficionados: una idea original que emulaba el comportamiento de las hermandades de los juegos de rol fueron las “peñas”. Eran agrupaciones de jugadores de una misma afición en una especie de “club” privado. Era un instrumento meramente presentacional y que permitía crear nuevas funcionalidades como un ranking de las peñas más activas de un servidor. Esta idea fue descartada para una primera iteración dadas las limitaciones de tiempo.
- Objetos: otro tema interesante que se trató fue la inclusión de objetos para los jugadores. Por ejemplo permitir que pudiesen comprar bufandas, bombos, etc. Al igual que ocurría con las peñas, esta idea se descartó pronto por limitaciones de tiempo. En futuros ciclos se retomará.

# JUGADOR #12

Información del juego: definición del juego

Ingeniería del Software, 2013

## Introducción

Este documento define la dinámica del juego «Jugador número 12». En él, están recogidos todos los aspectos relevantes del juego y cómo funcionan. Intentamos que esté explicado el juego para que una persona que no lo conozca lo entienda y lo pueda jugar.

## Índice

1. Presentación
2. Personajes
  - El ultra
  - La movedora de masas
  - El empresario
3. Afición
4. Recursos
  - 4.1 Dinero
  - 4.2 Ánimo
  - 4.3 Influencia
5. Acciones
6. Árbol de habilidades, nivel y experiencia
7. El partido

# JUGADOR #12

Información del juego: definición del juego

Ingeniería del Software, 2013

## 1. Presentación

«Jugador número 12» se puede clasificar como un juego de estrategia multijugador, orientado a un público no necesariamente aficionado al fútbol. La temática se centra en la gestión de los aficionados de un equipo.

«Jugador número 12» pondrá al usuario en la piel de un hincha de fútbol permitiéndole organizar la afición de su equipo.

Para ello, el jugador tendrá la posibilidad de escoger uno de los tres perfiles disponibles:

- *El ultra*, que con su espíritu conformará la primera línea de apoyo al equipo.
- *La movedora de masas*, que intentará conseguir llenar los estadios.
- *El empresario*, que representa las luchas en los despachos manejando grandes cantidades de dinero.

Acto seguido, el jugador deberá elegir a qué afición unirse: comenzar en solitario con un equipo que cuente con pocos seguidores, y llevarlo hasta la cima, o agruparse con sus amigos en uno de los grandes y mantener la categoría.

Una vez elegido el equipo el usuario tendrá la opción de proponer acciones para preparar el partido con la afición o podrá decidir poner recursos en alguna acción ya propuesta, de esta forma apoyará al equipo para permitirle ganar el siguiente partido

Dentro del juego, el jugador encontrará a su disposición tres recursos bien distinguidos:

- *El ánimo*, o capacidad que tiene un hincha para animar.
- *La influencia*, que refleja el renombre del hincha en la sociedad.
- *El dinero*, que se irá incrementando con el tiempo y permitirá comprar diversos objetos.

Una vez haya hecho acopio de recursos el jugador podrá gastarlos para apoyar a su equipo y hacerle ganar el próximo partido. Podrá gastar los recursos en distintos eventos convocados por él mismo o por integrantes de su afición.

Gástate el sueldo en una entrada vip, píntate el cuerpo con los colores de tu equipo o, con ayuda de tus camaradas, haz la mejor ola jamás vista. Eso sí, deberás prestar atención a todo lo que hagas, porque si realizas acciones ilícitas podrías acabar sancionado...

Cuando llegue el día del partido quedará de relieve cuál es la mejor afición; aquella cuyo equipo alcance la gloria. Y es que nos hemos dejado lo mejor para el final, el usuario tendrá la posibilidad de participar interactivamente durante los encuentros del

# JUGADOR #12

## Información del juego: definición del juego

Ingeniería del Software, 2013

equipo en un sistema de turnos. Si el hincha ha adquirido la entrada para un encuentro podrá interferir directamente en su desarrollo y tratar de acallar a la afición rival.

**Todo es admisible para ayudar a tu equipo a ganar la liga.**

## 2. Personajes

Como ya se ha visto en la introducción, el usuario deberá elegir uno de los tres perfiles disponibles para su personaje. Los perfiles determinan el modo de juego del jugador. Cada uno de ellos cuenta con su propio árbol de habilidades. Todos los jugadores realizan acciones<sup>1</sup> para apoyar a su equipo, sin embargo, cada perfil tiene acciones específicas para apoyar a su equipo a su manera.

Una pequeña descripción de cada uno:

- **El ultra**

Representa la fuerza bruta, el sector más radical de la afición, que siempre intenta hacer mella en la moral del equipo contrario para lograr que su equipo logre alzarse con la victoria. Aunque suelen ser pocos por el carácter agresivo y escandaloso que tienen, saben hacerse escuchar y animar a su equipo mejor que cualquier otro.

- **La movedora de masas**

Organizadora de eventos por naturaleza, utiliza las redes sociales y cualquier medio de comunicación a su alcance para mover a los aficionados a dar apoyo a su equipo. Nadie puede igualarse a la movedora de masas en su afán por conseguir adeptos y en ganarse su confianza tan fácilmente. En cambio, su perfil de estudiante o becario hace que su nivel económico sea limitado.

- **El empresario**

Está al frente de la lucha de las aficiones en los despachos, mueve cantidades abrumadoras de dinero. No pone pegas ni a las apuestas, ni a los sobornos y en general a nada que le proporcione rentabilidad económica. Representa un alto cargo dedicado en cuerpo y alma a los negocios, pero a la hora de ir a ver un partido, prefiere sentarse en los palcos y ser un mero observador.

## 3. Afición

Como mencionamos antes, los jugadores formarán parte de una **afición**, que consta de todos los jugadores que dan apoyo a un equipo de fútbol en concreto. Habrá una afición por cada equipo de fútbol. Hay que destacar la fuerza de la afición, que representa todo el esfuerzo que han puesto los hinchas en animar a su equipo. También hay que mencionar la posibilidad de que un jugador cambie de afición pero, no le saldrá gratis, tendrá una penalización.

# JUGADOR #12

Información del juego: definición del juego

Ingeniería del Software, 2013

Ser miembro de una afición permitirá al usuario realizar o participar en acciones grupales de la misma, las cuales necesitarán que varios miembros aporten recursos para que se lleven a cabo con éxito.

Para que la organización de la afición sea lo más sencilla posible se cuenta con un sistema de mensajes privados y un sistema de anuncios en el que cada mensaje llegará a los demás miembros de la misma.

## 4. Recursos

Cada jugador cuenta con sus propios recursos. Existen tres tipos de recursos (*dinero*, *árbol* e *influencia*) mediante los cuales los jugadores interactúan con el resto de elementos del juego, generalmente a través de acciones. Están claramente diferenciados en obtención, uso y mantenimiento.

### 4.1 Dinero

El uso principal de este recurso es el de pagar las entradas a los estadios para los partidos, realizar acciones que tengan que ver con negocios y poder comprar los objetos que se usarán para las acciones del juego.

El dinero se gana con el tiempo, el contador suma una cantidad de dinero cada minuto. Al comienzo del juego se gana una cantidad fija de dinero. Podemos aumentar la cantidad que ganamos aumentando de nivel y escogiendo habilidades específicas del árbol que supongan un beneficio.

El empresario, capaz de amasar grandes fortunas en poco tiempo, destaca en este área. La cantidad fija inicial del empresario es abundante. Así mismo cuando aumente de nivel y escoja habilidades específicas de dinero, la rentabilidad que obtenga será significativamente mayor a las de los otros dos perfiles.

El ultra, dispuesto a animar a su equipo con todas sus ganas, destina una cantidad de dinero moderada para apoyar a su equipo. La cantidad fija inicial del ultra es austera. Asimismo cuando aumente de nivel y escoja habilidades específicas de dinero, la rentabilidad que obtenga será mesurada.

Las movedoras de masas, con su perfil universitario, no disponen de mucho dinero. La cantidad fija inicial de este tipo de personaje es ordinaria. Asimismo cuando aumente de nivel y escoja habilidades específicas de dinero, la rentabilidad que obtenga será reducida.

### 4.2 Árbol

No todo el mundo anima con la misma fuerza. El ánimo es la cuantificación de la fuerza con la que apoyamos a nuestro equipo, necesaria para realizar ciertas actividades dentro y fuera del estadio.

## JUGADOR #12

### Información del juego: definición del juego

Ingeniería del Software, 2013

El ánimo no se gana, es una tasación de cómo nos sentimos con nuestro equipo. Si nuestro equipo gana, ganamos ánimo y si nuestro equipo pierde, lo perdemos. Por supuesto, todos llevamos a nuestro equipo en el corazón por lo que al final el ánimo siempre aumenta. Sin embargo, si nuestro equipo no nos aporta ninguna emoción el ánimo crecerá muy lentamente.

Por definición, ninguno nos podemos entregar más de al 100% por tanto la cantidad de ánimo que podemos acumular tiene un tope. Algunos tendrán un ánimo de 500 puntos y otros de 600, pero nadie puede acumular más ánimo del máximo. Debido a la naturaleza limitada del ánimo, es un recurso que llama a ser gastado, puesto que no se puede acumular para siempre.

El gasto de ánimo es como el del dinero, por cantidad. El ánimo se puede gastar, pero el máximo - nuestra fe en el equipo - nunca decrece.

El ultra se deja la piel para respaldar a su equipo. El tope inicial de su ánimo es alto. Cuando su equipo gana obtiene grandes cantidades del recurso y cuando pierde no se desanima demasiado. Además aún cuando lo gana lentamente la cantidad es relevante.

Los movedores de masa siempre dispuestos a montar cualquier fiesta con la excusa de animar al equipo, manejan unas cantidades de ánimo moderado. El tope inicial de su ánimo es medio. Cuando su equipo gana se lo dicen a todo el mundo y recuperan ánimo en cantidades aceptables. Por otro lado, a pesar de que su equipo pierda, siempre puede desahogarse en twitter para animarse, por lo que no pierde mucho ánimo. Recupera ánimo a un ritmo menor que en el caso del ultra, pero de forma notable..

El empresario, apoyando a su equipo desde los negocios, tiene un ánimo exiguo. El tope inicial de este recurso es bajo. Cuando su equipo gana no se alegra demasiado, pero cuando pierde piensa en la cantidad de dinero malgastada y su ánimo decae rápidamente. Su recuperación de ánimo básica resulta notablemente lenta.

### 4.3 Influencia

Para que un equipo se sienta como en casa, hacen falta contactos que nos permitan mover a las masas. La influencia en los medios es importante, y cuanta más tengamos, más fácil nos será animar a la gente para que vea el partido en directo. La influencia es un valor que indica nuestro poder para movilizar a la afición y promocionar a nuestro equipo. Es un valor que indica cuántos contactos tenemos y la facilidad que tenemos para movilizar a la gente.

Este recurso no se gasta; simplemente la empleamos para una determinada acción. Es decir, llamamos a nuestros contactos para pedirles un favor. Cuando esta acción acabe la influencia se restablecerá. Esto es, los contactos acabarán su trabajo y volverán a estar disponibles. Sin embargo, la gente a nuestro cargo se cansa y no

## JUGADOR #12

### Información del juego: definición del juego

Ingeniería del Software, 2013

podemos abusar de ellos, por lo que la influencia no se restablece inmediatamente si no a un ritmo muy lento.

Al igual que ocurría con el ánimo, existe un tope de influencias, que representa la máxima capacidad de movilización de aficionados por un usuario.

El perfil con mayor cantidad de este recurso es el de la movedora de masas, pudiendo ejercer su influencia sobre cualquier medio de comunicación a su alcance y ganar así seguidores de cara al próximo encuentro. Gana puntos de influencia con gran velocidad, y cuando los gasta, sus contactos vuelven a estar disponibles rápidamente.

El empresario cuenta también con su propio tipo de influencia, la que aporta el dinero. Tiene un máximo de este recurso inferior a la movedora de masas y lo regenera con mayor lentitud.

El ultra no cuenta con demasiada influencia. Los demás aficionados no ven bien sus habilidades radicales. Cuando gasta su escasa influencia tarda mucho en recuperarla.

Perfil \ Recurso	Dinero	Ánimo	Influencia
Ultra	Medio	Alto	Bajo
Movedora de masas	Bajo	Medio	Alto
Empresario	Alto	Bajo	Medio

Tabla resumen que indica la relación de los recursos y los perfiles.

## 5. Acciones

Las acciones conforman el grueso de la jugabilidad para los usuarios. Cada perfil dispone de unas acciones distintas, que se adquieren a través del árbol de habilidades. Las acciones gastan unos determinados recursos, cada perfil basa sus acciones en su recurso predilecto. Las acciones tendrán requerimientos como nivel mínimo para realizar la acción, perfil, o número de participantes.

Las acciones pueden ser:

- **De jugador:** las lleva a cabo un solo jugador. Éste gasta una cantidad de recursos para obtener un beneficio inmediato sea permanente o no. Las acciones permanentes se encuentran en el árbol de habilidades.
- **De afición:** varios jugadores de la misma afición colaboran para lograr un fin concreto. Un miembro de la afición abre un evento aportando una serie de

## JUGADOR #12

### Información del juego: definición del juego

Ingeniería del Software, 2013

recursos. Dicho evento tiene un determinado tiempo para que el resto de los miembros de la afición participen en él, una serie de recursos que se deben llenar para que tenga éxito y un número máximo de jugadores que pueden participar en la acción. Cada jugador debe aportar unos recursos mínimos para poder participar, de esa manera se asegura que todos los participantes han aportado algo a la acción. Si los miembros cumplen los requisitos del evento antes del tiempo de expiración conseguirán un beneficio para el equipo. Si no lo consiguen, la acción será eliminada y los recursos devueltos sin conseguir ninguna bonificación. Esto incrementa la necesidad de organización por parte de los miembros de la afición para poder completar este tipo de acciones.

Por el uso las acciones se clasifican como:

- **Estáticas:** se pueden realizar en cualquier momento y son permanentes.
- **Preparatorias:** se usan antes de un partido, pero cuando ya está planificado.
- **Instantáneas:** sólo se activan durante el partido y solo duran un turno.

Las acciones también se pueden clasificar por su ética

- **Honradas:** no tienen consecuencias negativas.
- **Ilícitas:** tienen un efecto mucho mayor, pero existe posibilidad de que te pillen en cuyo caso tendrás algún tipo de castigo (cárcel, multa, etc).

Algunas actividades cuentan con niveles de éxito en los eventos: una acción conjunta requiere unos requisitos mínimos para llevarse a cabo, pero una vez alcanzados estos, se puede seguir invirtiendo recursos para maximizar su efecto final.

### 6. Árbol de habilidades, nivel y experiencia

El **árbol de habilidades** define las habilidades que permiten desbloquear las acciones que puede realizar un hincha. El diseño del mismo será circular y, dependiendo del tipo de personaje que use el jugador (ultra, movedora de masas, empresario), se empezará en un punto del árbol (círculo) u otro.

El árbol tiene varios círculos concéntricos. Cada uno de ellos lleva las habilidades correspondientes a un nivel. Las habilidades de mayor nivel estarán situadas en los niveles más cercanos al centro y las de nivel más bajo en los niveles más externos. En el centro habrá una habilidad especial común a todos los tipos de personajes que será la mejor habilidad del juego.

El jugador tiene un **nivel**. Los jugadores comienzan con nivel uno y a medida que consiguen **experiencia** suben de nivel. La experiencia se consigue realizando acciones individuales y creando y participando en las acciones de afición. Los jugadores que hayan acudido al partido, al finalizar éste, conseguirán también un plus de experiencia, que será mayor o menor dependiendo del resultado del partido. Cada vez que el jugador aumente de nivel, se desbloquearán nuevas habilidades del árbol que habilitarán nuevas acciones para el hincha..

## JUGADOR #12

Información del juego: definición del juego

Ingeniería del Software, 2013

El jugador no podrá escoger todas las habilidades del árbol, así que tendrá que elegir aquellas que le gusten más. Gracias al diseño circular del árbol, el jugador tendrá dos opciones para escoger las habilidades. La primera opción es ir profundizando en el árbol de tal manera que las habilidades que escoge el jugador serán de mayor nivel. La segunda opción es ir hacia un lado o hacia el otro del árbol para poder escoger habilidades de otro tipo de jugador. Por ejemplo, el jugador es un empresario y si va hacia la izquierda se acercará a las habilidades del ultra y si va a la derecha se acercará a las habilidades de la movedora de masas. De esta manera el jugador podrá personalizar en mayor medida a su personaje haciendo que tenga acciones más fuerte o haciendo que pueda realizar acciones de dos tipos de personajes a la vez.

### 7. El partido

Cada jornada se divide en tres partes:

- **Pre-partido:** es el tiempo que hay entre la finalización del último partido y el inicio del siguiente. Durante ese tiempo los jugadores podrán realizar las acciones **preparatorias** antes mencionadas. Los eventos de este tipo completados con éxito mejorarán de forma acumulativa el equipo para su próximo encuentro. La afición que más haya animado conseguirá que su equipo vaya al partido con más opciones de ganar.
- **Partido:** lo primero que tiene que hacer el aficionado antes del partido es comprar una entrada. De esta manera podrá entrar al campo y seguir animando a su equipo durante el partido. Cada partido se divide en dos partes. Hay un tiempo entre las dos partes, el descanso, en el cual se podrán realizar acciones especiales que mejorarán a tu equipo o empeorarán al contrario durante la segunda parte. Las dos partes se dividen en turnos. Durante cada uno de ellos, los aficionados que estén dentro del campo se dedicarán a realizar acciones **instantáneas** de partido. Al final del turno se tomarán los efectos de todas las acciones realizadas. Esto hará que la balanza del partido se decante por uno de los equipos o se equilibre.

Por ejemplo, si el partido está igualado y la afición de tu equipo anima mucho más que la afición del equipo contrario, aumentarán las probabilidades de que tu equipo marque gol al siguiente turno. Si al siguiente turno se cambian las tornas y la afición del equipo contrario empieza a animar fuerte, entonces el partido se equilibrará o incluso puede que aumenten las probabilidades de que le marquen gol a tu equipo. En función de cómo vaya tu equipo en cada turno, habrá que realizar un tipo de acciones u otras. Si están a punto de marcarle un gol a tu equipo es un buen momento para realizar alguna gamberrada como apuntar con un puntero láser al delantero del equipo contrario y que falle la ocasión o incluso sobornar al árbitro para que favorezca a tu equipo, aunque pueda tener consecuencias negativas si te pillan. Una vez se acaben los turnos, el partido habrá finalizado y el resultado no cambiará.

## JUGADOR #12

Información del juego: definición del juego

Ingeniería del Software, 2013

- **Post-partido:** justo después del partido es cuando los recursos se recuperan o se ganan en función del resultado del mismo. Si tu equipo ha ganado, recuperarás más recursos que si ha empatado o ha perdido. Por ejemplo, si el aficionado es un empresario, ha apostado por su equipo gana y éste finalmente gana, entonces recuperará el doble de dinero de lo que ha apostado en cuanto acabe el partido, pero si pierde o empata, perderá toda la inversión.

# JUGADOR #12

Información del juego: funcionamiento del partido

Ingeniería del Software, 2013

## Introducción

Este documento define la dinámica de los partidos de «Jugador número 12». Se mostrarán todos los detalles relevantes para la comprensión del funcionamiento de los mismos.

## Índice

1. Visión general
2. Preparando el partido
3. Interactuando en el partido
4. La fórmula
5. Fin del partido

# JUGADOR #12

Información del juego: funcionamiento del partido

Ingeniería del Software, 2013

## 1. Visión general

Los partidos conforman uno de los pilares fundamentales del juego. El sistema programa un calendario de partidos para todos los equipos al comienzo de una temporada. Cada uno de ellos será ejecutado por el sistema independientemente de que haya participantes en el momento de comienzo.

Los usuarios pueden participar en los encuentros antes o durante los mismos. Se ofrecen dos mecanismos para ésto: las acciones grupales e individuales y las acciones de partido. Cada partido dura una serie de turnos de igual duración y está dividido en 3 secciones: primera parte (N turnos), descanso (1 turno) y segunda parte (N turnos).

## 2. Preparando el partido

Como se ha mencionado antes, la forma de preparar un partido es hacer uso de las diferentes acciones grupales o de afición. Existe la opción de que algunas acciones individuales del personaje puedan influir en menor medida en el encuentro.

Existen una serie de factores que se pueden modificar antes de un partido a través del mecanismo anterior:

- Ambiente: compartido tanto por el equipo local como visitante. Es un indicador del ánimo de ambas partes del encuentro. Se usará principalmente para aumentar la cantidad de ánimo devuelto al final de un partido.
- Aforo: representa la cantidad de aficionados virtuales movilizados por un equipo. Será un factor influyente durante el partido para equilibrar las opciones de gol hacia el equipo con mayor aforo (y por lo tanto, mayor capacidad de animar).
- Nivel de equipo: simboliza el estado de la plantilla de un equipo a la hora de entrar al campo de juego. Hay acciones, como comprar jugadores de última hora, que incrementarán este estado para el próximo encuentro. Tendrá una influencia parecida a la del aforo durante el partido.

Cuando un usuario abre un nuevo evento (acción) de afición y consigue llenar todos los recursos necesarios asociados al mismo, se aplica el beneficio de dicha acción a los factores del próximo partido de la afición.

## 3. Interactuando en el partido

Los usuarios pueden participar activamente durante el encuentro. Para ello disponen del mecanismo de acciones de partido.

Este tipo de habilidades están disponibles únicamente durante los encuentros en juego. A diferencia de las acciones preparatorias o grupales, afectan a una serie de nuevos factores presentes durante el encuentro:

- Moral: cada equipo tendrá un valor de moral, que reforzará las acciones ofensivas y defensivas de dicho equipo. Hay acciones como crear una ola en todo el estadio que harán que la moral del equipo asociado aumente considerablemente.

## JUGADOR #12

Información del juego: funcionamiento del partido

Ingeniería del Software, 2013

- Factor ofensivo: indica la capacidad y predisposición al ataque de un equipo. Cuanto mayor sea este valor, más se acercarán los jugadores a la portería rival para provocar un gol.
- Factor defensivo: al contrario que el caso anterior, simboliza la capacidad de un equipo de defenderse ante los ataques rivales. Cuanto mayor sea este valor, más efectiva será la defensa del equipo y más le costará al rival acercarse a la portería.

En principio, no existe límite a un participante para ejecutar acciones de partido. Puede llevar a cabo tantas como sus recursos le permitan. Una vez creada una acción con éxito, se aplicará el beneficio asociado a los factores relacionados.

### 4. La fórmula

Conforma el mecanismo principal del partido. Recibe todos los parámetros modificados al final de cada turno y genera el resultado apropiado.

El partido se basa en un sistema de 20 estados. Los extremos, -10 y +10 representan los goles visitante y local respectivamente. Los estados intermedios indican cómo de cerca está cada una de las partes de lograr un tanto. El intervalo (-10, 0) corresponde a la superioridad visitante y el intervalo (0, 10) a la local. En el estado 0, ambos equipos se encuentran igualados.

Para manejar y modificar este sistema de estados se hace uso de una función interna con forma de campana de Gauss que representará las probabilidades de un equipo de saltar a un estado cercano.

Los parámetros y variación de los mismos vienen determinados por los 6 factores mencionados en los puntos anteriores. Al ir variando dichos factores mediante acciones grupales o de partido, la campana de Gauss se deforma acorde a esto y provoca los cambios de estado que hacen que el partido avance hacia una u otra dirección. Asimismo, se incluye un pequeño factor de aleatoriedad en los cálculos que permite simular los cambios de tornas en los partidos reales.

### 5. Fin del partido

Una vez finalizado el encuentro se detiene la participación de los usuarios mediante acciones de partido. Las nuevas acciones de afición (grupales) completadas pasarán a asociarse al siguiente encuentro de la afición.

Se toman todos los datos del partido y se recalcula la clasificación acorde a los mismos. Asimismo, se habilita la crónica del encuentro para ser visualizada por los usuarios de la aplicación y se activa la vista previa del siguiente encuentro de los equipos involucrados.

## JUGADOR #12

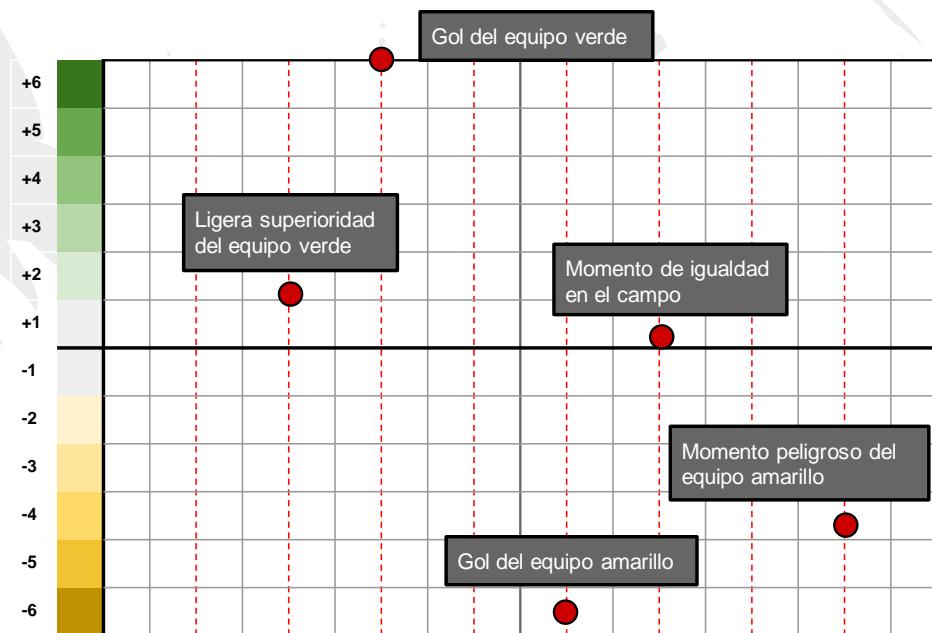
Información del juego: estados en el partido

Ingeniería del Software, 2013

# DESARROLLO DE LOS PARTIDOS

*Definición interna*

## Representación de un partido

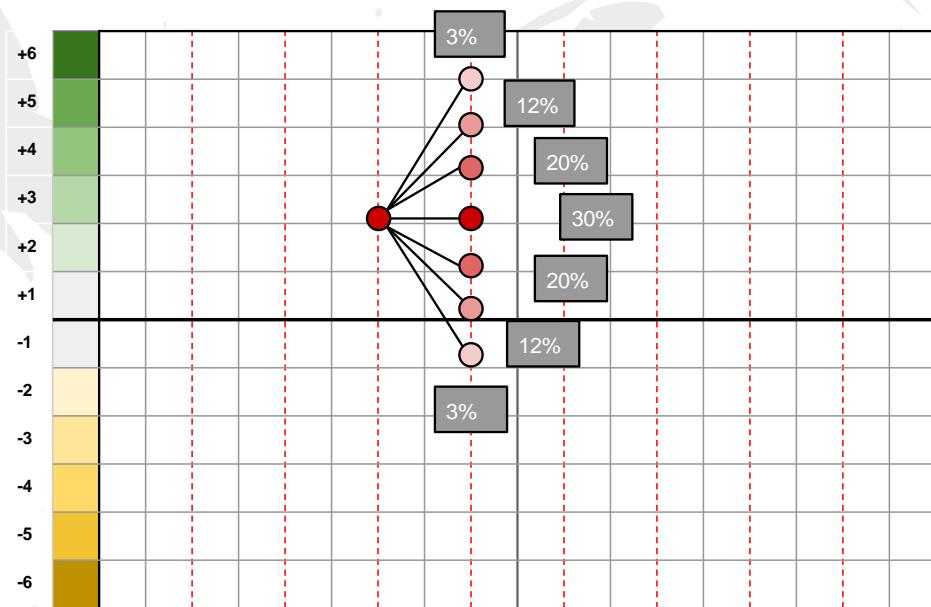


# JUGADOR #12

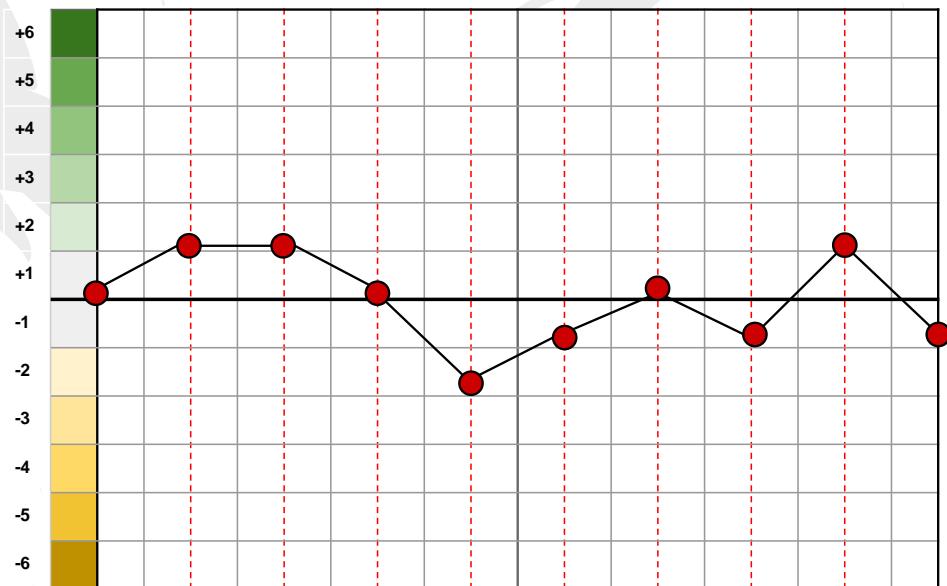
Información del juego: estados en el partido

Ingeniería del Software, 2013

## Desarrollo sin interferencias. Equipos igualados



## Desarrollo sin interferencias. Equipos igualados

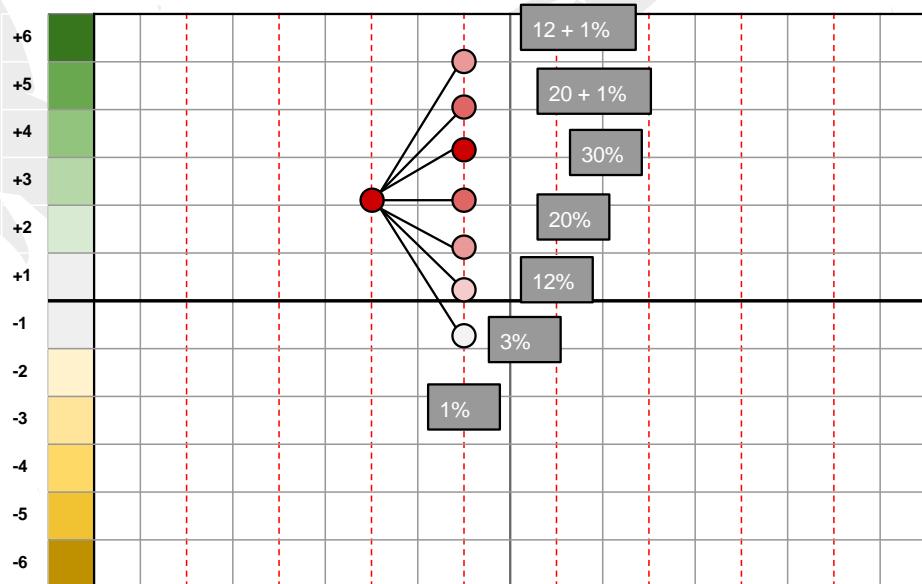


# JUGADOR #12

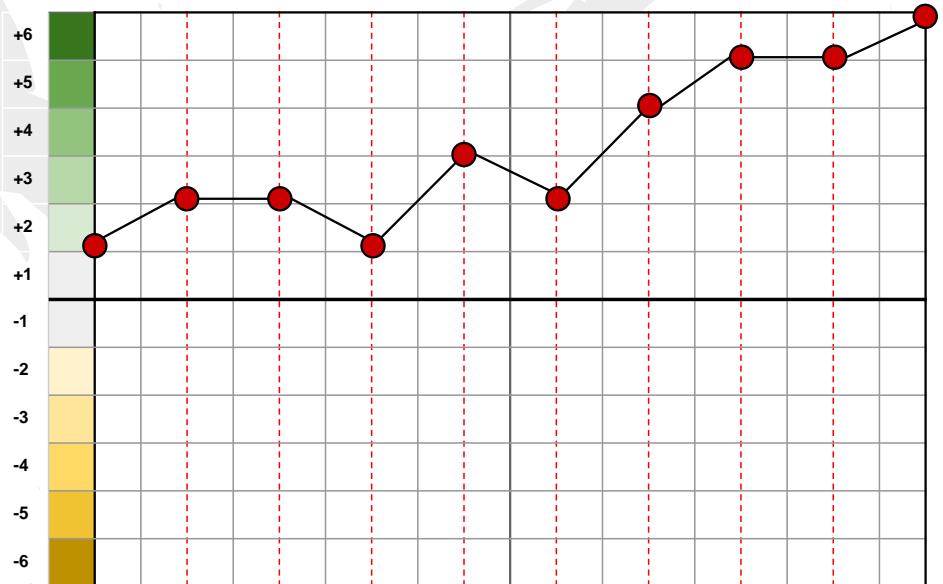
Información del juego: estados en el partido

Ingeniería del Software, 2013

Desarrollo sin interferencias. Ligera superioridad



Desarrollo sin interferencias. Ligera superioridad

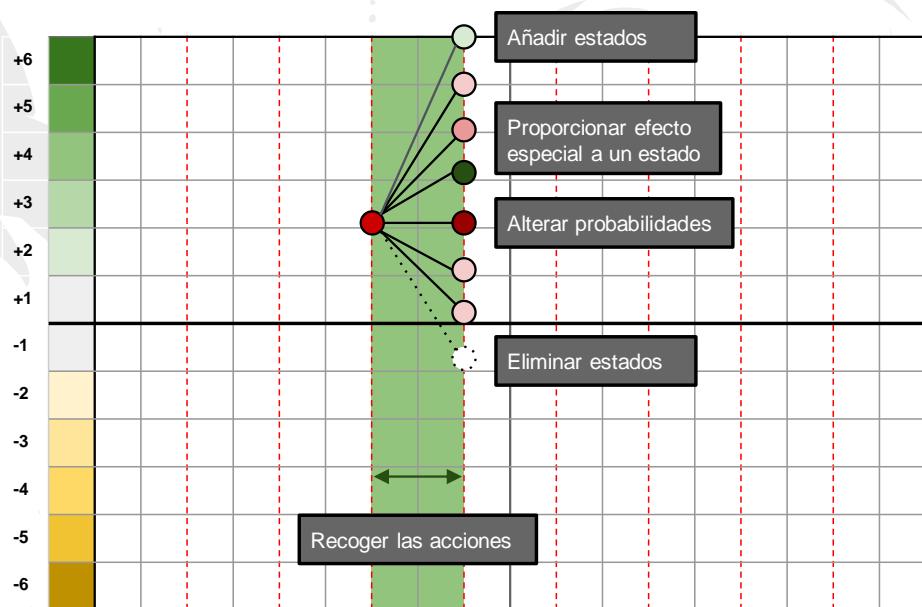


# JUGADOR #12

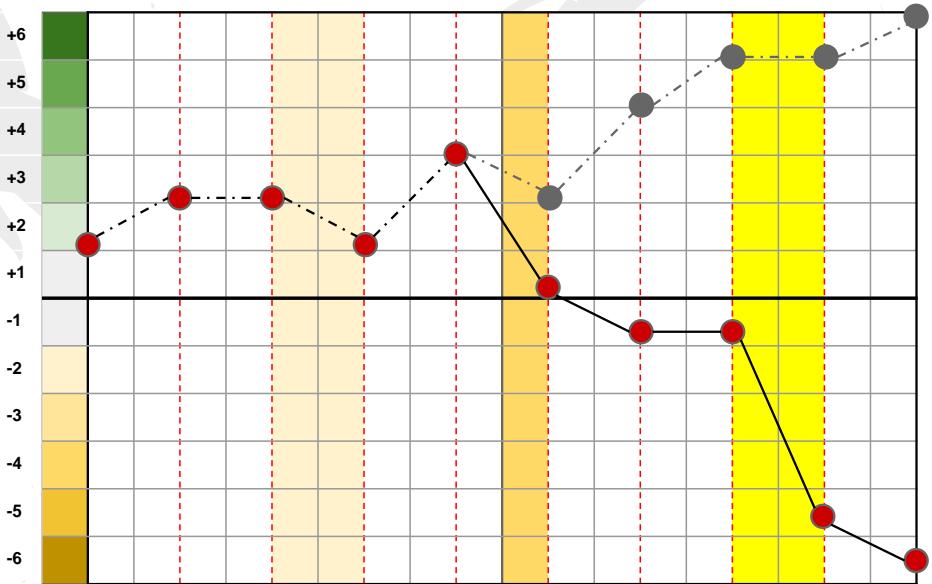
Información del juego: estados en el partido

Ingeniería del Software, 2013

## Introducir las interferencias



## Desarrollo con interferencias. Ligera superioridad



# JUGADOR #12

Información del juego: definición del prototipo SimuMatch

Ingeniería del Software, 2013

## Introducción

Este documento se creó para dejar constancia por escrito de las especificaciones que debía cumplir la parte del prototipo implementada en Java (llamado SimuMatch dada que su función era la de simular partidos).

El prototipo consta de dos partes, como se ha indicado en el documento “Reglas del juego de cartas” y la implementación en Java, quedando ésta última explicada a lo largo de este documento.

Nota: las imágenes son completamente originales y creadas expreso para el documento.

## Índice

1. Propósito del programa
2. Definición del programa
  - 2.1 Clases
    - 2.1.1 Class GraphicInterface  
Definición esquemática de la interfaz gráfica
    - 2.1.2 Class AbilitiesData
    - 2.1.3 Class Engine
    - 2.1.4 Class MatchSimu

# JUGADOR #12

Información del juego: definición del prototipo SimuMatch      Ingeniería del Software, 2013

## 1. Propósito del programa

El programa servirá para **simular el sistema de partidos** introduciendo las variables que condicionan un partido y mostrando paso por paso los resultados de estos.

Las variables de entrada comprenderán el abanico de habilidades o acciones que interfieren en el desarrollo de los partidos.

Los resultados mostrados comprenderán

- Un registro de las tablas de estados/probabilidad asociada a cada estado de todos los turnos jugados
- El estado actual

## 2. Definición del programa

### 2.1 Clases

El programa, en principio, contará con cuatro clases. Los nombres de las clases así como su funcionalidad quedan definidas en este documento; no así la distribución de paquetes que se deja a elección de los programadores. Tendremos:

- Class GraphicInterface
- Class AbilitiesData
- Class Engine
- Class MatchSimu

#### 2.1.1 Class GraphicInterface

La interfaz gráfica se hará por JSwing. Se busca la funcionalidad, no la estética ya que el programa es exclusivo para el equipo. Recomiendo hacer la interfaz gráfica con NetBeans. Un detalle de funcionamiento: una vez se carguen los modificadores de preparativos, el panel quedará bloqueado y no se podrán modificar ni cargar nuevas opciones para ningún equipo

##### *Definición esquemática de la interfaz gráfica*

- Panel contenedor
  - Panel izquierdo
    - Panel de modificadores de preparativos
      - Matriz de selectores, botones o entradas de texto para activar o fijar el valor de todas las acciones preparatorias.
      - Botón para cargar las acciones preparatorias
    - Panel de acciones inmediatas
      - Matriz de selectores, botones o entradas de texto para activar o fijar el valor de las acciones inmediatas
      - Botón para cargar las acciones inmediatas en el siguiente turno
  - Panel derecho
    - Panel registro de valores generados por turno

# JUGADOR #12

Información del juego: definición del prototipo SimuMatch Ingeniería del Software, 2013

- Tabla con los valores y resultado que se obtuvo en cada turno; el turno actual se resaltará sobre los demás
- Panel de resultado
  - Resultado del turno actual
  - Botón para calcular siguiente turno

The screenshot shows the user interface of the SimuMatch prototype. On the left, under 'Variables de entrada' (Input Variables), there are two sections: 'Modificadores de preparativos' (Preparation Modifiers) and 'Acciones inmediatas' (Immediate Actions). The 'Modificadores de preparativos' section contains eight buttons labeled 'Mod. 1' through 'Mod. 8'. The 'Acciones inmediatas' section contains four pairs of buttons labeled 'Acc. 1' through 'Acc. n'. Each pair consists of a small button and a larger empty box. Below each section is a 'Cargar' (Load) button. On the right, under 'Salida generada' (Generated Output), there are three sections: 'Registro de valores generados' (Generated Value Registry), 'Resultado (momento actual)' (Current Result), and a 'Siguiente' (Next) button. The 'Registro de valores generados' section displays a table with four rows. The first row is highlighted in light blue and contains the text 'Turno i | 2 | 12 | 20 | 30 | 20 | 12 | 2 | +2'. The subsequent rows are 'Turno i | 2 | 12 | 20 | 30 | 20 | 12 | 2 | +4', 'Turno i | 2 | 12 | 20 | 30 | 20 | 12 | 2 | -3', and 'Turno i | 2 | 12 | 20 | 30 | 20 | 12 | 2 | +3'. The 'Resultado (momento actual)' section shows a large '+3' button and a smaller 'Siguiente' button.

## 2.1.2 Class AbilitiesData

Esta clase representará el acceso a una “base de datos” para el programa. No tiene sentido en estos momentos de desarrollo conectar el programa con una base de datos real; así que optaremos por almacenar la información en ficheros estáticos que leeremos y procesaremos en esta clase.

Todos los ficheros de datos tendrán que ir en un **paquete data** diferente al paquete de código.

La finalidad de esta clase es encapsular al resto de clases el acceso a datos estáticos del estilo <Flag activo en el modificador 1 significa una distribución X de probabilidades>. En el siguiente enlace he dejado indicaciones en Java para hacer el procesamiento de ficheros. No es obligatorio usar esa forma de procesar ficheros ya que podría haber formas mejores o más eficientes.

## JUGADOR #12

Información del juego: definición del prototipo SimuMatch

Ingeniería del Software, 2013

### 2.1.3 Class Engine

Será la clase encargada de hacer los cálculos para determinar la tabla de estados y las probabilidades asociadas a los estados. Por lo tanto, usará un objeto de la tabla AbilitiesTable para obtener los datos.

Tendrá una estructura de datos para almacenar la tabla de estados/probabilidades. Podría ser un HashMap

Contendrá el método **calculateNextMatch** que generará un número aleatorio y consultará la tabla que previamente habrá generado para determinar el siguiente estado.

### 2.1.4 Class MatchSimu

Será la clase principal, la clase orquesta. Los eventos generados en la interfaz gráfica llamarán a métodos públicos de esta clase.

Usará un objeto **engine** para hacer las operaciones, por ejemplo recibir el resultado de un turno y dárselo a la interfaz para que lo muestre.

Los eventos más importantes son

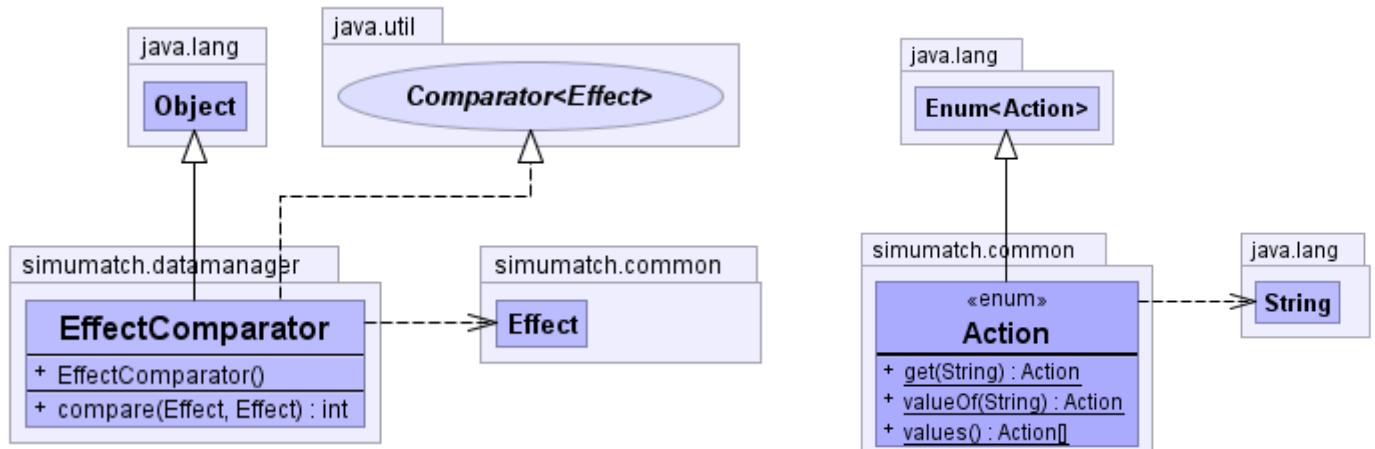
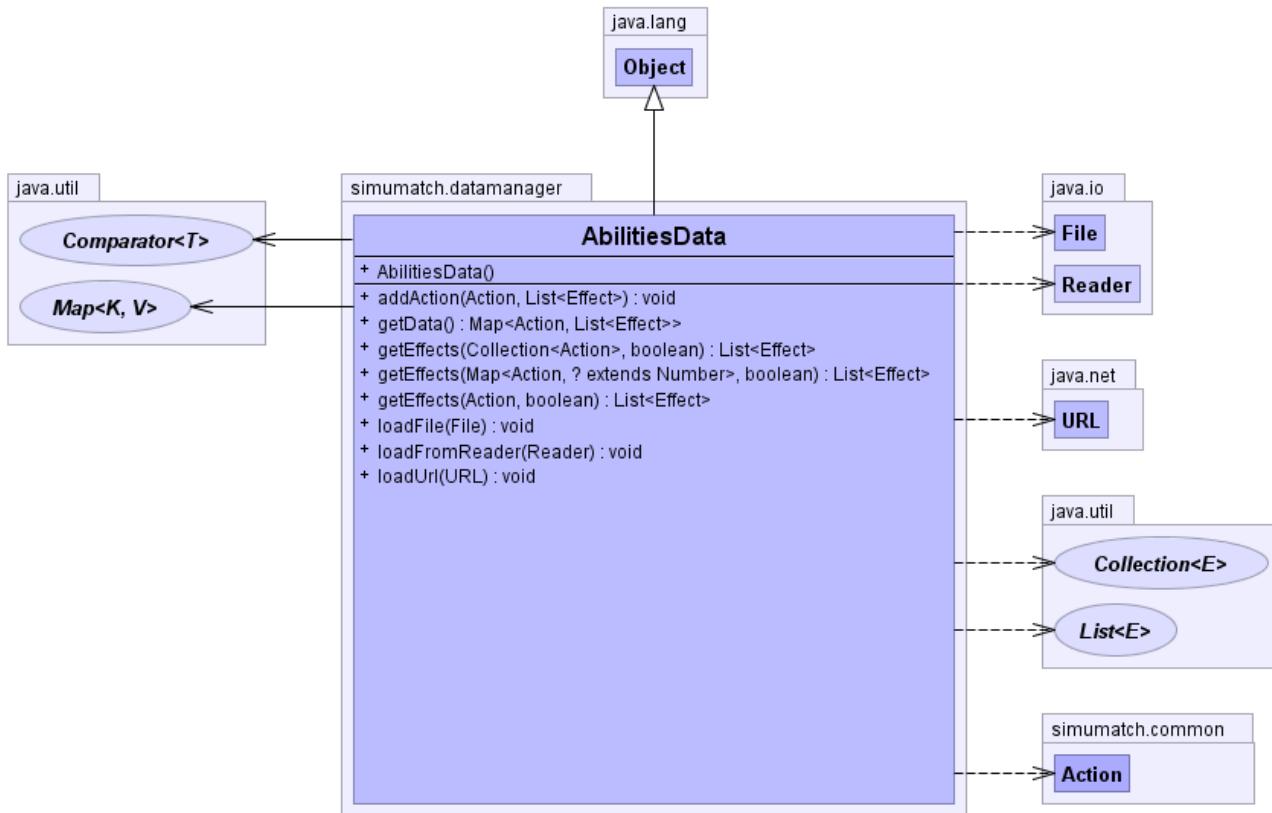
- El evento “cargar modificadores de preparativos” llamará al método initialMod para cargar los datos a engine y bloqueará el panel correspondiente.
- El evento “cargar modificaciones inmediatas” llamará al método actualMod para cargar los datos a engine.
- El evento “calcula el próximo partido” llamará al método nextMatch de SimuMatch <<que a su vez hará una llamada a engine.calculateNextmatch()>>

# JUGADOR #12

Información del juego: diagramas de clases de SimuMatch

Ingeniería del Software, 2013

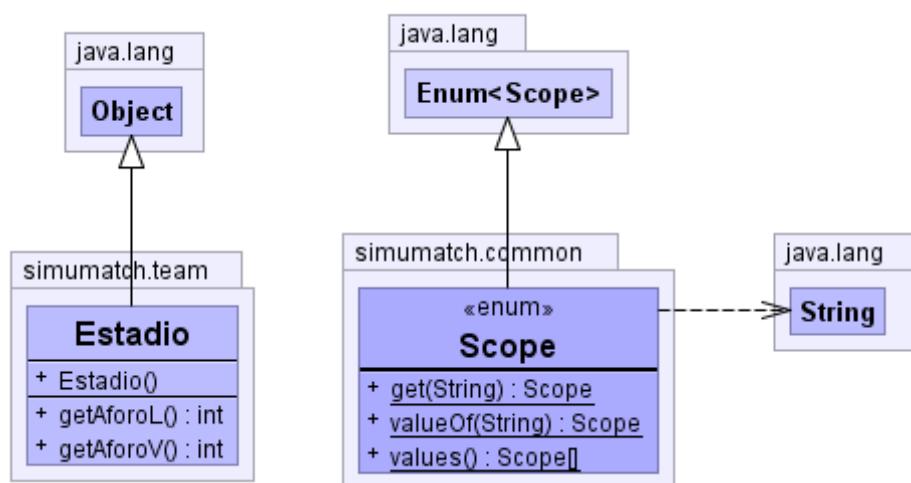
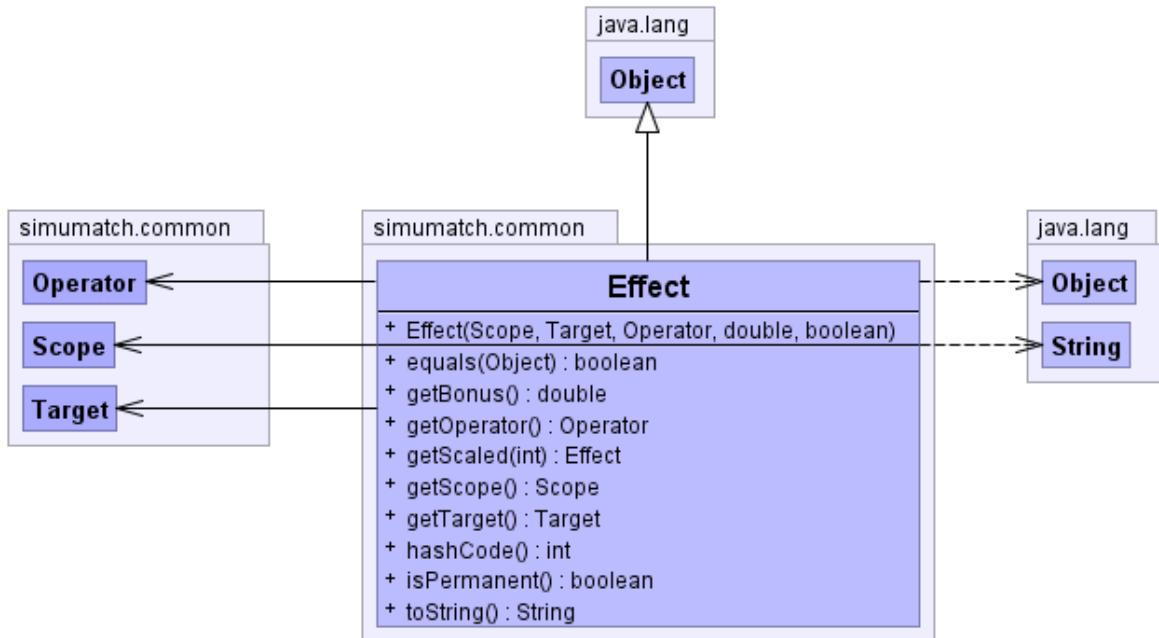
## Diagramas de clases de SimuMatch



# JUGADOR #12

Información del juego: diagramas de clases de SimuMatch

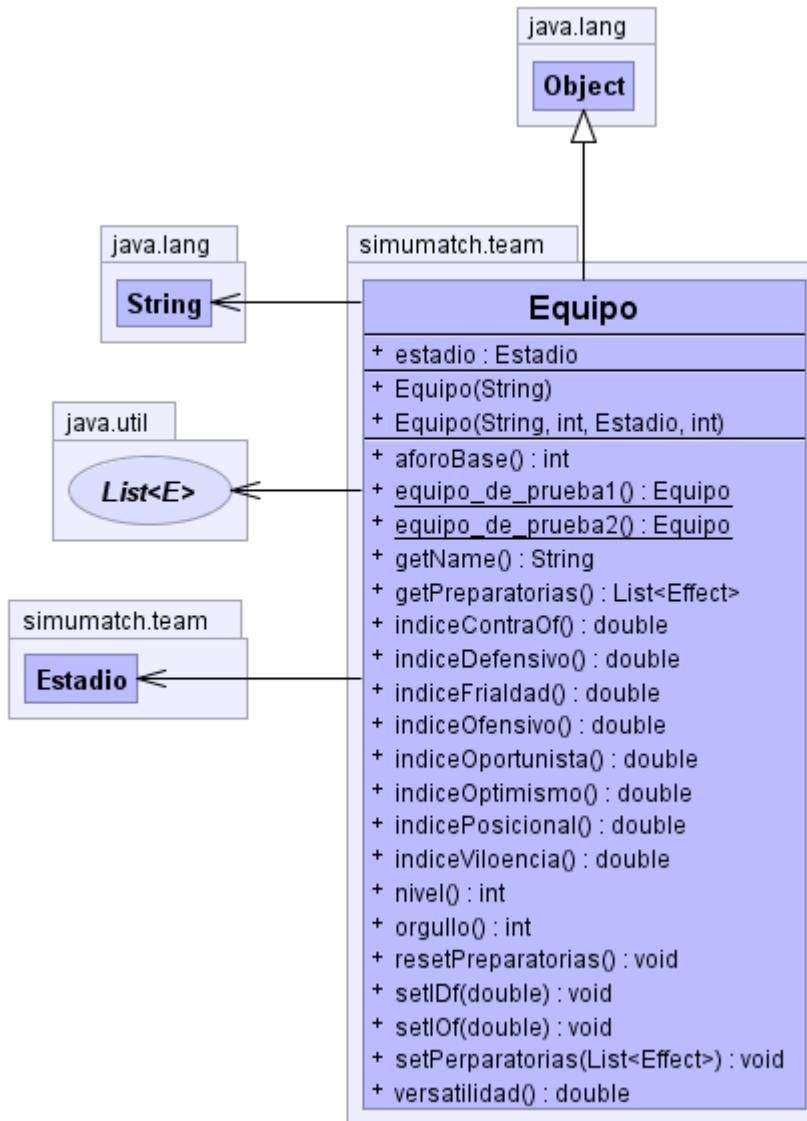
Ingeniería del Software, 2013



# JUGADOR #12

Información del juego: diagramas de clases de SimuMatch

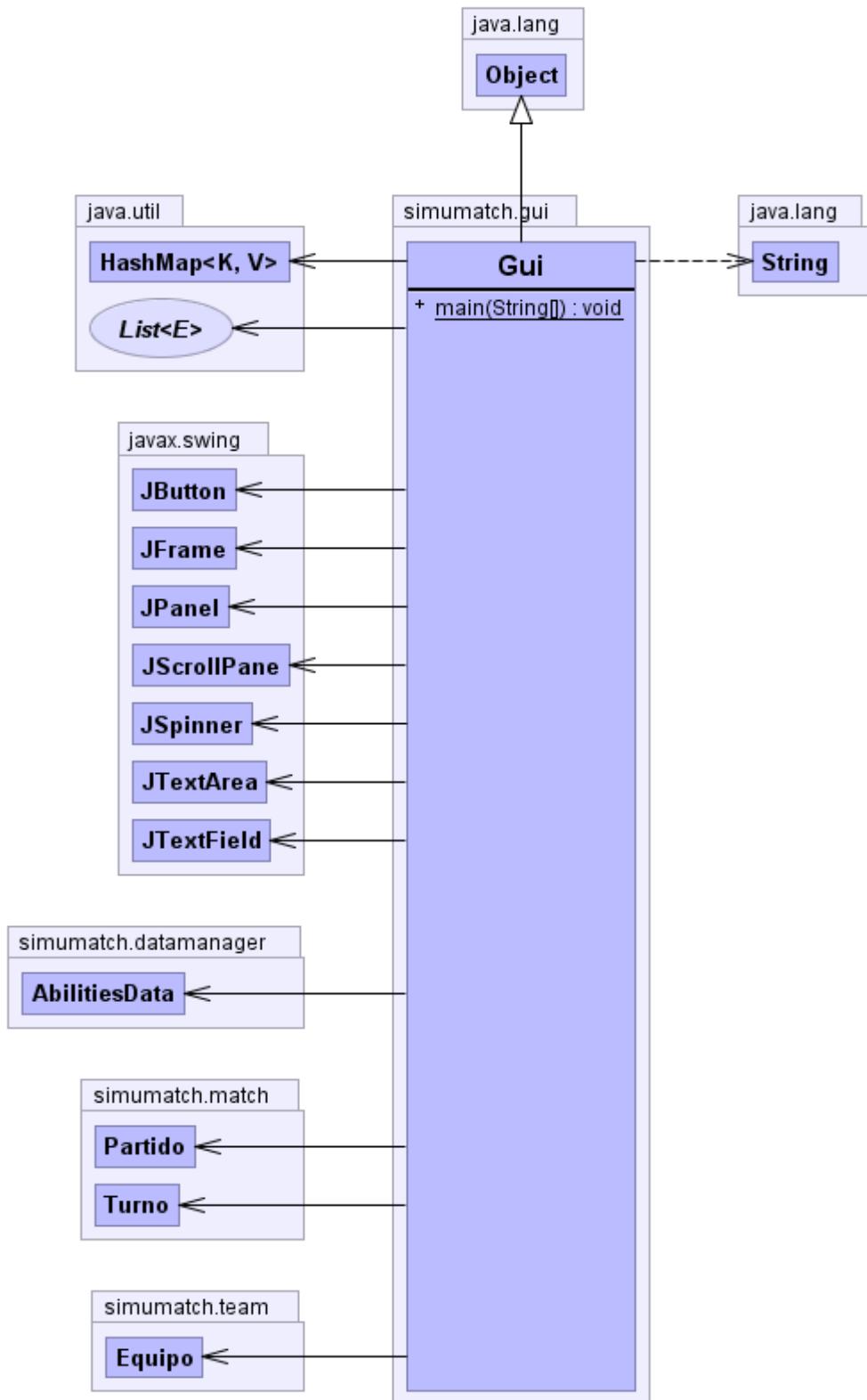
Ingeniería del Software, 2013



# JUGADOR #12

Información del juego: diagramas de clases de SimuMatch

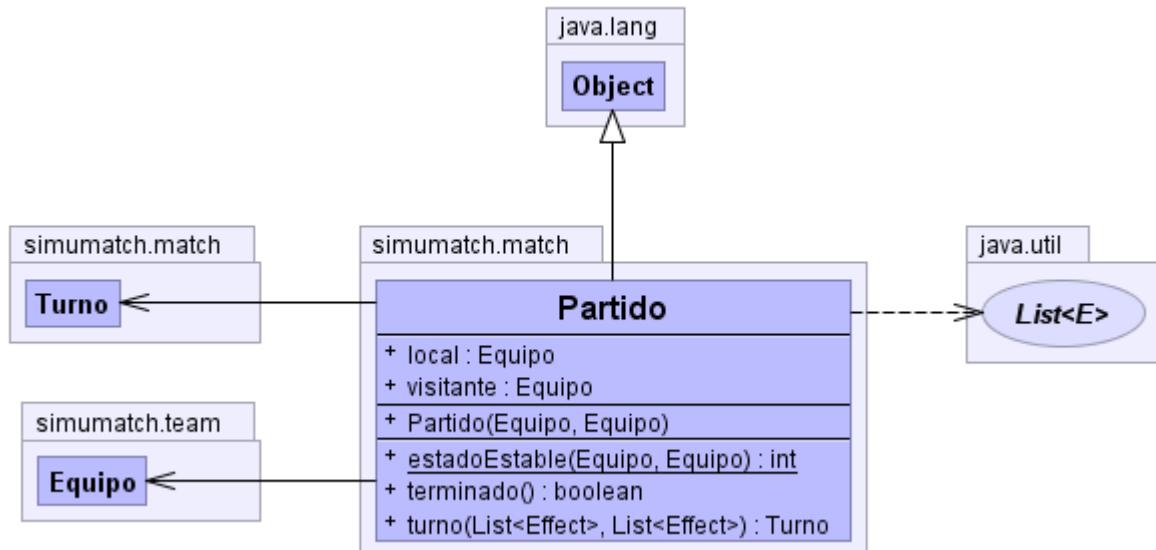
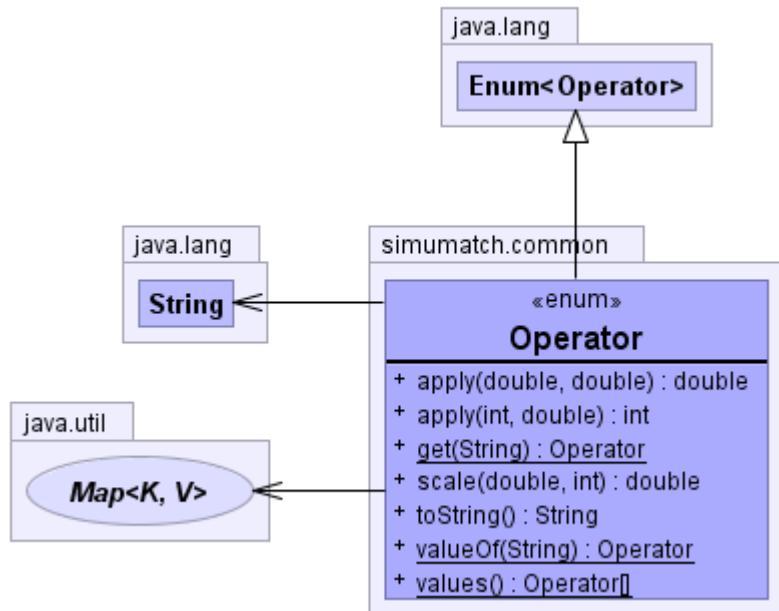
Ingeniería del Software, 2013



# JUGADOR #12

Información del juego: diagramas de clases de SimuMatch

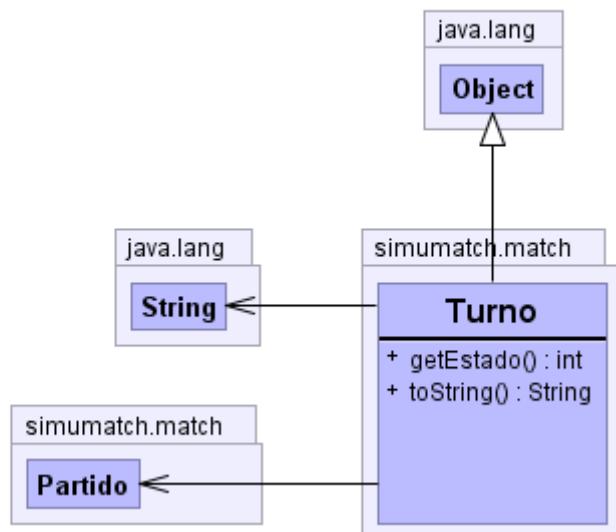
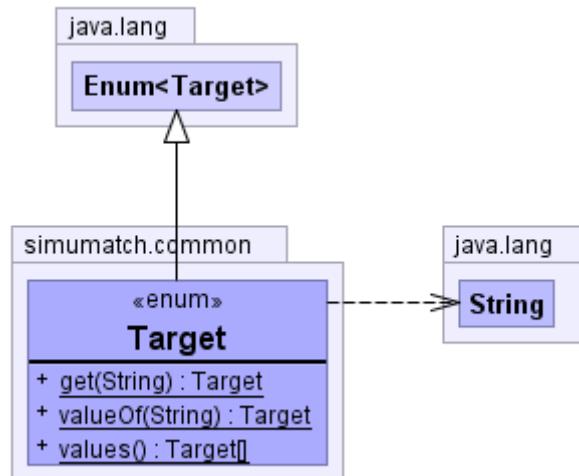
Ingeniería del Software, 2013



# JUGADOR #12

Información del juego: diagramas de clases de SimuMatch

Ingeniería del Software, 2013



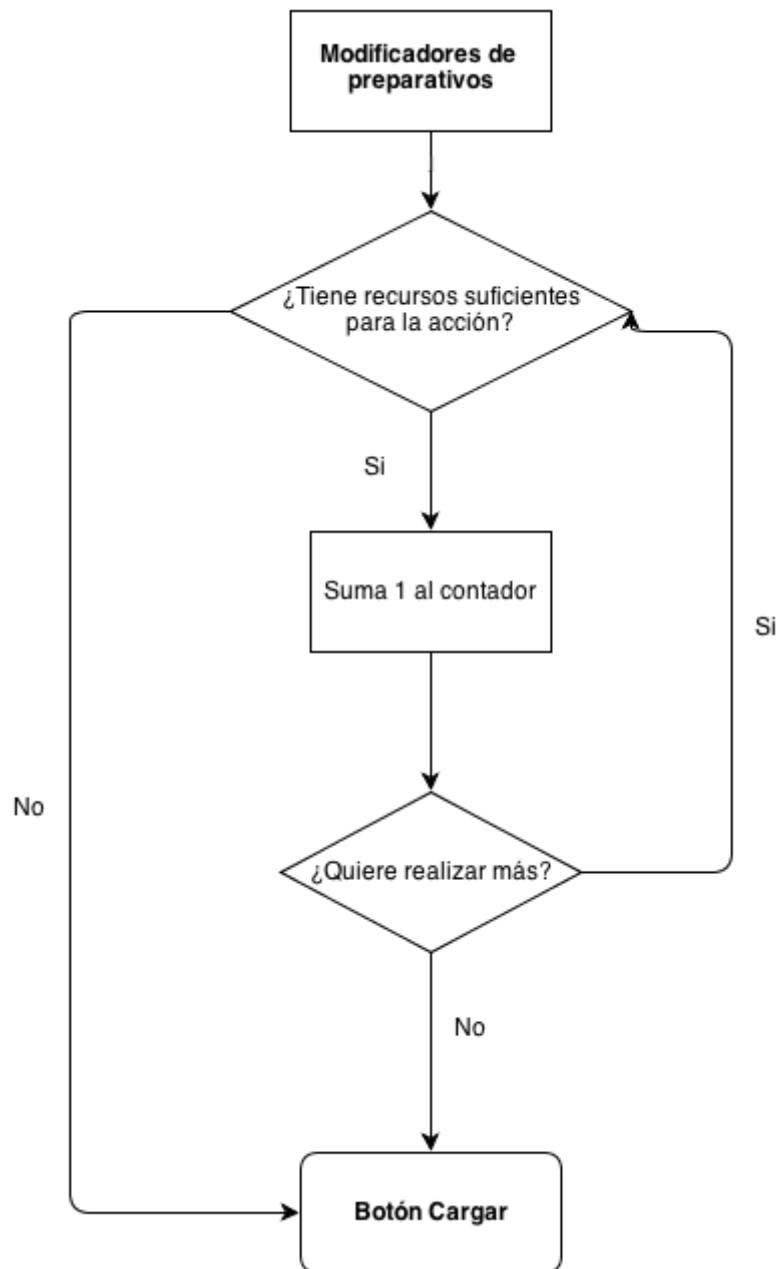
# JUGADOR #12

Información del juego: diagramas de flujo de SimuMatch

Ingeniería del Software, 2013

## Diagramas de flujo de SimuMatch

Ejecutar acción preparatoria

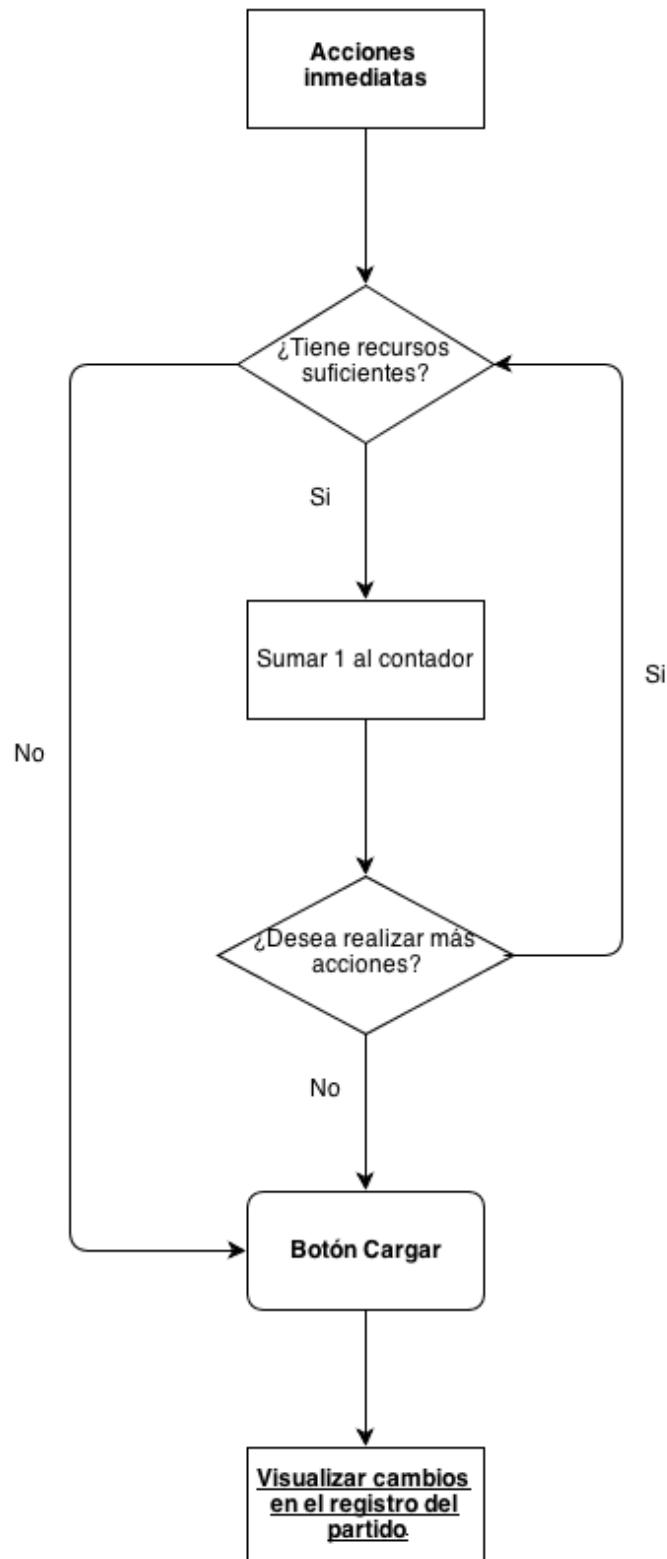


# JUGADOR #12

Información del juego: diagramas de flujo de SimuMatch

Ingeniería del Software, 2013

## Ejecutar acción inmediata

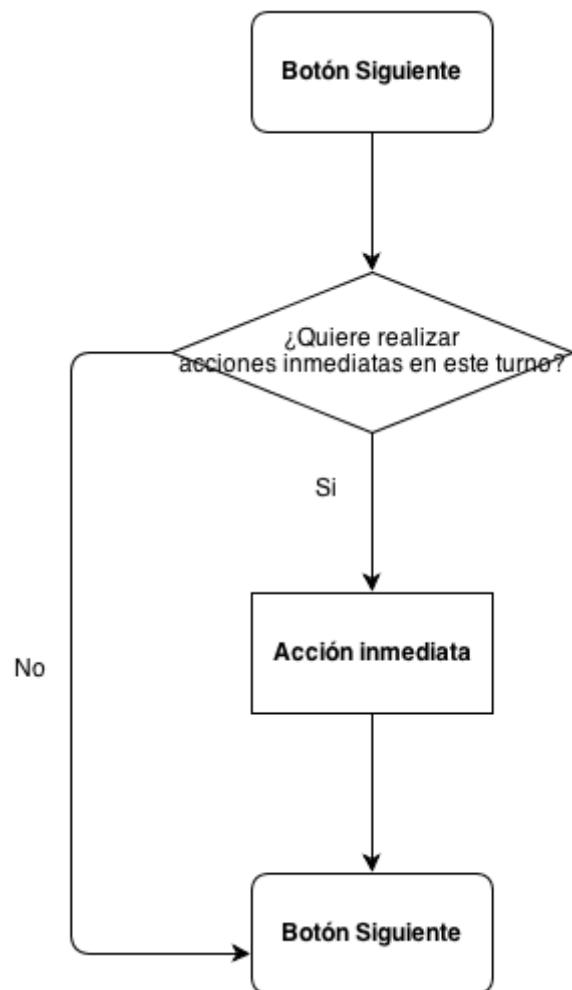


# JUGADOR #12

Información del juego: diagramas de flujo de SimuMatch

Ingeniería del Software, 2013

Siguiente turno



# JUGADOR #12

Información del juego: conclusiones de SimuMatch

Ingeniería del Software, 2013

## Introducción

Este documento recoge las conclusiones sacadas de las partidas jugadas entre miembros del grupo al prototipo de Jugador número 12.

Estas conclusiones conciernen a la dinámica del juego, y sobre todo a ideas y posibles enfoques para hacerlo más atractivo y jugable.

## Índice

1. Puntos para mejorar la jugabilidad
  - 1.1. Tiempo de preparación del partido
  - 1.2. Interacción entre miembros de una misma afición
  - 1.3. Interacción entre jugadores de distinto equipo
  - 1.4. Coste de las acciones - número de jugadores
2. Conclusiones

## 1. Puntos para mejorar la jugabilidad

### 1.1. Tiempo de preparación del partido

Un primer punto que era muy básico pero que nos tuvo ocupados nada más empezar es el tiempo necesario para que cada equipo prepare el partido. Cinco turnos por equipo nos parecía bien, pero en la práctica era demasiado. En el juego *online* este tiempo no irá en turnos, pero nos hacemos una idea del tiempo a dejar entre partidos para un equipo.

### 1.2. Interacción entre miembros de una misma afición

Otro punto que tuvimos que mejorar fue la interacción entre los miembros de una misma afición. Cuando un jugador está pensando qué recursos aportar y cuánta cantidad de su total es recomendable hablarlo con el resto de la afición para planear una acción en conjunto, por ejemplo. Aquí se reforzó la idea de necesitar un **chat** entre los jugadores.

### 1.3. Interacción entre jugadores de distinto equipo

También es necesario una **interacción** entre los equipos que se enfrentan durante el partido. Es como el póker, se juega con las cartas de uno pero también con las del resto de oponentes. Hace falta conocer un poco la estrategia del contrario para saber cómo atacarlos donde más duele. Por lo tanto serán necesarias habilidades que “destruyan” o “debiliten” al rival y le impidan mejorar

### 1.4. Coste de las acciones - número de jugadores

En relación a las habilidades, fue importante ver que la cantidad de recursos necesarios deben ir en relación al número de jugadores de la afición, ya que había algunas acciones inalcanzables para 3 jugadores.

Fue también en el prototipo cuando aumentamos el número de acciones de cada perfil de usuario, porque en un partido “largo” se acababan repitiendo las acciones.

## 2. Conclusiones

Gracias al prototipo sacamos los siguientes puntos dónde habrá que profundizar para mejorar la experiencia de juego:

1. Necesidad del chat interno
2. Problema del nivelado del coste de acciones - número de jugadores en la afición.
3. Necesidad de interacción entre jugadores de distinto equipo

# JUGADOR #12

Información del juego: reglas del juego de cartas

Ingeniería del Software, 2013

## Introducción

En este documento, quedan especificadas las reglas para jugar al prototipo del juego “Jugador número 12”.

El prototipo, combina una implementación en Java con un juego de mesa clásico. Con las reglas del juego, cualquier persona podría echar una partida al juego que hemos inventado.

## Índice

1. Objetivo
2. Preparación
3. Reglas del juego
4. Paso 1: Preparación del partido
  - 4.1. Fase 1: Obtención de dinero
  - 4.2. Fase 2: Uso de las cartas de acción
  - 4.3. Fase 3: Aporte de recursos
  - 4.4. Fase 4: Resolver las acciones
5. Paso 2: Desarrollo del partido
  - 5.1. Obtención ánimo
6. Ganar la partida

# JUGADOR #12

Información del juego: reglas del juego de cartas

Ingeniería del Software, 2013

## 1. Objetivo

Cada jugador se mete en la piel de un hincha de fútbol. Los jugadores se dividen en 2 bandos, cada uno apoya a un equipo de fútbol. Durante la partida los jugadores se preparan para el partido, ganan recursos para gastarlos animando a su equipo y finalmente asisten al partido apoyando a su equipo para ganar. Gana el grupo de jugadores cuyo equipo gane el partido.

## 2. Preparación

- Los jugadores se dividen en dos grupos, cada grupo apoyará a un equipo. Se colocan los jugadores, en torno a una mesa, alternativamente uno de cada equipo. Si son impares se echa a suertes qué equipo coloca dos jugadores seguidos.
- Cada jugador elige el perfil que desee, empresario, movedora de masas o ultra.
- A cada jugador se le reparten 5 cartas de acción correspondientes a su perfil.
- Se reparten los recursos a los jugadores:
  - Dinero: 1000
  - Ánimo: 300 al ultra, 75 a la animadora y 50 al empresario.
  - Influencia: 40 a la animadora, 12 al empresario y 6 al ultra.

## 3. Reglas del juego

La partidas de «Jugador número 12» se dividen en 2 pasos: **preparación del partido(1)** y **desarrollo del partido(2)**. Cada paso cuenta con un número de rondas determinado, 5 rondas por cada jugador en el paso 1 y 10 rondas en el paso 2. La partida avanza del paso 1 al paso 2 cuando los jugadores agotan las rondas del paso 1.

En el paso 2 cada ronda está formada por 4 fases. En cada una de estas fases, todos los jugadores realizan sus acciones siguiendo un orden de juego específico antes de continuar la partida con la fase siguiente. Las 4 fases son:

- Obtención de recursos
- Uso de las cartas de acción
- Aporte de recursos
- Resolver las acciones

## 4. Paso 1: Preparación del partido

### 4.1. Fase 1: Obtención de dinero

En esta fase, el jugador que juega la ronda obtiene recursos en base a su perfil:

- **Movedora de masas:** 100 de dinero.
- **Ultra:** 200 de dinero.
- **Empresario:** 600 de dinero.

Determinadas cartas de acción pueden aumentar, para algún recurso, el número de unidades que se reciben. En ese caso jugar como indique la carta.

## JUGADOR #12

Información del juego: reglas del juego de cartas

Ingeniería del Software, 2013

### 4.2. Fase 2: Uso de las cartas de acción

En esta fase, el jugador puede, pagando los recursos que indique en la carta bajar una carta de acción de su mano. Las cartas de acción pueden ser de dos tipos, instantáneas y de peña.

- **Acciones instantáneas:** El jugador recibe el beneficio indicado en la carta de manera automática. Si el efecto se aplica a las siguientes rondas dejar la carta sobre la mesa delante del jugador para que todos puedan verla. La influencia invertida al pagar estas acciones se recuperará cuando la acción finalice. La mayoría de las acciones finalizarán cuando finalice el juego, es decir, que no se recuperará la influencia y que el beneficio será permanente para toda la partida. .
- **Acciones de peña:** El jugador crea un evento, en el que los demás jugadores de su equipo pueden participar. Estas acciones tienen un número de rondas que estarán activas. También tienen un número de recursos que han de ser aportados para que se resuelva la acción. Si los jugadores aportan los recursos necesarios antes de que la acción se agote ganarán el beneficio que indique. La influencia invertida para apoyar estas acciones se recuperará cuando se resuelva la acción.  
Las acciones se colocarán encima de la mesa a la vista de todos los jugadores.

### 4.3. Fase 3: Aporte de recursos

En esta fase, el jugador puede aportar recursos a las acciones de su peña que estén activas. Puede repartir los recursos entre las acciones que quiera no hay límite. Tampoco hay un mínimo de aportación.

### 4.4. Fase 4: Resolver las acciones

En esta fase, se resuelven las acciones de peña. Se quita un contador a todas las acciones activas. Se resuelven de dos maneras. Aquellas que han conseguido todos los recursos necesarios, se apartan y se anotan los beneficios. Las que han agotado el contador sin conseguir los recursos se retiran sin anotar el beneficio. El dinero y el ánimo invertido se pierden. La influencia invertida se recupera.

## 5. Paso 2: Desarrollo del partido

En esta paso se simula el desarrollo del partido en el que se enfrentan los dos equipos en el campo de juego. El paso tiene 10 rondas. Cada ronda dura 30 segundos en el que los jugadores pueden pensarse que cartas de su mano jugar. Las jugarán boca abajo y las dejarán delante suyo encima de la mesa. Pasados los 30 segundos se levantarán las cartas y se introducirán en el programa que calculará el resultado del partido.

### 5.1. Obtención ánimo

Al acabar el partido los aficionados ganarán o perderán ánimo en función del resultado de su equipo.

Si su equipo gana, ganan el siguiente ánimo:

- **Movedora de masas:** 75 de dinero.

## JUGADOR #12

Información del juego: reglas del juego de cartas

Ingeniería del Software, 2013

- **Ultra:** 300 de dinero.
- **Empresario:** 50 de dinero.

Si su equipo pierde, ganan el siguiente ánimo:.

- **Movedora de masas:** 40 de dinero.
- **Ultra:** 150 de dinero.
- **Empresario:** 25 de dinero.

### 6. Ganar la partida

La partida termina cuando se acaban todas las rondas del desarrollo del partido. Ganará la peña cuyo equipo gane el partido.

Se puede jugar un torneo a varios partidos. En ese caso ganará el equipo que gane el torneo.

# JUGADOR #12

Información del juego: detalles del juego de cartas

Ingeniería del Software, 2013

## REPARTO DE RECURSOS ENTRE PERFILES

DINERO	Base	Turno	Nº turno	Total
Empresario	1000	600	5	4000
Movedora	1000	100	5	1500
Ultra	1000	200	5	2000

ÁNIMO	Base	Máximo	Si gana	Si pierde
Empresario	50	100	50	25
Movedora	75	150	75	40
Ultra	300	600	300	150

INFLUENCIA	Base
Empresario	12
Movedora	40
Ultra	6

## SIMULADOR DE TURNOS

	DINERO	ÁNIMO	INFLUENCIA	TURNO
JUGADOR 1 (U)	1400	1500	6	2
JUGADOR 2 (E)	4600	650	12	6
JUGADOR 3 (A)	1600	975	40	6
JUGADOR 4 (E)	1000	75	40	0

GASTO	DINERO	ÁNIMO	INFLUENCIA
JUGADOR 1	0	0	0
JUGADOR 2	0	0	0
JUGADOR 3	0	0	0
JUGADOR 4	0	0	0

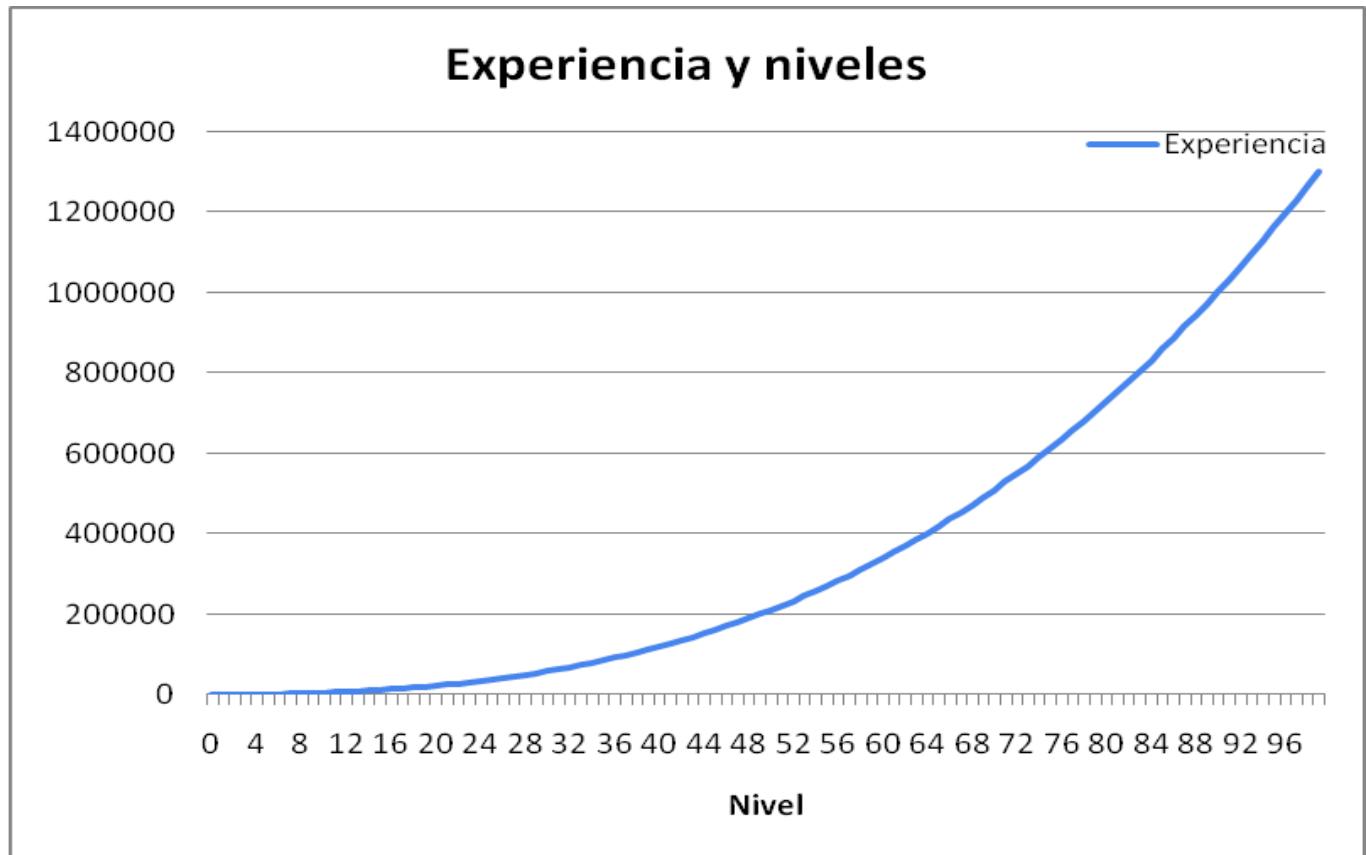


## JUGADOR #12

Información del juego: simulación de experiencia

Ingeniería del Software, 2013

### Simulación de experiencia y niveles

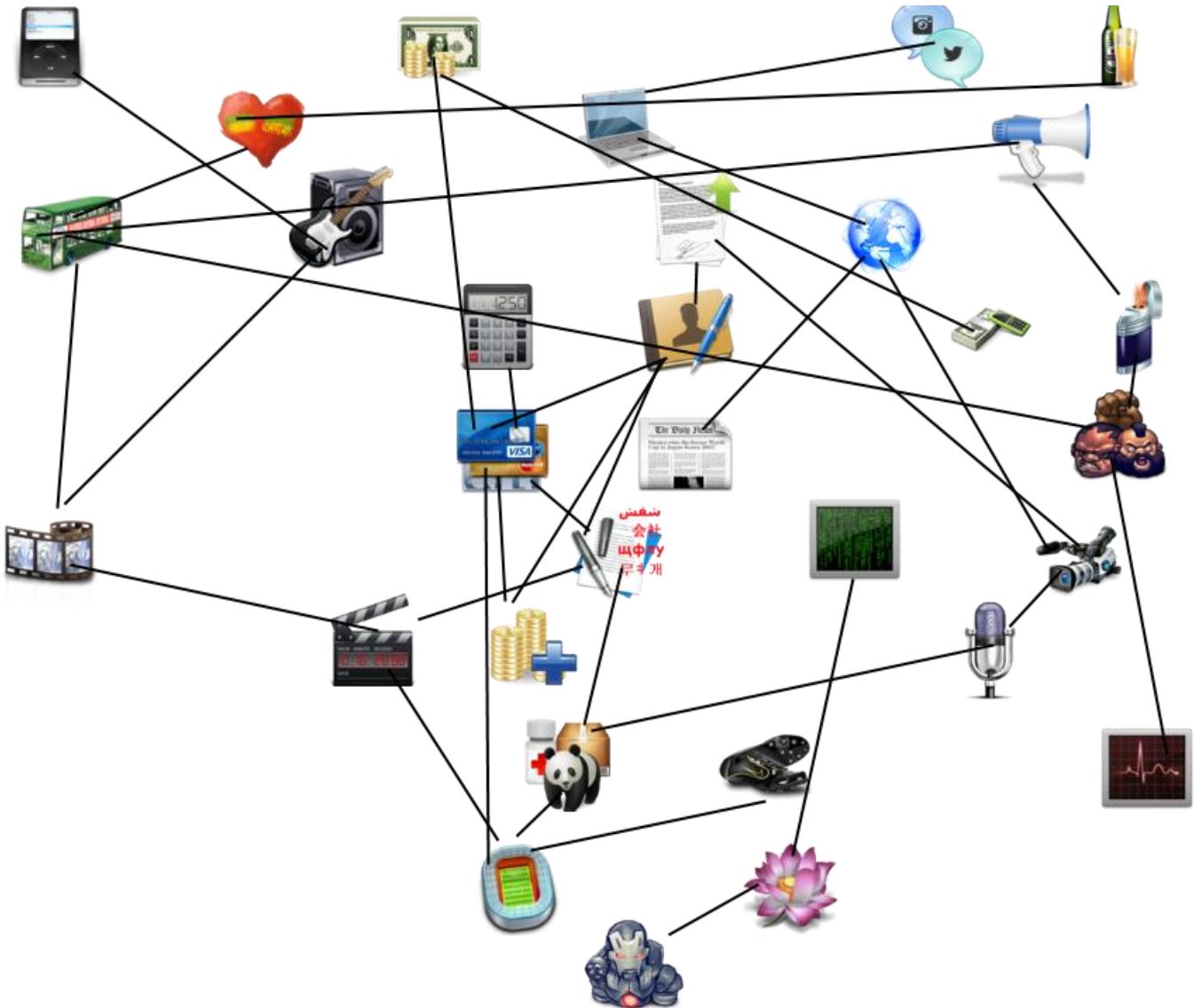


JUGADOR #12

## Información del juego: relación de habilidades

Ingeniería del Software, 2013

## **Relaciones entre habilidades del juego**



# JUGADOR #12

Documentación técnica: ERS

Ingeniería del Software, 2013

## Introducción

El objetivo de este documento es presentar con detalle todo lo relacionado con la especificación de nuestro proyecto de Ingeniería del Software. Mediante este documento toda la funcionalidad, limitaciones de nuestro sistema, posibles ampliaciones y herramientas utilizadas, quedan reflejadas de manera formal para que puedan ser revisadas por el profesor que guía el proyecto.

## Índice

1. Visión general del documento
  - 1.1. Ámbito del sistema
  - 1.2. Definiciones, Acrónimos y Abreviaturas
    - 1.2.1. Definiciones
  - 1.3. Referencias
2. Descripción general
  - 2.1. Perspectiva del producto
  - 2.2. Funciones del producto
  - 2.3. Características de los usuarios
  - 2.4. Restricciones
  - 2.5. Suposiciones y dependencias
  - 2.6. Requisitos futuros
3. Requisitos específicos
  - 3.1. Interfaces externas de usuario
  - 3.2. Funciones
    - 3.2.1. Requisitos funcionales para un usuario invitado
    - 3.2.2. Requisitos funcionales para un usuario autenticado
    - 3.2.3. Requisitos funcionales para un usuario administrador
  - 3.3. Requisitos de rendimiento
  - 3.4. Atributos del sistema
    - 3.5.1. Seguridad
    - 3.5.2. Portabilidad
    - 3.5.3. Mantenibilidad
    - 3.5.4. Fiabilidad
4. Apéndices
  - 4.1. Modelo RUP

# JUGADOR #12

Documentación técnica: ERS

Ingeniería del Software, 2013

## 1. Visión general del documento

En esta sección vamos a dar una perspectiva general de la estructura del documento, el cual está dividido en cuatro secciones principales:

1. **Introducción**, en ella se comenta el propósito del sistema, su ámbito, la definición de los conceptos que se utilizan en él y los documentos externos que referenciamos.
2. **Descripción general**, en esta sección se explican todos los factores que afectan al producto y a sus requisitos.
3. **Requisitos específicos**, contiene los requisitos a un nivel de detalle suficiente como para permitir a los diseñadores diseñar un sistema que los satisfaga.
4. **Apéndice**, contiene todo tipo de información relevante para la especificación pero que, propiamente, no forma parte de ésta.

### 1.1. Ámbito del sistema

El sistema va a ser un juego por navegador el cual hemos denominado Jugador Número 12. Éste pondrá al usuario en la piel de un hincha de fútbol, el cual tendrá la posibilidad nada más entrar de seleccionar su perfil dentro del juego, pudiendo elegir entre los siguientes:

- *El ultra*, que con su espíritu conformará la primera línea de apoyo al equipo.
- *La animadora de masas*, que intentará conseguir llenar los estadios.
- *El empresario*, que representa las luchas en los despachos manejando grandes cantidades de dinero.

Al usuario se le dará la opción de unirse como aficionado a un equipo.

Una vez hecho esto el jugador podrá realizar acciones para preparar el partido, individuales o que hayan sido iniciadas por integrantes de su afición. Para poder realizarlas dispondrá de tres recursos:

- *El ánimo*, o capacidad que tiene un hincha para animar.
- *La influencia*, que refleja el renombre del hincha en la sociedad.
- *El dinero*, que se irá incrementando con el tiempo y permitirá comprar diversos objetos.

Pero no sólo habrá acciones que preparen el partido, sino que el usuario podrá participar en el encuentro, es decir, se le dará la posibilidad de influir en la reacción del equipo durante el transcurso de éste haciendo que sea capaz de meter un gol o reponerse de un tanto del contrario. Para todo esto, el partido dispondrá de un sistema de turnos, y para poder participar en él, el usuario deberá comprar una entrada.

¿Qué pretendemos con este sistema? Tras la presentación del proyecto en mayo pretendemos refinar y modificar el proyecto para subirlo a la web con el fin de que toda la gente que le guste juegue y así nosotros poder sacar beneficio económico de ello. Pero no

# JUGADOR #12

Documentación técnica: ERS

Ingeniería del Software, 2013

solo pensamos en este tipo de beneficio ya que realizar este proyecto nos aportará experiencia en el ámbito del desarrollo de aplicaciones web.

## 1.2. Definiciones, Acrónimos y Abreviaturas

En esta sección vamos a explicar las definiciones, acrónimos y abreviaturas utilizadas en el documento:

### 1.2.1. Definiciones

- **Hincha:** Un hincha es el partidario entusiasta de un equipo de fútbol o en general de cualquier deporte.
- **Rol o perfil:** Papel que desempeña el usuario dentro del juego, pudiendo elegir entre tres diferentes.

## 1.3. Referencias

1. Marina Bezares, “Documento definición del juego”

## 2. Descripción general

### 2.1. Perspectiva del producto

Jugador Número 12 será un producto final e independiente de cualquier otro. Por ello no es necesario mostrar ninguna relación y bastará con la documentación restante del documento para su definición completa.

### 2.2. Funciones del producto

Mostraremos a continuación, en diferentes secciones, una visión a grandes rasgos de las funcionalidades que soportará el producto:

- **Mantenimiento de cuentas de usuario:** el sistema será capaz de almacenar y modificar los datos personales de los usuarios del mismo.
- **Comunicación interna:** permitirá a los usuarios y al propio sistema el envío, almacenamiento y recepción de mensajes privados.
- **Asignación del rol del usuario:** como ultra, empresario o movedora de masas, el sistema permitirá al usuario asumir dicho papel con sus correspondientes funcionalidades, acciones y atributos específicos.
- **Gestión de aficiones:** el producto admitirá la gestión de varias aficiones de fútbol por parte de los usuarios con diversas funcionalidades como añadir nuevos usuarios o crear eventos para los miembros asociados.
- **Facilitar información de contexto:** el sistema ofrecerá la capacidad de mostrar datos representativos para los usuarios como puede ser la clasificación de los equipos en un determinado momento.
- **Participación y resolución de partidos:** el producto permitirá a los usuarios participar mediante acciones en los diferentes encuentros y será capaz de generar el resultado de un partido virtual en función de los aportes durante y antes del mismo.

## JUGADOR #12

Documentación técnica: ERS

Ingeniería del Software, 2013

### 2.3. Características de los usuarios

Los requisitos mínimos que un usuario deberá cumplir para poder hacer uso del sistema “Jugador Número 12” son:

- **Nivel educacional:** el producto está destinado a un público general con un mínimo de educación básica y lingüística.
- **Experiencia:** los usuarios no necesitarán conocimientos previos acerca de juegos por navegador o estrategia para poder hacer uso del sistema. Se tratará de implementar la interfaz más sencilla posible para acercar el producto al mayor número de usuarios posible.
- **Experiencia técnica:** todo usuario del sistema deberá tener un mínimo de conocimientos sobre internet y navegadores web.

### 2.4. Restricciones

A continuación se presentará una lista de restricciones impuestas por los propios desarrolladores del producto:

- **Limitaciones de hardware:** inicialmente, el sistema estará pensado para funcionar en un servidor web de nivel básico/medio consumiendo el mínimo de recursos posibles para permitir el mayor número de usuarios simultáneos.
- **Operaciones paralelas:** el sistema debe ser capaz de administrar las transacciones concurrentes a la base de datos de forma correcta.
- **Funciones de auditoría:** se realizarán pruebas mínimas sobre el sistema para verificar la integridad y protección de los datos. Por ejemplo, comprobación de inyección SQL, ataques a la web, etc.
- **Funciones de control:** a cargo de varios supervisores estará la revisión de elementos como el repositorio, fechas de entrega, documentación, código, etc.
- **Lenguajes de programación:** se hará uso de las tecnologías de HTML, PHP, CSS, Javascript y MySQL. Todo ello se hará mediante el uso del framework Yii.
- **Requisitos de habilidad:** los miembros del grupo deberán tener unos conocimientos mínimos sobre el uso del framework y el repositorio que serán adquiridos durante una sesión didáctica.
- **Consideraciones acerca de la seguridad:** el sistema deberá incluir gran cantidad de pruebas sobre seguridad para prevenir la mayor cantidad de ataques posibles y mejorar la protección de datos de los usuarios.

### 2.5. Suposiciones y dependencias

Se han supuesto los requisitos para que el sistema trabaje correctamente con especificaciones estándar: resolución de pantalla media, navegadores actualizados en mayor o menor medida, etc. Es posible tratar de ampliar la capacidad de mercado optimizando la compatibilidad del sistema con navegadores antiguos aunque pueda suponer algún cambio en los requisitos.

## 2.6. Requisitos futuros

En un futuro se tratarán de implementar nuevas funcionalidades que requerirán una modificación de los requisitos iniciales:

- **Chat interno al juego:** para permitir la comunicación de los usuarios en tiempo real.
- **Aplicación móvil del sistema:** para aumentar la capacidad de negocio del mismo y permitir una mayor accesibilidad por parte de los usuarios.

## 3. Requisitos específicos

### 3.1. Interfaces externas de usuario

Las interfaces de usuario son las ventanas (formularios) con las que debe interactuar el usuario para realizar una operación determinada. Dicha manipulación será realizada por medio del teclado y el ratón.

Las interfaces de Jugador número 12 serán ventanas de navegador de Internet para ordenadores. Para poder jugar el usuario deberá de disponer de un navegador que soporte Javascript y CSS.

Probaremos el sistema para que funcione correctamente en, al menos, los navegadores más comunes: Chrome y Firefox.

### 3.2. Funciones

En “Jugador Número 12” distinguiremos entre 3 tipos de usuarios: invitado, autenticado y administrador. Cada uno de ellos tendrá asociados distintos niveles de privilegios, definidos en la sección 3.5.1, y por tanto, el comportamiento externo del sistema para cada uno de ellos será diferente.

#### 3.2.1. Requisitos funcionales para un usuario invitado

Un usuario invitado es aquel que no está registrado en el juego, es decir, que no ha hecho login en el sistema con una cuenta previamente creada. Este tipo de usuario solo tendrá acceso a la página principal del juego y a los tutoriales. No podrá acceder a la zona de juego. La única función que puede llevar a cabo en el sistema es registrarse en él. Este requisito funcional forma parte de la arquitectura del sistema.

#### 3.2.2. Requisitos funcionales para un usuario autenticado

Un usuario autenticado es aquel que ha hecho login en el sistema con una cuenta creada con anterioridad. Este tipo de usuario puede acceder a la zona de juego así como a sus datos personales.

Tendrá los siguientes requisitos funcionales, de **gestión de usuario**, a su disposición:

- **Consultar datos de usuario.** Los usuarios podrán consultar en una página llamada “mi perfil” sus datos de usuario introducidos. Éstos no serán compartidos, ni podrán ser vistos por ningún otro usuario. —Véase seguridad.— Los datos que podrán consultar son: nombre, usuario, contraseña, e-mail registrado, personaje escogido y afición escogida. Este requisito funcional forma parte de la arquitectura.

## JUGADOR #12

Documentación técnica: ERS

Ingeniería del Software, 2013

- **Modificar datos de usuario.** Los usuarios tendrán la posibilidad de cambiar sus datos a través de la página de perfil. Podrán modificar aquellos datos indicados en el párrafo anterior, excepto el personaje. Si un usuario cambia la contraseña, se le pedirá que introduzca la contraseña actual antes de ingresar la nueva contraseña. Este requisito funcional no forma parte de la arquitectura.
- **Mensajería privada.** Un usuario registrado podrá usar el servicio de mensajería privada. Será capaz de mandar un mensaje a otro usuario, que solo lo podrán ver los jugadores implicados. También podrá recibir mensajes privados de otros usuarios. Para mandar un mensaje privado a otro usuario hace falta saber el nombre del destinatario. Podrá obtener un listado de los mensajes recibidos y enviados para consultarlos cuando quiera. Podrá también borrar un mensaje — o varios — del listado. Este requisito funcional no forma parte de la arquitectura.
- **Darse de baja.** Un usuario podrá, en su perfil, dar de baja su cuenta. Ésta se borrará y el usuario dejará de existir. Sus datos serán borrados de la base de datos. Este requisito forma parte de la arquitectura.

Tendrá los siguientes requisitos funcionales, de **gestión de hincha**, a su disposición:

- **Elegir el personaje.** Cada usuario podrá elegir entre 3 personajes<sup>1</sup>: animadora, empresario y ultra. Cada uno de ellos tendrá una forma de juego distinta. Al registrarse, el usuario deberá seleccionar qué personaje desea, y una vez elegido no lo podrá cambiar. Este requisito funcional forma parte de la arquitectura.
- **Consultar datos del personaje.** Los usuarios podrán consultar los datos relativos a su personaje, que serán el tipo de personaje elegido y recursos que posee. También podrán consultar su nivel y habilidades, pero lo hemos separado en otro epígrafe. Ésto se hará desde la página “mi perfil”. Este requisito funcional forma parte de la arquitectura.
- **Consultar habilidades, nivel y experiencia.** El personaje del jugador obtendrá experiencia. Cuando consiga una determinada cantidad de experiencia subirá de nivel. El usuario podrá consultar su nivel actual, la experiencia actual y cuánta experiencia le falta para llegar al siguiente nivel.  
Cuando suba de nivel, el jugador podrá elegir diferentes habilidades<sup>1</sup> para mejorar a su personaje. El usuario podrá consultar el árbol de habilidades total. Podrá saber qué habilidades cogidas del árbol tiene y cómo le afectan. Este requisito funcional forma parte de la arquitectura.
- **Elegir habilidades.** Cuando un jugador suba de nivel podrá elegir una nueva habilidad de árbol de habilidades. El jugador podrá saber qué habilidades tiene escogidas, cuales tiene bloqueadas y las que podrá escoger. Este requisito funcional forma parte de la arquitectura.
- **Ejecutar acciones individuales.** El jugador tendrá un abanico de acciones que puede ejecutar antes del partido. Éste, será más grande, cuanto más nivel tenga el personaje. El jugador podrá, gastando sus recursos, activar acciones disponibles de su abanico.  
Las acciones que sean de efecto inmediato y permanente no se podrán volver a

## JUGADOR #12

Documentación técnica: ERS

Ingeniería del Software, 2013

ejecutar, es decir, serán de un solo uso. Este requisito funcional forma parte de la arquitectura.

Tendrá las siguientes funcionalidades a nivel de **gestión de afición**:

- **Consultar datos generales de afición.** Los jugadores podrán visualizar los datos correspondientes a su afición como pueden ser la descripción y escudo del equipo asociado. Este requisito funcional forma parte de la arquitectura.
- **Unirse a una afición.** Los usuarios podrán unirse a una afición una vez creada su cuenta en el sistema y tras elegir el perfil de jugador deseado. Este requisito funcional forma parte de la arquitectura.
- **Cambiar de afición.** En caso de no estar satisfecho, a un usuario se le permitirá cambiar la afición a la que está asociado y pasar a colaborar con otra. Este requisito funcional no forma parte de la arquitectura.
- **Consultar miembros.** Los jugadores podrán consultar en todo momento los miembros asociados a una afición. Este requisito funcional forma parte de la arquitectura.
- **Consultar acciones de afición.** Un usuario tendrá la posibilidad de consultar las acciones activas de afición. Estas acciones, a diferencia de las individuales, serán creadas por un miembro de la afición y permitirán la participación del resto de componentes de la misma en ellas. Este requisito funcional forma parte de la arquitectura.
- **Participar en acciones de afición.** Una vez revisadas las acciones de afición, cada jugador podrá participar - haciendo uso de sus recursos - en aquellas que elija y hayan sido creadas por alguien de su misma afición. Este requisito funcional forma parte de la arquitectura.
- **Consultar clasificación.** Los usuarios tendrán la capacidad de consultar la clasificación de la liga para comprobar el estado de los equipos en cualquier instante. Este requisito funcional forma parte de la arquitectura.

A nivel de **gestión de partido** un usuario autenticado podrá:

- **Consultar información previa del partido.** Esto es, un jugador podrá visualizar datos como pueden ser el ambiente, nivel de equipos y aforo previsto para el encuentro antes de que éste tenga lugar.
- **Comprar entrada.** Los usuarios tendrán la posibilidad de comprar la entrada para asistir a un encuentro. En caso de no hacerlo, no podrán participar durante el mismo.
- **Asistir al partido.** Los jugadores podrán acudir al encuentro y seguir los sucesos del mismo.
- **Realizar acciones de partido.** Habiendo cumplido el requisito de comprar la entrada, los jugadores podrán participar durante un encuentro realizando diversas acciones. Esto influirá en el resultado del partido.

### 3.2.3. Requisitos funcionales para un usuario administrador

Un usuario administrador es aquel que tiene una mayor capacidad de acceso y modificación de datos del sistema a los cuales un usuario de menor rango no puede acceder con total

## JUGADOR #12

Documentación técnica: ERS

Ingeniería del Software, 2013

libertad. Este tipo de usuario podrá hacer uso de todo lo mencionado con anterioridad y además poseerá ciertas funcionalidades extra.

- A nivel de **gestión de usuarios**: El administrador podrá crear, modificar y borrar los datos de cualquier usuario del sistema.
- A nivel de **gestión de hinchas**: El usuario tendrá la capacidad de crear, modificar o eliminar los datos del personaje asociado a cualquier usuario.
- A nivel de **gestión de aficiones**: Cualquier administrador podrá crear, modificar o borrar los datos de una afición como pueden ser las acciones activas o los miembros asociados.
- A nivel de **gestión de ligas**:  
Administrador la liga. Los administradores tendrán la posibilidad de crear, modificar o eliminar cualquier dato referente a la liga. Esto es, por ejemplo, cambiar la posición de un determinado equipo en la clasificación.  
Administrador los equipos. Este tipo de usuarios podrán crear, modificar y borrar cualquier equipo y los datos referentes al mismo.

### 3.3. Requisitos de rendimiento

A continuación se muestran algunos datos facilitados por la compañía creadora del juego de navegador Hattrick. Es información muy básica (ya que el resto está destinada sólo a administradores) pero suficiente para obtener un modelo de requisitos de rendimiento básicos.

En el caso de Hattrick cuentan con unos 600000 usuarios activos en el sistema y 660000 logins (285000 de los cuales son únicos). En cuanto a la base de datos, reciben alrededor de 3500 escrituras por segundo y una media de 660 conexiones diarias (parte de las cuales permanecen abiertas durante el día completo).

En base a los datos recopilados, y de cara al primer ciclo de desarrollo del sistema, se creará una aplicación que permita manejar como máximo unos 100 usuarios con 50 escrituras por segundo a la base de datos y una media de 25 conexiones diarias como mucho. Con esta versión se obtendrá un producto prototípico a pequeña escala para probar la estabilidad y funcionamiento del sistema. En futuros ciclos de desarrollo se mejorará el producto para soportar una cantidad mucho mayor de usuarios y conexiones.

### 3.4. Atributos del sistema

#### 3.5.1. Seguridad

Cada usuario al registrarse deberá proporcionar un nombre de usuario único entre los ya registrados (login), una contraseña o password y su correo electrónico. Además deberá escoger una pregunta de una lista predefinida e insertar su respuesta a la misma. En caso de olvidar su contraseña, siempre podrá optar a cambiarla contestando correctamente a la pregunta escogida en el proceso de registro.

Se hará que aquellos datos que requieran un mayor grado de confidencialidad sean almacenados en la base de datos con su correspondiente encriptamiento (como puede ser

## JUGADOR #12

Documentación técnica: ERS

Ingeniería del Software, 2013

el caso de las contraseñas) para evitar la sustracción de los mismos por algún ataque al sistema.

El sistema de login e identificación de usuarios permitirá también restringir el acceso de los mismos a ciertos datos. Por ejemplo, se impedirá que un usuario pueda modificar los datos personales de otro y se negarán capacidades administrativas, como modificar datos del sistema, a los clientes del mismo.

Se incluirá también una capa de seguridad a nivel de implementación para evitar el mayor número de ataques posible. Esto es, por ejemplo, revisar y evitar inyección SQL en todo momento, evitar guardar datos en cookies sin su correspondiente encriptación, etc.

### 3.5.2. Portabilidad

El sistema será inmediatamente portable entre distintos ordenadores ya que bastará con tener un navegador más o menos actualizado para poder hacer uso de él. Con respecto a otras plataformas, la que mayor viabilidad ofrece de cara a la portabilidad son los sistemas móviles ya que para ellos se puede realizar una adaptación del producto y permitir una correcta integración con los mismos.

### 3.5.3. Mantenibilidad

El sistema será altamente mantenible, es decir, la duración y el esfuerzo requeridos por las funciones de mantenimiento (de la base de datos y del servidor) tendrán un coste pequeño.

El mantenimiento del servidor se hará mediante una herramienta automatizada que nos informará del estado del mismo, o de forma manual. Para la parte software nos encargaremos de que se realicen todas las actualizaciones necesarias, de los análisis periódicos del antivirus y de la buena configuración del cortafuegos. Únicamente se dejarán activos los programas y servicios necesarios. Aseguraremos también la limpieza y el aislamiento del servidor.

Por otro lado, en cuanto al mantenimiento de la base de datos usaremos una herramienta que proporcione las funciones necesarias (ej: SQL Server Management Studio) para el funcionamiento óptimo de la misma. Para ellos, ejecutaremos las operaciones de mantenimiento (comprobación de la coherencia de la base de datos, actualización de estadísticas) periódicamente, y siempre nos aseguraremos de tener una copia de seguridad actualizada para poder restaurarla en caso de fallo inesperado.

En lo referente al producto, se tratará de hacer un software que permita sencillas modificaciones y ampliaciones para su posible inclusión en un futuro.

### 3.5.4. Fiabilidad

Se le garantiza al usuario un juego intuitivo, con una dinámica fácil de comprender y exento de errores excepto los causados por su conexión a internet. Para ello se contará con un completo plan de pruebas en cada una de las fases de desarrollo del producto.

## 4. Apéndices

### 4.1. Modelo RUP

El modelo RUP (Proceso Unificado de Desarrollo) es un proceso de desarrollo software que, junto con el Lenguaje Unificado de Modelado (UML) conforma la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

Dado que el RUP está dirigido por casos de uso, podemos resaltar la importancia de ciertas secciones de este documento como pueden ser los requisitos de funcionalidad. A partir de ellos se podrá extraer con gran facilidad la estructura de funcionalidades del sistema y asignar los casos de uso correspondientes a cada apartado.

Por otro lado, mencionar también que a través de la información de esta especificación presentada se podrá obtener un modelo de requisitos bastante sólido. Hacer uso del mismo, y de los casos de uso derivados, permitirá crear una estructura arquitectónica básica del sistema. Dicha base será ampliada y refinada mediante diversas iteraciones del producto hasta obtener la versión final.

Según el Proceso Unificado de Desarrollo, la vida de un sistema se define como una serie de ciclos, cada uno de los cuales consta de 4 fases (descritas a continuación). Dada la limitación de tiempo actual, se llevará a cabo tan sólo un ciclo de desarrollo durante el curso 2012-2013 y, una vez finalizado éste, se procederá a nuevas versiones y ciclos del producto.

El mencionado ciclo, al igual que el resto, constará de varias etapas bien delimitadas:

- Fase de inicio: en ella se especificará una visión general y suficiente del proyecto para garantizar su viabilidad y posterior fase de elaboración. Durante este periodo se discutirán a fondo los detalles de funcionalidad más básicos del producto. En general, se hará un brainstorming de ideas durante las reuniones iniciales del equipo para tomar las más apropiadas para este ciclo del sistema y se descartarán (momentáneamente) aquellas que no puedan llevarse a cabo por limitaciones de tiempo/recursos.
- Fase de elaboración: una vez finalizada la etapa de inicio, y con unas ideas del producto más o menos definidas, se pasará a esta nueva fase. En ella se comenzará a diseñar la arquitectura básica del sistema y sus requisitos en función de las decisiones tomadas con anterioridad. Asimismo, se repartirán los recursos y actividades disponibles para permitir el comienzo del desarrollo del producto.
- Fase de construcción: una vez el producto y arquitectura del mismo hayan sido definidos en su forma más básica, se procederá a su implementación. Para ello se llevarán a cabo diversas iteraciones basándose en las funcionalidades (casos de uso) que irán acumulando módulos sobre el sistema base hasta completarlo. Esta

## JUGADOR #12

Documentación técnica: ERS

Ingeniería del Software, 2013

fase finalizará una vez el producto esté listo para ser distribuido, en su versión inicial, a la comunidad de usuarios.

- Fase de transición: cuando la construcción del sistema haya sido finalizada, el software se pondrá en manos de la comunidad de usuarios o, en nuestro caso, se iniciará un nuevo ciclo para obtener una versión mejorada y más estable del producto. Dicho ciclo comenzará al margen del presente curso de Ingeniería del Software.

# JUGADOR #12

Documentación técnica: gestión de riesgos

Ingeniería del Software, 2013

## Introducción

El propósito de este tipo de documento es identificar y prevenir riesgos inherentes al desarrollo de un proyecto de software, así como proponer formas de minimizar los daños que puedan causar. Cualquier riesgo detectado durante dicho desarrollo debería incluirse en este documento para su gestión.

## Índice

1. Lista de riesgos
  - 1.1 Tecnología inadecuada
  - 1.2 Cambio en la dinámica docente
  - 1.3 Pérdida de datos esenciales
  - 1.4 Pérdida de un miembro esencial
  - 1.5 Fallo de planificación
  - 1.6 Magnitud inabarcable
  - 1.7 Retrasos por causas académicas
  - 1.8 Visiones conflictivas
  - 1.9 Curva de aprendizaje / Formación insuficiente
  - 1.10 Falta de tolerancia a los cambios
  - 1.11 Excesivos recortes en el proyecto original
  - 1.12 Fallo grave de arquitectura

# JUGADOR #12

Documentación técnica: gestión de riesgos

Ingeniería del Software, 2013

## 1. Lista de riesgos

### 1.1 Tecnología inadecuada

**Probabilidad:** Media-baja (10-15%)

**Causa:** Mala elección de los lenguajes, el framework, o del gestor de la base de datos.

**Impacto:** Desastroso, obligaría a volver a la fase de inicio.

**Plan de prevención:** Estudio previo de proyectos similares y testing de las tecnologías.

**Plan de contingencia:** Cambio de tecnología reciclando cuanto sea posible del desarrollo fracasado.

### 1.2 Cambio en la dinámica docente

**Probabilidad:** Media-alta (15-30%)

**Causa:** El profesor del segundo cuatrimestre hace un seguimiento del proyecto distinto y cambia el procedimiento de las entregas.

**Impacto:** Medio, obliga a replanificar el proyecto y el método de trabajo.

**Plan de prevención:** Hablar con el profesor en cuestión para adelantar la planificación.

**Plan de contingencia:** Defender la metodología y adaptar en lo posible el trabajo sin rehacerlo.

### 1.3 Pérdida de datos esenciales

**Probabilidad:** Baja (2-5%)

**Causa:** Borrado accidental de documentos o archivos, problemas de acceso al drive o al repositorio.

**Impacto:** Bajo (pérdida parcial de documentación) ~ Desastroso (pérdida importante de código).

**Plan de prevención:** Replicar los datos, copias locales, tener cuidado al borrar o modificar.

**Plan de contingencia:** Intentar recuperar los datos o reconstruir los archivos perdidos.

### 1.4 Pérdida de un miembro esencial

**Probabilidad:** Muy Baja (0-3%)

**Causa:** Abandono o baja prolongada de un miembro con conocimientos únicos.

**Impacto:** Variable.

**Plan de prevención:** Escribir manuales para cualquier campo especializado que requiera el proyecto.

**Plan de contingencia:** Otro miembro del grupo tendrá que formarse en la materia por su cuenta.

### 1.5 Fallo de planificación

**Probabilidad:** Alta (30-50%)

**Causa:** Interferencias del cliente, retrasos internos, mala gestión de cualquier otro riesgo.

**Impacto:** Alto, puede causar que el proyecto no termine.

**Plan de prevención:** Aplicar sentido común al calcular los plazos y minimizar los cambios no planificados.

**Plan de contingencia:** Paralelizar las tareas atrasadas con el plan original, eliminar partes prescindibles para paliar retrasos.

## JUGADOR #12

Documentación técnica: gestión de riesgos

Ingeniería del Software, 2013

### 1.6 Magnitud inabarcable

**Probabilidad:** Media-baja (10-15%)

**Causa:** Fallo al medir la extensión del proyecto o al calcular los recursos necesarios.

**Impacto:** Alto, el proyecto no terminará como estaba planeado.

**Plan de prevención:** No subestimar el trabajo requerido, dejar márgenes razonables para contingencias.

**Plan de contingencia:** Recortar el proyecto cuanto sea necesario para terminar a tiempo.

### 1.7 Retrasos por causas académicas

**Probabilidad:** Alta (30-50%)

**Causas:** Paralización del proyecto durante fechas de exámenes/entregas de otras asignaturas.

**Impacto:** Bajo, muchas tareas son paralelizables y no tienen dependencias temporales.

**Plan de prevención:** Planificar una carga de trabajo muy baja durante épocas conflictivas.

**Plan de contingencia:** Repartir el trabajo entre los miembros del grupo que no coincidan en las mismas asignaturas.

### 1.8 Visiones conflictivas

**Probabilidad:** Baja (2-5%)

**Causa:** Distintas visiones del proyecto causan incompatibilidad de los módulos.

**Impacto:** Medio, hay que reconstruir los módulos conflictivos.

**Plan de prevención:** Discutir cada detalle **antes** de empezar a implementarlo, documentar para unificar versiones y ante la duda preguntar al líder.

**Plan de contingencia:** Rehacer las partes que no coincidan con el plan original.

### 1.9 Curva de aprendizaje / Formación insuficiente

**Probabilidad:** Media-alta (15-30%)

**Causa:** Los miembros no conocen las herramientas que tienen que usar.

**Impacto:** Asumible, somos mayorcitos para aprender a utilizarlas.

**Plan de prevención:** Incluir en la planificación periodos de formación.

**Plan de contingencia:** Preguntar al experto del grupo, consultar manuales, buscar en google...

### 1.10 Falta de tolerancia a los cambios

**Probabilidad:** Alta (30-50%)

**Causa:** Modelo de proceso unificado, excesivas interferencias del cliente (profesor).

**Impacto:** Alto, ya ha provocado retrasos importantes respecto la planificación.

**Plan de prevención:** La próxima vez elegir una filosofía más ágil.

**Plan de contingencia:** Simplificar el proyecto para compensar los retrasos.

### 1.11 Excesivos recortes en el proyecto original

**Probabilidad:** Media-alta (15-30%)

**Causa:** Directivas del cliente o retrasos excesivos.

**Impacto:** Asumible, el producto pierde funcionalidades y atractivo para los usuarios, pero se puede corregir una vez aprobada la asignatura.

## JUGADOR #12

Documentación técnica: gestión de riesgos

Ingeniería del Software, 2013

**Plan de prevención:** Defender el modelo ante el profesor cuando sea razonable hacerlo, evitar a toda costa los retrasos generales, planificar una arquitectura flexible.

**Plan de contingencia:** Añadir funcionalidades a posteriori.

### 1.12 Fallo grave de arquitectura

**Probabilidad:** Baja (2-5%)

**Causa:** Inexperiencia en la previsión de las necesidades del producto.

**Impacto:** Desastroso, requiere rehacer la arquitectura y repetir todas las fases desde este punto.

**Plan de prevención:** Proyectar una arquitectura flexible y bien planificada.

**Plan de contingencia:** Rehacer a toda prisa el proyecto reciclando código y confiando en terminar a tiempo.

## Introducción

Este documento presenta la previsión de las líneas de código del proyecto Jugador número 12. Se ha hecho analizando en la medida de lo posible proyectos existentes de envergadura similar, pero sobre todo tomando como punto de partida las líneas de código existentes a comienzos del mes de diciembre del 2012.

En ese momento, la funcionalidad era mínima y se ha extrapolado una aproximación para implementar la funcionalidad buscada. Se intenta una aproximación para cada lenguaje de programación que se va a usar en el proyecto

## Índice

1. Análisis de la previsión de las líneas de código.
  - 1.1 CSS
    - 1.1.1. Menús
    - 1.1.2. Divisiones
    - 1.1.3. Estilos
    - 1.1.4. CSS del framework
    - 1.1.5. Clase principal
  - 1.2. Javascript
    - 1.2.1 Validación de formularios
    - 1.2.2. Uso de Ajax
    - 1.2.3. Llamadas a funciones del framework
  - 1.3. MySQL
    - 1.3.1. Estructura básica de las tablas
    - 1.3.2. Relleno inicial de datos
  - 1.4. HTML
    - 1.4.1 Layouts
    - 1.4.2. Vistas
  - 1.5. PHP
    - 1.5.1. Controladores
    - 1.5.2. Modelos
    - 1.5.3. Componentes adicionales
    - 1.5.4. Añadido extra
2. Conclusiones
3. Anexos
  - Previsión de las líneas de código del proyecto

## 1. Análisis de la previsión de las líneas de código.

En primer lugar debemos resaltar que, dado el uso de un framework de PHP (Yii), la cantidad de líneas de código puede verse considerablemente reducida en comparación con otras aplicaciones reales. Hay que mencionar también que trataremos de redondear a lo alto las líneas de código calculadas ya que se supondrá que inicialmente no habrá optimizaciones en el código y puede haber fragmentos del sistema poco refinados o con redundancias.

### 1.1 CSS

Analizaremos en primer lugar una de las partes más sencillas, que se corresponde a los documentos de hojas de estilo (CSS). Dado que son comunes a toda la aplicación (en concreto a las vistas) trataremos de hacer una estimación basándonos en divisiones de funcionalidades:

#### 1.1.1. Menús

Partiendo de que la aplicación contiene actualmente un sencillo menú con una funcionalidad más que suficiente para la misma, nos basaremos en este diseño para realizar una estimación. Actualmente esta sección consta de 134 líneas de código CSS. Teniendo en cuenta que tiene actualmente incluida funcionalidad para submenús (de los cuales no haremos uso), podemos establecer una cota inferior en unas 100 LDC. Suponiendo que podamos añadir aún más funcionalidad/decoración que la actual, pasariamos a una cota superior de 180 LDC.

Esto nos ofrece una media de LDC para los menús de **140**.

#### 1.1.2. Divisiones

Esta sección tratará de las CSS asociadas a la distribución de elementos. Dado que en el mockup se cuenta actualmente con unas 80 LDC para la distribución más básica, daremos en este apartado un margen mucho más amplio. Podemos suponer que se rondarán las 200 LDC para maquetación básica y le añadiremos un extra de 50 líneas para refinado.

Esto suma un total esperado de **250** líneas de código CSS.

#### 1.1.3. Estilos

Apartado que corresponde a la definición de estilos (color, fuente, etc.) para las distribuciones y elementos anteriores. Actualmente consta de tan sólo 20 líneas de código, con lo que ampliaremos el margen hasta **150** líneas ya que pueden ser más que suficientes para nuestros objetivos.

#### 1.1.4. CSS del framework

Actualmente, y posiblemente sea lo único necesario, se hace uso de los elementos del widget de formularios de Yii.

Estos ocupan un total aproximado de **130** LDC.

#### 1.1.5. Clase principal

Clase principal: se encargará de reunir todos los elementos CSS a través de sentencias “Import”. Su número de líneas de código estará situado cerca de 10.

## 1.2. Javascript

Usado sobre todo para validaciones o funciones sencillas a ejecutar en el lado del cliente. Incluiremos en esta sección las LDC de Ajax. Puesto que haremos uso del framework de Yii, la mayor parte de este apartado vendrá dado por funciones internas ya creadas, por lo que el número de LDC esperado será mucho menor al de una aplicación real creada sin ayuda de librerías.

### 1.2.1 Validación de formularios

Conformará una gran parte de este tipo de código. Supondremos una cantidad media de 5 LDC por campo validado y una media de 20 campos o secciones a validar. El número de validaciones es reducido ya que se ha tratado de eliminar las realizadas por el framework Yii.

Previsión de **100** LDC.

### 1.2.2. Uso de Ajax

Uso de Ajax: en situaciones como la realización de acciones durante un partido. Dado que no viene implementado por defecto en Yii, este apartado aumentará bastante la cantidad de LDC de Javascript. Supondremos una media de 10 subprogramas para llevar a cabo toda la funcionalidad necesaria asincrónicamente y unas 50 líneas de código por cada uno.

Esto suma un total de **500** LDC.

### 1.2.3. Llamadas a funciones del framework

Llamadas a funciones del framework: será una mínima parte del total, pero añadiremos un extra de **40** LDC para esta sección.

## 1.3. MySQL

Crearemos una estimación basándonos en dos parámetros principales:

### 1.3.1. Estructura básica de las tablas

Partiendo de que el prototipo se ha creado a partir de un archivo de unas 120 LDC.

Suponiendo que dicha estructura se corresponda a un 60% de la total, la composición final supondría un archivo de unas **200** líneas de código SQL.

### 1.3.2. Relleno inicial de datos

Añadiremos un extra de **500** LDC ya que será necesario inicializar determinados elementos como los equipos o habilidades.

## 1.4. HTML

Conformará el grueso de la parte de presentación, por lo que dividiremos esta sección en los siguientes apartados:

### 1.4.1 Layouts

Formados en su mayor parte por código HTML. Su estimación se basará en los existentes actualmente en el mockup.

Éstos contienen alrededor de **120** LDC.

# JUGADOR #12

Documentación técnica: previsión de LDC

Ingeniería del Software, 2013

## 1.4.2. Vistas

Supondremos una media de 70 líneas de código para cada una, y el uso de un total aproximado de 50 vistas.

Esto nos da un total de **3500** LDC.

## 1.5. PHP

Esta sección conformará el grueso de la aplicación. Al igual que las anteriores la dividiremos en diferentes apartados:

### 1.5.1. Controladores

De nuevo, realizando suposiciones, partiremos de los siguientes parámetros: 8 controladores de media, 5 acciones por cada uno y 2 funciones auxiliares creadas. Una posible estimación en líneas básicas de código por controlador es de 70, 50 por acción y 20 por función auxiliar creada. Teniendo todo esto en cuenta.

Obtenemos un total aproximado de **3000** LDC.

### 1.5.2. Modelos

Actualmente contamos con 13 modelos para la estructura básica. Teniendo en cuenta que aún faltan modelos para el sistema de mensajes privados y posiblemente la organización de roles de usuario, tomaremos una media de 16 modelos con 150 LDC para cada uno.

Esto nos da una medida aproximada de **2400** LDC.

### 1.5.3. Componentes adicionales

Aquí incluiremos elementos adicionales como clases auxiliares, componentes y scripts.

1. Clases auxiliares: el grueso de las mismas estará compuesto por los Singleton usados para la funcionalidad de las acciones del juego. Suponiendo una media de 50 clases (con una acción asociada) y 70 líneas de código para cada una obtenemos un total de **3500** líneas de código. Se ha engrosado un poco este resultado para tener en cuenta posibles clases más pequeñas como Validadores personalizados.
2. Componentes: conformarán una muy pequeña parte pero se mencionarán para mayor precisión. Actualmente se hace uso tan sólo del archivo UserIdentity.php, que consta de 30 LDC. Suponiendo la creación de algún componente extra, aumentaremos esta cifra hasta las **100** LDC.
3. Scripts: de nuevo, conformarán gran parte de la funcionalidad del sistema. Vamos a suponer el uso de un total de 3 scripts pequeños (unas 200 LDC) y 1 más grande para simular partidos (unas 400 LDC). Esto suma un total de **1000** líneas de código para este apartado.

### 1.5.4. Añadido extra

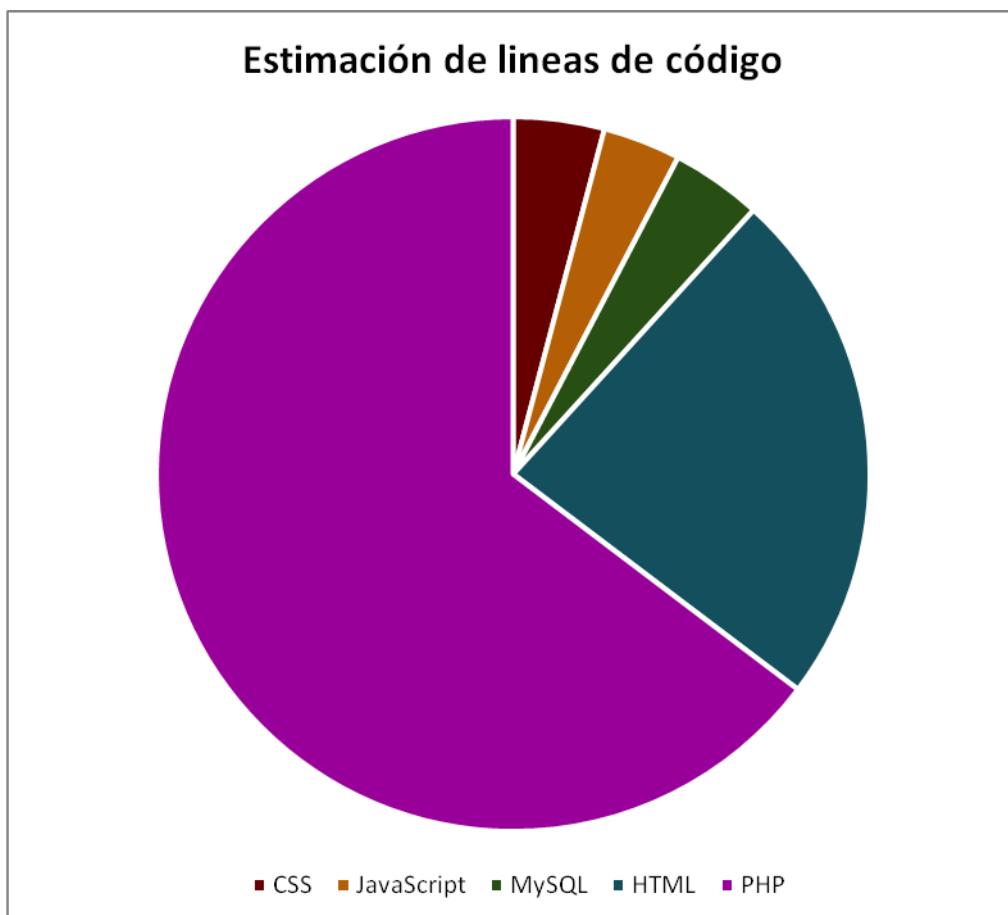
Sumamos una pequeña cantidad de 1300 líneas de código para cubrir posibles ampliaciones o funciones auxiliares inesperadas así como algunos huecos que pudiesen haber quedado en secciones anteriores.

## 2. Conclusiones

Tras todo este estudio, los cálculos finales para la estimación de líneas de código de la aplicación se encuentran en la hoja de cálculo del anexo1. Los redondearemos ligeramente con respecto a los obtenidos con anterioridad.

## 3. Anexos

1. Manuel Artero, Hoja de cálculo con el reparto de las líneas de código estimadas :  
<https://docs.google.com/spreadsheets/ccc?key=0Ao09kH8yz2wzdDUxcGdDRIVoWGtrUnp2NUFkOEhSWHc#gid=0>



# JUGADOR #12

Documentación técnica: otras métricas (puntos de función) Ingeniería del Software, 2013

## Introducción

Este documento presenta y define la métrica escogida para medir el desarrollo del proyecto. Se analizan las posibles métricas explicadas en clase explicando por qué se ha rechazado o se ha aceptado cada una.

## Índice

1. Decisión de la métrica utilizada
  - 1.1. Descarte de otras métricas
  - 1.2. Decisión de los puntos de función como métrica utilizada
2. Definición de los puntos de función utilizados
  - 2.1. Módulos contemplados y división de los puntos de función
  - 2.2. Definición de la progresión en las tareas
3. Anexos

## 1. Decisión de la métrica utilizada

Se ha definido un sistema de medición basado en puntos de función como métrica al desarrollo del proyecto. El baremo utilizado; así como las razones por las que se han rechazado otras métricas se explican a lo largo del documento.

### 1.1. Descarte de otras métricas

Se ha rechazado utilizar el **número de líneas de código** por los siguientes motivos:

1. Considerar que no representan una medida fiable sobre el estado el proyecto: posibilidad de avanzar en el desarrollo del mismo reduciendo el número de líneas; por lo tanto no sirve como métrica del desarrollo
2. Métrica fuertemente dependiente de factores ajenos como pueden ser el lenguaje utilizado (incluso de la versión del lenguaje) o el estilo de programación de los desarrolladores.
3. Posibilidad de crear una tendencia de escribir líneas “inútiles” o “superfluas” con el fin de engordar los resultados de cara a una revisión del proyecto.

Sin embargo, para poder comparar el proyecto con el resto de grupos de la asignatura, también se han definido las líneas de código previstas. Para cada cifra se ha recurrido a la comparación con proyectos similares para intentar acertar lo más posible en nuestra previsión (anexo1).

Igualmente, se ha rechazado la **actividad en servidor remoto** como métrica para el proyecto (frecuencia y número de *commits*) por lo siguientes motivos, similares a los anteriores:

1. No representa una medida fiable sobre el estado del proyecto: igual que para el número de líneas de código, un mayor número de commits no implica un avance en el desarrollo del proyecto al existir la posibilidad de tratarse de modificaciones a código ya existente.
2. Métrica dependiente del estilo de programación de los desarrolladores. No existe una definición para “cuándo deben hacerse los *commits*” y cada desarrollador es libre de marcarlos en el momento que se crea oportuno
3. Posibilidad de crear una tendencia a hacer *commits* “vanos” para intentar crear la ilusión de un mayor avance en el proyecto.

### 1.2. Decisión de los puntos de función como métrica utilizada

La métrica escogida son los **puntos de función** por los siguiente motivos:

1. Mide directamente la funcionalidad entregada al usuario; por lo tanto, un mayor número de puntos de función completados sí implicará un avance en las funcionalidades disponibles.
2. Son independientes de la tecnología utilizada para la construcción.
3. Capacidad de medir un avance en el desarrollo en cualquiera de las fases de vida del software (desde inicio hasta transición) de forma que podemos hablar de “puntos de función completados” antes de la implementación del sistema.
4. Métrica adaptable al proyecto; es decir, los puntos de función se pueden definir para las funcionalidades propias del proyecto.

# JUGADOR #12

Documentación técnica: otras métricas (puntos de función) Ingeniería del Software, 2013

## 2. Definición de los puntos de función utilizados

El proyecto consta de **350 puntos de función** que representan funcionalidad del sistema agrupada en varios módulos (aproximadamente 7 puntos de función / persona en un mes). Cada módulo lleva asociado un valor que representa el **peso del módulo** sobre el proyecto completo. A mayor peso, mayor funcionalidad asociada a ese módulo.

Sobre este valor base, se aplica un modificador que representa la **dificultad prevista** para realizar el módulo. Este modificador varía entre uno y cinco; siendo uno “sin dificultad extra” y cinco “módulo con una gran dificultad prevista”.

Los puntos de función totales para un módulo se calculan multiplicando el peso base por el modificador de dificultad. Siendo M los módulos totales:

$$\text{Puntos de función del proyecto} = \sum_{i=1}^M (\text{PF}_i) = \sum_{i=1}^M (\text{peso}_i * \text{dificultad}_i)$$

### 2.1. Módulos contemplados y división de los puntos de función

Se ha dividido el proyecto en 20 módulos que agrupan funcionalidad del sistema. Qué agrupa cada módulo, así como cuántos puntos de función totales (incluyendo peso y dificultad) ocupa cada módulo se detalla a continuación.

El desglose completo de los puntos de función y gráficos de la división del proyecto se encuentra en el anexo2

1. **Definición del juego:** engloba la parte de creación y definición de las reglas del juego que se va a implementar.  
Puntos de función totales: **42**
2. **Diagramas y casos de uso:** mide el proceso de definición, discusión y pasos necesarios en el modelo de desarrollo escogido (guiado por casos de uso).  
Puntos de función totales: **16**.
3. **Login y registro de usuarios:** engloba la implementación, validación y pruebas de seguridad para la parte de registro y manejo de usuarios en el sistema.  
Puntos de función totales: **6**.
4. **Selección de personaje y afición:** implementa la “creación” de un nuevo personaje  
Puntos de función totales: **1**.
5. **Desbloqueo de habilidades:** engloba la implementación y la interacción del usuario con el árbol de habilidades.  
Puntos de función totales: **2**.
6. **Gestión de recursos:** implementación de la gestión completa de recursos de un jugador (transacciones y ganancia automática de recursos).  
Puntos de función totales: **18**.
7. **Uso de habilidades de jugador:** implementación de todo lo relacionado con las habilidades de jugador (habilidades individuales).  
Puntos de función totales: **14**.

## JUGADOR #12

Documentación técnica: otras métricas (puntos de función) Ingeniería del Software, 2013

8. **Participar en acciones grupales:** interacción de los jugadores con acciones grupales ya abiertas (donación de recursos en las acciones)  
Puntos de función totales: **3.**
9. **Registro de acciones completadas:** Representa la implementación del efecto al completar las acciones grupales. Por ejemplo, aplicar el beneficio correspondiente a todos los participantes de una acción grupal.  
Puntos de función totales: **36.**
10. **Funcionamiento de la liga (encuentros):** Engloba la gestión de los equipos y el calendario; es decir la creación de partidos y el registro de partidos jugados.  
Puntos de función totales: **27.**
11. **Asistir a los partidos (jugados o no):** Implementación de la funcionalidad de “asistir a un partido”  
Puntos de función totales: **3.**
12. **Partido: jugarse:** Implementación del algoritmo de un partido.  
Puntos de función totales: **40.**
13. **Partido: participar: Implementación de**  
Puntos de función totales: **60.**
14. **Ganancia de experiencia:** implementación de la evolución del personaje.  
Puntos de función totales: **3.**
15. **Mensajería: Entendido como “añadido” al proyecto al estar relacionado**  
Puntos de función totales: **1.**
16. **Chat**  
Puntos de función totales: **15.**
17. **Gestión de objetos:** definición e implementación de objetos que puedan comprar los jugadores para mejorar el personaje (añadido a la arquitectura)  
Puntos de función totales: **2.**
18. **Arte:** engloba todo lo relacionado con la parte artística propia de a) un juego; y b) diseño UX (*User eXperience*): Diseño de la página web con vistas a una posible comercialización, incluye la elección de los colores, disposición de elementos y menús para el usuario.  
Puntos de función totales: **27.**
19. **Maquetación de la página web:** “Implementación” del diseño de las vistas de la página web  
Puntos de función totales: **14.**
20. **Tareas estructurales de apoyo al proyecto:** Módulo que representa la organización del grupo de programación y otras tareas estructurales.  
Puntos de función totales: **20.**

### 2.2 Definición de la progresión en las tareas

Además de los puntos de función, se ha definido una escala para medir la progresión en una determinada tarea.

Se tendrán en cuenta cuatro factores:

1. **Implementación** de la funcionalidad. Hasta un 40% del total del trabajo.

## JUGADOR #12

Documentación técnica: otras métricas (puntos de función) Ingeniería del Software, 2013

2. **Seguridad:** la implementación es segura; esto es, si hay acceso a una base de datos asegura la atomicidad por medio de transactions, se controla la inyección sql por parte del usuario. Hasta un 20%
3. **Testing:** la implementación supera una serie de test automatizados. Hasta un 20%
4. **Integración:** la funcionalidad se integra con el resto de funcionalidades y se pasan nuevos test de funcionalidad conjunta. Hasta un 20%

Por ejemplo; en un punto del desarrollo la implementación del partido está prácticamente completa (75%) y asegurada la seguridad; pero no se han diseñado aún tests ni se ha pasado a la parte de la integración.

En esas condiciones la funcionalidad del partido se ha completado en un 50%. Si los puntos de función totales previstos son 40, se habrían completado 20 puntos de función.

### 3. Anexos

1. Manuel Artero, Documento con la previsión de las líneas de código del proyecto :  
<https://docs.google.com/document/d/1zllt-PmU0VfOqBMmNYBOliZVvQXJMViPBM4NFi5XU/edit#heading=h.a8e2sbpde1ha>
2. Manuel Artero, Documento con el desglose de los puntos de función y la división del proyecto. :  
<https://docs.google.com/spreadsheets/ccc?key=0Ao09kH8yz2wzdEtUZWhzWi02bWtjLUt5bjJXVXVzOEE#gid=0>

# JUGADOR #12

Documentación técnica: otras métricas (puntos de función) Ingeniería del Software, 2013

## 1. Definición de los puntos de función y pesos asociados

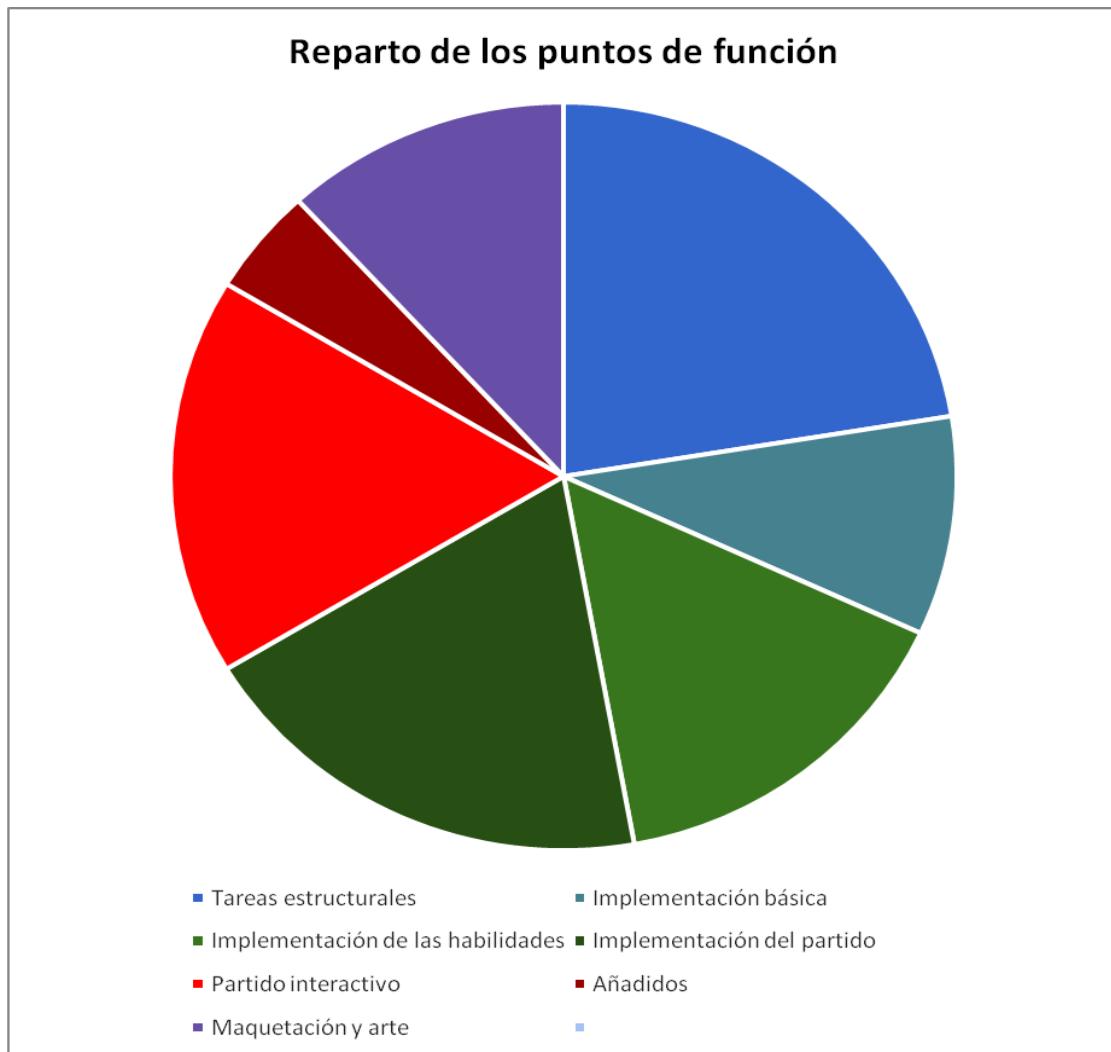
Módulo de funcionalidad	Puntos de función	Modificador de dificultad	Estado actual	Puntos de función actuales	Valor si completado
Definicion del juego	14	3	0,9	37,8	42
Casos de uso	8	2	0,5	8	16
Login y registro de usuarios	3	2	0,9	5,4	6
Seleccion de personaje y aficion	1	1	0,9	0,9	1
Desbloqueo de habilidades	2	1	0,6	1,2	2
Gestion de recursos	6	3	0,8	14,4	18
Uso de habilidades de jugador	7	2	0,7	9,8	14
Participar en acciones grupales	3	1	0,9	2,7	3
Registro de acciones completadas	9	4	0,8	28,8	36
Funcionamiento de la liga (encuentros)	9	3	0,2	5,4	27
Asistir a los partidos (jugados o no)	3	1	0,8	2,4	3
Partido: jugarse	10	4	0,8	32	40
Partido: participar	12	5	0,2	12	60
Ganancia de experiencia	3	1	0	0	3
mensajeria	1	1	0,2	0,2	1
chat	5	3	0	0	15
Gestion de objetos	0	0	0	0	0
Arte	9	3	0,4	10,8	27
Maquetacion	7	2	0,7	9,8	14
Tareas estructurales de apoyo al proyecto	10	2	0,7	14	20
Total				195,6	348

## JUGADOR #12

Documentación técnica: otras métricas (puntos de función)

Ingeniería del Software, 2013

Reparto de los puntos de función en el proyecto	Puntos de función totales
Tareas estructurales	78
Implementación básica	33
Implementación de las habilidades	53
Implementación del partido	67
Partido interactivo	60
Añadidos	16
Maquetación y arte	41



## JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

### Casos de uso: Índice

1. Acceder al árbol de habilidades
2. Adquirir habilidades
3. Asistir a un partido
4. Autenticación de usuarios
5. Borrar mensaje privado
6. Borrar notificaciones
7. Cambiar afición
8. Consultar acciones de afición
9. Consultar aficiones
10. Consultar datos del personaje
11. Consultar datos de un partido
12. Consultar y modificar datos de usuario
13. Ejecutar acciones de partido
14. Ejecutar acciones grupales
15. Ejecutar acciones individuales
16. Enviar mensaje privado
17. Expulsar jugador
18. Participar en acciones grupales
19. Registrar usuario
20. Ver mensaje privado

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

## Caso de uso: Acceder al árbol de habilidades

### 1. Objetivo

Generar lista de habilidades tanto bloqueadas como desbloqueadas ofreciendo la posibilidad de seleccionar cualquiera de ellas para mostrar información avanzada.

### 2. Actores

Usuario registrado.

### 3. Precondición

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar previamente registrado en el sistema y correctamente autenticado en el mismo (haber hecho login).

### 4. Postcondición si éxito

Se visualizará la lista completa de habilidades con sus características básicas (descripción, efectos positivos, efectos negativos, lista de requisitos) diferenciando entre aquellas que el usuario tiene bloqueadas y las desbloqueadas. Asimismo, por cada una de ellas, se ofrece una lista de las acciones que desbloquea dicha habilidad..

### 5. Postcondición si fallo

Se mostrará un mensaje informativo al usuario y se continuará la ejecución del sistema.

### 6. Entradas

Habilidades que se desean visualizar (opcional).

### 7. Salidas

Se mostrará la lista general de habilidades con información básica así como información más avanzada de cada una al ser seleccionada. En caso de error, se mostrará un mensaje informativo.

### 8. Secuencia normal

1. El usuario accede al menú relacionado con el árbol de habilidades.
2. Se recopila la información necesaria de la base de datos para mostrar la lista de habilidades completa. Haciendo uso de dicha información, asimismo, se diferenciará entre las habilidades que ya han sido desbloqueadas y aquellas que aún permanecen bloqueadas. También se mostrará información básica de cada habilidad sin necesidad de seleccionarla. Si hay un error, pasar a S-1.
3. Se selecciona una habilidad para mostrar información avanzada.
4. Nuevo acceso a la base de datos. De ahí se recopila la descripción de la habilidad, efectos, acciones que desbloquea y costes de recursos de las mismas. Si hay un error, pasar a S-2.
5. Se muestra una nueva vista con los datos recopilados. En ella el usuario puede consultar todos los datos relacionados con dicha habilidad así como desbloquearla en caso de que no la tenga ya (véase caso de uso asociado).

### 9. Secuencia alternativa

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

- S-1. Error al obtener la información de las habilidades. Se muestra un mensaje informativo al usuario y se continúa la ejecución del sistema.
- S-2. Error al obtener la información de la habilidad concreta. Se muestra un mensaje informativo al usuario y se continúa la ejecución del sistema.

## Caso de uso: Adquirir habilidades

### 1. Objetivo.

Permitir al usuario la adquisición de nuevas habilidades y desbloquear nuevas mejoras para el personaje.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar registrado en el sistema y correctamente autenticado.

El usuario debe tener puntos de habilidad disponibles para gastar.

Cumplir los requisitos para adquirir la habilidad. Por ejemplo, disponer del nivel necesario o de aquellas habilidades necesarias para desbloquear la actual.

### 4. Postcondición si éxito.

La nueva habilidad se desbloqueará para el personaje aplicándole los beneficios asociados (bonificaciones, nuevas acciones, etc.).

### 5. Postcondición si fallo.

Se mostrará un mensaje informativo al usuario y se continuará la ejecución del sistema.

### 6. Entradas.

Habilidades que se desean desbloquear.

### 7. Salidas.

Se mostrará un mensaje de éxito en caso de que la operación se lleve a cabo correctamente. En caso de error, se mostrará un mensaje informativo del mismo.

### 8. Secuencia normal.

1. El usuario accede al árbol de habilidades (ver caso de uso asociado). Si el usuario selecciona una de ellas para ver información avanzada, pasar a S-1.
2. El usuario elige la habilidad deseada y selecciona la opción de desbloquearla.
3. Se accede a la base de datos y se modifican para agregar la nueva habilidad a la lista de desbloqueadas del usuario y darle los beneficios asociados al personaje. Si error, pasar a S-2.

### 9. Secuencia alternativa.

- S-1. En la ventana de información avanzada el usuario tiene la opción de desbloquear dicha habilidad al igual que desde el árbol completo. Una vez seleccionada, pasar a secuencia normal, punto 2.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

S-2. Se muestra un error informativo al usuario y se continúa la ejecución normal del sistema.

## Caso de uso: asistir a un partido.

### 1. Objetivo.

Permitir la asistencia de un usuario a uno de los partidos en juego.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar previamente registrado en el sistema correctamente autenticado.

El encuentro a visualizar debe estar en juego.

### 4. Postcondición si éxito.

Se mostrará una vista al usuario con toda la información de dicho encuentro.

### 5. Postcondición si fallo.

Se mostrará un mensaje de error al usuario.

### 6. Entradas.

El identificador del encuentro a visualizar.

### 7. Salidas.

Mensaje de error en caso de fallo.

### 8. Secuencia normal.

1. El usuario accede a la sección del calendario de partidos. Opcionalmente puede llegar al próximo partido de su equipo a través de la sección de información general.

2. El jugador elige el partido al que desea asistir.

3. Se comprueba en la base de datos que dicho partido esté en juego. En caso contrario, pasar a S-1.

4. Se toman los datos necesarios referentes al encuentro de la base de datos (marcador, estadísticas, turno, etc.). En caso de que el encuentro a visualizar sea de la afición del usuario se recopilará también la lista de acciones de partido desbloqueadas por el mismo (véase caso de uso “ejecutar acciones de partido”). En caso de error, pasar a S-2.

5. Se muestran los datos del partido en una nueva vista. Ésta deberá tener la capacidad de actualizarse automáticamente sin necesidad de recargar el navegador para mejorar la experiencia de juego del usuario.

### 9. Secuencia alternativa.

S-1. Error, el partido no está en juego. Se informa de ello al usuario y se redirige a la sección del calendario de encuentros.

S-2. Error al recopilar información del encuentro. Se informa al jugador y se redirige a la sección del calendario de partidos.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

## Caso de uso: autenticación de usuarios (login)

### 1. Objetivo.

Identificar a un usuario y permitir su entrada al sistema.

### 2. Actores.

Usuario invitado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar previamente registrado en el sistema y no haber sido autenticado.

### 4. Postcondición si éxito.

El usuario será identificado, creándose una sesión propia y permitiendo el acceso al resto de funcionalidades del sistema.

### 5. Postcondición si fallo.

Se mostrará un mensaje de error al usuario.

### 6. Entradas.

Nombre de usuario y contraseña a autenticar.

### 7. Salidas.

Mensaje de error en caso de fallo.

### 8. Secuencia normal.

1. El usuario accede a la página principal de la aplicación. En ella se encuentra un formulario de inicio de sesión que solicita el nombre de usuario y contraseña del jugador.

2. El usuario rellena dicho formulario con sus datos personales y selecciona la opción de iniciar sesión en el sistema.

3. Se toman los datos proporcionados y se verifican con los de la base de datos. En caso de error, pasar a S-1.

4. Con los datos validados, se crea una nueva sesión en el servidor y se prepara todo lo necesario para la entrada del nuevo usuario. En caso de error, pasar a S-2.

5. Se redirige al usuario a la sección principal de la aplicación, que puede ser la página de perfil u otra vista general.

### 9. Secuencia alternativa.

S-1. Error al verificar los datos del formulario. Se muestra un mensaje de error indicando cuál de los campos es incorrecto y se muestra de nuevo el formulario de inicio de sesión.

S-2. Error al crear la nueva sesión e inicializar los datos. Se muestra un mensaje de error (opcional) y se redirige a la página principal con el usuario sin autenticar.

## Caso de uso: borrar mensaje privado

### 1. Objetivo.

Borrar un mensaje privado de la lista de enviados o recibidos.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

## 2. Actores.

Usuario registrado.

## 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar previamente registrado en el sistema y correctamente autenticado.

## 4. Postcondición si éxito.

Se eliminará el mensaje seleccionado de la bandeja del jugador.

## 5. Postcondición si fallo.

Se mostrará un mensaje de error al usuario.

## 6. Entradas.

Identificador del mensaje que se desea eliminar.

## 7. Salidas.

Mensaje de error en caso de fallo.

## 8. Secuencia normal.

1. El usuario accede a la sección de mensajería privada.

2. El usuario selecciona la opción de la bandeja de entrada o de salida para ver todos los mensajes en los que participa (como emisor o receptor).

3. El usuario selecciona uno para borrar.

4. Se accede a la base de datos para eliminar la referencia a dicho mensaje, en caso de que el otro jugador implicado ya lo haya borrado, se borra el mensaje completo de la base de datos. En caso de error, pasar a S-1.

5. Se recarga la bandeja de mensajes del jugador sin el mensaje en cuestión.

## 9. Secuencia alternativa.

S-1. Error al recopilar los datos del mensaje. Se muestra un mensaje de error y se vuelve a la bandeja sin modificar, para continuar la ejecución del sistema.

## Caso de uso: borrar notificaciones

### 1. Objetivo.

Borrar uno o varios elementos de la lista de notificaciones.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar previamente registrado en el sistema y correctamente autenticado.

### 4. Postcondición si éxito.

Se eliminará(n) la(s) notificación(es) seleccionada(s).

### 5. Postcondición si fallo.

## JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

Se mostrará un mensaje de error al usuario.

### 6. Entradas.

(Opcional) Identificador de la notificación que se desea eliminar.

### 7. Salidas.

Mensaje de error en caso de fallo.

### 8. Secuencia normal.

1. El usuario accede a la sección de notificaciones.

2. El usuario quiere eliminar sólo una notificación:

2.1. En este caso, bastará con seleccionar la notificación deseada y hacer click sobre el botón de “Borrar”. En caso de error, pasar a S-1.

2.2. Una vez eliminada, hay dos opciones. Si el usuario quiere eliminar alguna otra, puede volver al punto 1. Si desea continuar usando el sistema, pasar al punto 4.

3. El usuario desea eliminar todas las notificaciones:

3.1. Bastará con hacer click sobre el botón de “Borrar todas”. En caso de error, pasar a S-2.

3.2. Dado que en este punto ya no quedará nada a eliminar, pasar al apartado 4.

4. Mostrar la página de notificaciones sin aquellas eliminadas.

### 9. Secuencia alternativa.

S-1. Error al eliminar la notificación. Se muestra un mensaje de error y se vuelve a la sección de notificaciones para continuar la ejecución del sistema.

S-2. Error al eliminar todas las notificaciones. Se muestra un mensaje de error y se vuelve a la sección de notificaciones para continuar la ejecución del sistema.

## Caso de uso: Cambiar afición

### 1. Objetivo.

El cambio de afición por parte de un jugador.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar registrado en el sistema y correctamente autenticado.

La afición seleccionada para el cambio debe ser distinta a la actual.

### 4. Postcondición si éxito.

El usuario es traspasado a la nueva afición. Todas las acciones grupales abiertas por él serán cerradas y sus participaciones en las demás mantenidas.

### 5. Postcondición si fallo.

El jugador permanecerá en su afición y se mostrará un mensaje indicativo del error.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

## 6. Entradas.

Afición a la que se desea ser traspasado.

## 7. Salidas.

Mensaje de error en caso de fallo.

## 8. Secuencia normal.

1. El usuario selecciona la vista de información del equipo al que desea ser traspasado. Una vez en ella, elige la opción de cambiar de afición.
2. Se pregunta al jugador si realmente desea llevar a cabo el traspaso. En caso negativo, pasar a S-1.
3. Se procede a cerrar todas las acciones grupales abiertas por el usuario en la afición inicial. Como penalización, se dejarán abiertas las participaciones del usuario en las acciones de afición de otros jugadores. En caso de error, pasar a S-2.
4. Se realizan los cambios necesarios en la base de datos para llevar a cabo el traspaso de afición. Asimismo, se reasignan las variables de sesión necesarias para mantener la coherencia en la ejecución del sistema. En caso de error, pasar a S-3.
5. Se muestra la vista con la información de la nueva afición.

## 9. Secuencia alternativa.

S-1. El usuario no desea realizar el cambio por lo que permanece en su afición original.

S-2. Error al cerrar acciones grupales. Se muestra un mensaje con el error y el usuario permanece en su afición original. Se continúa la ejecución normal del sistema.

S-3. Error al realizar el cambio de afición. Se muestra un mensaje con dicho error y el usuario permanece en su afición original. Se continúa la ejecución normal del sistema.

## Caso de uso: consultar acciones de afición.

### 1. Objetivo.

Mostrar una lista con todas las acciones abiertas en las que puede participar el usuario, con los recursos necesarios y el estado actual.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar registrado y correctamente autenticado en el sistema.

El usuario sólo puede consultar las acciones de la afición a la que pertenece.

### 4. Postcondición si éxito.

Se mostrará la lista de acciones abiertas, junto con el nombre del creador y el número actual de participantes. Asimismo, de forma opcional, se mostrará la información avanzada de la acción en caso de ser necesario.

### 5. Postcondición si fallo.

Se mostrará un mensaje indicativo del error y se continuará la ejecución del sistema.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

## 6. Entradas.

La acción grupal de la que se desea obtener información avanzada (opcional).

## 7. Salidas.

Lista de acciones grupales abiertas en la afición e información detallada de las mismas (opcional). Mensaje indicativo en caso de error.

## 8. Secuencia normal.

1. El usuario selecciona el menú asociado a la afición.
2. Se accede a la base de datos y se muestra una vista con la información de dicha afición, entre la que se incluye la lista de acciones abiertas. En caso de error, pasar a S-1.
3. Opcionalmente, el usuario selecciona una de las acciones abiertas para pasar a una vista avanzada.
4. Se obtienen los datos completos de esa nueva acción de la base de datos. En caso de error, pasar a S-2.
5. Se muestra una nueva vista al usuario donde se visualizan estos datos avanzados. Entre ellos se incluye el número de participantes, recursos aportados y restantes, tiempo de finalización, etc.

## 9. Secuencia alternativa.

S-1. Error al consultar los datos de la afición. Se muestra un mensaje de error y se continúa la ejecución normal del sistema.

S-2. Error al consultar los datos de la acción grupal. Se muestra el mensaje de error asociado y se continúa la ejecución normal del sistema.

## Caso de uso: Consultar aficiones.

### 1. Objetivo.

Ver la información de la afición seleccionada: aforo de su estadio, número de hinchas, nivel del equipo, miembros, etc. Si la afición es la del propio usuario, se visualizarán además datos más avanzados.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario está registrado y correctamente autenticado en el sistema.

### 4. Postcondición si éxito.

Se muestra la información de la afición solicitada.

### 5. Postcondición si fallo.

Se muestra un mensaje de error indicando el fallo.

### 6. Entradas.

El identificador de la afición de la cual se desean obtener los datos.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

## 7. Salidas.

La información de la afición seleccionada o mensaje de error.

## 8. Secuencia normal.

1. El usuario accede a la sección del calendario, la clasificación u otro punto desde el que poder ver una afición concreta.
2. Acto seguido selecciona aquella que desea visualizar.
3. Se accede a la base de datos y se recogen los datos necesarios: aforo, miembros de la afición, nivel del equipo, etc. Si la afición es la del usuario, además, se recogerán los datos de las acciones grupales abiertas. En caso de error, pasar a S-1.
4. Mostrar al usuario una vista con los datos recopilados.

## 9. Secuencia alternativa.

S-1. Error al obtener la información. Mostrar mensaje de error y continuar la ejecución del sistema.

## Caso de uso: Consultar datos del personaje.

### 1. Objetivo.

Mostrar la información del perfil de un jugador.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario está registrado y correctamente autenticado en el sistema.

### 4. Postcondición si éxito.

Se mostrará la información del perfil de usuario seleccionado.

### 5. Postcondición si fallo.

Se mostrará un mensaje de error indicando el problema.

### 6. Entradas.

El usuario del que se desea ver el perfil.

### 7. Salidas.

Información del perfil del usuario.

### 8. Secuencia normal.

1. Si el usuario selecciona la opción de su perfil a través de los menús, continuar secuencia normal. En caso de seleccionar el perfil de otro usuario pasar a S-1.
2. Se recopila la información necesaria de la base de datos. Esto incluye nombre del jugador, tipo de personaje, recursos, acciones pasivas, etc. En caso de error, pasar a S-2.
3. Se muestra una nueva vista al usuario presentando los datos recogidos.

### 9. Secuencia alternativa.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

S-1. El usuario desea visualizar el perfil de otro. Se accede a la base de datos y se recoge la información básica de ese perfil: nombre y tipo de personaje. Continuar la secuencia normal, punto 3.

S-2. Error al obtener los datos. Se muestra un mensaje informativo del error y se continúa la ejecución del sistema.

## Caso de uso: Consultar datos de un partido

### 1. Objetivo.

Obtener la información relativa a un partido que puede haberse jugado o no.

En caso de asistir a un partido pasado, se puede consultar su resultado así como la crónica del mismo; si se asiste a un partido que aún no se ha jugado se obtiene la información actual relativa a los equipos implicados así como a la información de acciones completadas por las aficiones para ese partido y la fecha, el aforo previsto para el encuentro y el ambiente.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar registrado y correctamente autenticado en el sistema.

El partido seleccionado debe estar pendiente de jugarse o haber tenido lugar.

### 4. Postcondición si éxito.

Se mostrará la información disponible sobre el encuentro.

### 5. Postcondición si fallo.

Se mostrará un mensaje indicativo de error.

### 6. Entradas.

El identificador del partido a visualizar.

### 7. Salidas.

Información completa sobre el estado del encuentro.

### 8. Secuencia normal.

1. El usuario selecciona un encuentro a través de la sección del calendario.
2. Se comprueba si el encuentro ha tenido lugar o aún está pendiente. En caso de estar en juego o error, pasar a S-1.
3. Se recopilan los datos del encuentro en cuestión. En el caso de haberse jugado, se tomarán los equipos implicados, la fecha, el marcador, la crónica y opcionalmente algunas estadísticas. En caso de ser un encuentro pendiente, se tomarán los equipos, los datos de aforo, diferencia de niveles, etc. y demás datos relevantes. En caso de error, pasar a S-2.
4. Se presenta la información al usuario en una nueva vista.

### 9. Secuencia alternativa.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

S-1. Error, el partido está actualmente en juego. Mostrar un mensaje informativo y continuar la ejecución del sistema.

S-2. Error al recopilar los datos del encuentro. Mostrar un mensaje informativo del error y continuar la ejecución normal del sistema.

## Caso de uso: Consultar y modificar datos de usuario.

### 1. Objetivo.

Consultar y modificar los datos personales de un usuario.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario está registrado y correctamente autenticado en el sistema.

### 4. Postcondición si éxito.

Se mostrará un listado con los datos del usuario.

Si se han realizado cambios en la cuenta:

- Mensaje de confirmación de los cambios.
- Listado de los datos del usuario con los nuevos cambios realizados.

### 5. Postcondición si fallo.

No se realizarán los cambios y se mostrará mensaje indicativo del error.

### 6. Entradas.

De forma opcional, si se desea modificar el perfil, el usuario deberá completar los siguientes campos:

- Nuevo e-mail.
- Nueva contraseña.

### 7. Salidas.

Mensaje de confirmación o de error.

### 8. Secuencia normal.

1. El usuario selecciona la sección pertinente para acceder a sus datos personales.
  2. Se accede a la base de datos para tomar el nombre de usuario, afición, email y contraseña. En caso de error, pasar a S-1.
  3. Se muestran los datos al usuario en una nueva vista.
  4. Opcionalmente, el usuario puede seleccionar el cambio de contraseña. En este caso pasar al punto 6.
  5. Opcionalmente el usuario puede seleccionar el cambio de email. En este caso pasar al punto 7.
  6. Se crea una nueva vista en la que se permite que el usuario introduzca su antigua clave y la nueva (por duplicado).
- 6.1 Una vez rellenados los campos se procesa el formulario.

## JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

6.2 Se establece una conexión con la base de datos para verificar la información proporcionada. En caso de fallo de validación, pasar a S-2.

6.3 Si los datos son correctos, se procede a la modificación de la clave en la base de datos. En caso de error, pasar a S-3. En caso de éxito, pasar al punto 8.

7. Se crea una nueva vista en la que se permite que el usuario introduzca su antiguo email y el nuevo (por duplicado).

7.1 Una vez rellenados los campos se procesa el formulario.

7.2 Se establece una conexión con la base de datos para verificar la información proporcionada. En caso de fallo de validación, pasar a S-4.

7.3 Si los datos son correctos, se procede a la modificación del email en la base de datos. En caso de error, pasar a S-3.

8. Mostrar de nuevo la vista de información de la cuenta tras modificar los datos personales.

### 9. Secuencia alternativa.

S-1. Error al obtener los datos del sistema. Mostrar mensaje informativo y continuar la ejecución normal del sistema.

S-2. Error al validar los datos del cambio de clave. Informar de los campos inválidos, descartar los cambios y continuar la ejecución del sistema.

S-3. Error al introducir los cambios en el sistema. Informar al usuario y continuar la ejecución del sistema.

S-4. Error al validar los datos del cambio de email. Informar de los campos inválidos, descartar los cambios y continuar la ejecución del sistema.

## Caso de uso: ejecutar acciones de partido.

### 1. Objetivo.

Permitir la ejecución de acciones durante un encuentro para interactuar activamente con el mismo.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar previamente registrado en el sistema correctamente autenticado.

El encuentro debe estar en juego.

La acción de partido debe haber sido previamente desbloqueada.

### 4. Postcondición si éxito.

Se aplicarán los beneficios asociados a la acción de partido.

### 5. Postcondición si fallo.

Se mostrará un mensaje de error al usuario.

### 6. Entradas.

El identificador de la acción a ejecutar.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

## 7. Salidas.

Mensaje de error en caso de fallo.

## 8. Secuencia normal.

1. El usuario accede a un partido en juego de su propia afición. Ver caso de uso “asistir a un partido”.
2. El usuario selecciona una de las acciones de partido habilitadas para ser ejecutada.
3. Se comprueba que el usuario disponga de suficientes recursos. En caso contrario, pasar a S-1.
4. Se marca al jugador como participante del encuentro (con fines estadísticos) y se aplica el beneficio de la acción. En caso de error, pasar a S-2.
5. Se continúa con la ejecución normal del sistema sin redirecciones. La ejecución de acciones debe evitar, de ser posible, realizarse con una recarga de la página completa (usando tecnologías como Ajax).

## 9. Secuencia alternativa.

S-1. Error por recursos insuficientes. Se informa al usuario y se continúa con la visualización del encuentro.

S-2. Error al actualizar los datos en la base de datos. Se deshacen los cambios anteriores y se informa al usuario de que debe intentar ejecutar la acción de nuevo. Asimismo se continúa con la visualización del partido.

## Caso de uso: ejecutar acciones grupales.

### 1. Objetivo.

Abrir una nueva acción grupal en la afición para permitir la participación del resto de miembros en la misma.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar previamente registrado en el sistema correctamente autenticado.

No debe haber otra acción abierta del mismo tipo y usuario en la afición.

El jugador debe disponer de dicha acción desbloqueada.

### 4. Postcondición si éxito.

La nueva acción será creada y visible al resto de miembros de la afición.

### 5. Postcondición si fallo.

Se mostrará un mensaje de error al usuario.

### 6. Entradas.

El identificador de la acción a crear.

### 7. Salidas.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

Mensaje de error en caso de fallo.

## 8. Secuencia normal.

1. El usuario accede a la sección de habilidades desbloqueadas de la aplicación. Una vez allí debe seleccionar aquella acción grupal que desea ejecutar.

2. Se comprueba en la base de datos que no existe otra acción del mismo tipo abierta por el usuario en la afición. Si existe, pasar a S-1.

2.1 Opcionalmente, se informa al usuario en caso de existir una acción del mismo tipo pero abierta por otro usuario por si prefiere participar en ella.

3. Se verifica que el usuario dispone de recursos suficientes para llevar a cabo la acción. En caso de error, pasar a S-2.

4. Se procede a la creación de los nuevos recursos:

4.1 Se restan los recursos asociados al coste de la acción de los del usuario.

En caso de error, pasar a S-3.

4.2 Se crea una nueva entrada en la base de datos para la acción grupal abierta. En caso de error, pasar a S-4.

4.3 Se crea una nueva entrada en la base de datos para la participación inicial del creador. En caso de error, pasar a S-5.

5. Se informa al usuario de la creación de la nueva acción y se muestran los datos básicos de la misma. Opcionalmente, se pueden mostrar también los recursos restantes tras la ejecución.

## 9. Secuencia alternativa.

S-1. Error por haber otra acción del mismo tipo y usuario abierta. Se redirige al usuario a la sección de participación en dicha grupal sin informar del error subyacente.

S-2. Error de recursos insuficientes. Se informa al usuario y se redirige a la sección de habilidades desbloqueadas.

S-3. Error al restar recursos. Se informa al usuario y se redirige a la sección de habilidades desbloqueadas.

S-4. Error al crear la entrada de la acción grupal. Se descartan los cambios realizados, se informa al usuario de error interno de servidor y se redirige a la sección de habilidades desbloqueadas.

S-5. Error al crear la entrada de participación. Se descartan los cambios realizados, se informa al usuario de error interno de servidor y se redirige a la sección de habilidades desbloqueadas.

## Caso de uso: ejecutar acciones individuales.

### 1. Objetivo.

Ejecutar una acción individual del jugador con el fin de obtener la bonificación asociada a la misma.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

El usuario debe estar previamente registrado en el sistema y correctamente autenticado. Deben satisfacerse las siguientes condiciones:

- La acción está en el árbol de habilidades del usuario.
- La acción está desbloqueada.
- La acción está preparada para ser ejecutada.
- El usuario cumple con los requisitos necesarios para poder ejecutarla.

## 4. Postcondición si éxito.

Acción ejecutada con éxito. Se aplicará la bonificación correspondiente y la acción entrará en cooldown.

## 5. Postcondición si fallo.

La acción no se ejecutará y se mostrará un mensaje de error.

## 6. Entradas.

Acción que selecciona el usuario para ejecutar.

## 7. Salidas.

Mensaje de confirmación o de error.

## 8. Secuencia normal.

1. El usuario accede a la sección de acciones desbloqueadas.
2. El usuario selecciona la acción que desea ejecutar. Si dicha acción está en cooldown, pasar a S-1<sup>1</sup>.
  3. Se accede a la base de datos y se realizan las operaciones necesarias:
    - 3.1 Restar los recursos asociados al coste de dicha acción. En caso de error pasar a S-2.
    - 3.2 Refrescar el cooldown para evitar el uso continuado de la misma. En caso de error pasar a S-3.
    - 3.3 Aplicar el beneficio de la acción al personaje. En caso de error pasar a S-4.
  4. Mostrar al usuario un mensaje de confirmación y mantenerlo en la vista de acciones desbloqueadas.

## 9. Secuencia alternativa.

S-1. Error, habilidad en cooldown. Se informa al usuario del error y se continúa la ejecución normal del sistema..

S-2. Error al restar los recursos. Se informa del error y se continúa la ejecución normal del sistema.

S-3. Error al fijar el nuevo cooldown. Se revierten los cambios anteriores, se informa del error y se continúa con la ejecución normal del sistema.

S-4. Error al aplicar el beneficio. Se revierten los cambios anteriores, se informa del error y se continúa con la ejecución normal del sistema.

---

<sup>1</sup> La comprobación consta en la secuencia normal a pesar de ser una precondición con el fin de hacer hincapié en la misma.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

## Caso de uso: enviar mensaje privado

### 1. Objetivo.

Enviar un mensaje privado a otro jugador.

### 2. Actores.

Usuario registrado remitente, usuario registrado destinatario.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario remitente debe estar previamente registrado en el sistema y correctamente autenticado.

El usuario destinatario debe estar previamente registrado en el sistema.

### 4. Postcondición si éxito.

Se creará un nuevo mensaje con el contenido deseado y se enviará del remitente al destinatario.

### 5. Postcondición si fallo.

No se enviará el mensaje y se mostrará un mensaje de error al usuario.

### 6. Entradas.

Identificador del usuario al que se desea mandar el mensaje, asunto y el contenido del mismo.

### 7. Salidas.

Mensaje de confirmación o de error.

### 8. Secuencia normal.

1. El usuario accede a la sección de mensajería privada.
2. El usuario selecciona la opción de enviar nuevo mensaje. Opcionalmente, puede haber otro mecanismo auxiliar que facilite el envío de mensajes privados. Por ejemplo, un botón al lado de los nombres de usuario en las listas del sistema.
3. Se muestra un formulario al usuario donde deberá llenar el identificador del destinatario del mensaje (si ha sido generado de forma automática), el asunto y el contenido del mismo.
4. Una vez completados los datos, se procede al envío del formulario haciendo uso de la opción (botón) oportuna.
5. Se accede a la base de datos y se procesa la información recibida.
  - 5.1. Se valida el contenido de los campos. En caso de error, pasar a S-1.
  - 5.2. Se crea la nueva entrada para el mensaje en el buzón del destinatario. En caso de error, pasar a S-2.
6. Se muestra un mensaje informativo de éxito al usuario y se continúa la ejecución del sistema.

### 9. Secuencia alternativa.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

S-1. Error al validar los campos del formulario. Puede ser debido a que los campos no cumplen las restricciones o bien a que el usuario no existe. Se muestra un mensaje con los campos incorrectos y se vuelve a la sección del mensaje privado.

S-2. Error al crear el mensaje. Se muestra un mensaje al usuario indicándole el problema y se continúa la ejecución normal del sistema.

## Caso de uso: expulsar jugador.

### 1. Objetivo.

Permitir la expulsión de un jugador como participante de una acción grupal abierta.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar previamente registrado en el sistema correctamente autenticado.

La acción grupal debe existir y no haber finalizado.

El jugador a expulsar debe formar parte de los participantes de la acción de afición.

El jugador que expulsa debe ser el creador de la acción.

### 4. Postcondición si éxito.

El jugador será expulsado de la acción, su aportación retirada de la misma y los recursos devueltos al participante.

### 5. Postcondición si fallo.

Se mostrará un mensaje de error al usuario.

### 6. Entradas.

El identificador del participante a expulsar.

### 7. Salidas.

Mensaje de error en caso de fallo.

### 8. Secuencia normal.

1. El usuario accede a la sección de acciones abiertas por la afición y selecciona la vista de información avanzada de una creada por el mismo. En esta vista aparecerá una lista de participantes en dicha acción.

2. De esta lista, el usuario elige el jugador al que desea echar y selecciona la opción de expulsar.

3. Se llevan a cabo algunas comprobaciones de seguridad:

3.1 Se verifica con la base de datos que el creador es el usuario peticionario. Si error, pasar a S-1.

3.2 Se verifica que el jugador a expulsar es un participante de la acción. Si error, pasar a S-2.

4. Se retiran los recursos destinados por el jugador expulsado en la acción y se le devuelven. Si error, pasar a S-3.

## JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

5. Se elimina el jugador de la lista de participantes y se hacen los ajustes pertinentes en los datos de la acción grupal. Si error, pasar a S-4.

### 9. Secuencia alternativa.

S-1. Error, el solicitante no es el creador de la acción. Se informa al usuario de que no dispone de permiso para llevar a cabo la acción y se redirige a la sección de información del evento de afición.

S-2. Error, el jugador a expulsar no es un participante. Se informa al usuario del uso indebido de la acción y se redirige a la sección de información del evento grupal.

S-3. Error al retirar los recursos. Se informa de error interno del servidor al usuario y se redirige a la sección de información avanzada de la acción grupal.

S-4. Error al eliminar al jugador expulsado de la lista de participantes. En este caso se deshacen todos los cambios realizados, se informa al usuario del error y se redirige a la sección de información avanzada de la acción de afición.

### Caso de uso: participar en acciones grupales.

#### 1. Objetivo.

Aportar recursos a una acción grupal abierta en la afición.

#### 2. Actores.

Usuario registrado.

#### 3. Precondición.

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar previamente registrado en el sistema y correctamente autenticado.

La acción debe haber sido creada previamente.

La acción aún no ha expirado.

El usuario cumple los requisitos necesarios para participar (tener recursos).

#### 4. Postcondición si éxito.

Los recursos aportados se sumarán a los acumulados en la acción y se restarán de los del usuario. En caso de conseguir completar la acción, se aplicará el beneficio de la misma y se dará por completada.

#### 5. Postcondición si fallo.

Se mostrará un mensaje indicativo del error.

#### 6. Entradas.

Identificador de la acción en la que el usuario desea participar y cantidad de recursos a aportar.

#### 7. Salidas.

Mensaje de confirmación o error.

#### 8. Secuencia normal.

1. El usuario accede a la sección de la afición y solicita información avanzada de una de las acciones grupales abiertas (ver casos de uso asociados).

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

2. El usuario escoge la opción de participar en dicha acción.
3. Se muestra un nuevo formulario para aportar recursos. El usuario debe rellenarlo de forma apropiada. Esto es, por ejemplo, sin dejar todos los recursos a 0 o valores negativos.
4. Se toman los datos del formulario y se validan. En caso de error, pasar a S-1.
5. Se ejecutan las comprobaciones antes de agregar la nueva participación.
  - 5.1 Si la acción ha finalizado, pasar a S-2.
  - 5.2 Si la acción ha alcanzado el número máximo de participantes y el nuevo colaborador no se encuentra entre ellos, pasar a S-3.
6. Se acumulan los nuevos recursos y se restan de los del usuario.
  - 6.1 Si el usuario ya había participado, no se agrega un nuevo participante sino que se acumulan los recursos al ya existente.
  - 6.2 Si el usuario no había participado, se incrementa el número de participantes y se asignan como participación los recursos aportados.
7. Se muestran de nuevo los datos de información avanzada de la acción grupal al usuario junto con un mensaje de confirmación.

## 9. Secuencia alternativa.

- S-1. Error al validar los campos. Se informa al usuario de que debe rellenarlos de forma correcta y se muestra de nuevo el formulario de participación.
- S-2. Error, la acción ya ha expirado o se ha completado. Se informa del error y se redirige a la vista de la afición.
- S-3. Error por número máximo de participantes. Se informa al usuario y se redirige a la vista de la afición.

## Caso de uso: registrar usuario.

### 1. Objetivo

Validar y tomar los datos de un formulario e introducirlos en el sistema.

### 2. Actores

Usuario invitado.

### 3. Precondición

Es posible establecer conexión con la base de datos.

### 4. Postcondición si éxito

Se creará un nuevo usuario con los datos proporcionados.

### 5. Postcondición si fallo

Se mostrará un mensaje informativo del error al usuario.

### 6. Entradas

Los datos del formulario serán los siguientes:

- Nombre de usuario (campo obligatorio).
- Contraseña (campo obligatorio).
- Repetir contraseña (campo obligatorio).

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

- E-mail (campo obligatorio).

## 7. Salidas

Mensaje de confirmación o de error.

## 8. Secuencia normal

1. El usuario accede a la sección de registro a través de la página principal de la aplicación. Dado que el registro de usuario se divide en 3 fases para permitir completarlas por separado, es posible entrar a la secuencia normal en uno de sus puntos intermedios (véase caso de uso de Autenticación de usuarios).
2. Se muestra una nueva sección con un formulario que permite introducir un nombre de usuario, email y contraseña (por duplicado).
3. Una vez llenados estos datos, se validan. Si alguno de ellos ha sido introducido de forma incorrecta o no es válido, pasar a S-1.
4. Si todos los datos son correctos, se crea una nueva entrada en la base de datos para el usuario rellenando tan sólo los datos proporcionados. En caso de error, pasar a S-2.
5. Se muestra una nueva vista que permite seleccionar la afición a la que se desea unir el usuario.
6. Una vez elegida ésta, se incorpora la nueva información a la base de datos para continuar complementando los datos de usuario. En caso de error, pasar a S-3.
7. A continuación, se muestra una vista que permite seleccionar el perfil del usuario (animadora, ultra o empresario).
8. Una vez elegido uno de ellos, se incorpora de nuevo esta información a la base de datos. Asimismo, se crean las entradas necesarias en el resto de tablas (por ejemplo, recursos). En caso de error pasar a S-4.
9. Se muestra un mensaje informativo de éxito al usuario y se redirige a la página principal de la aplicación.

## 9. Secuencia alternativa

- S-1. Error al validar los campos del formulario. Se informa de los errores cometidos y se vuelve a la vista del formulario.
- S-2. Error al crear la nueva entrada de usuario. Se informa del error y se solicitan de nuevo todos los datos.
- S-3. Error al actualizar la afición. Se informa del error y se descartan los datos. El usuario deberá volver a seleccionar una afición.
- S-4. Error al incorporar la información del perfil o crear las tablas finales. Se informa del error y el usuario deberá introducir de nuevo los datos restantes.

## Caso de uso: ver mensaje privado

### 1. Objetivo.

Leer un mensaje privado recibido por de jugador.

### 2. Actores.

Usuario registrado.

### 3. Precondición.

# JUGADOR #12

Documentación técnica: casos de uso

Ingeniería del Software, 2013

Debe ser posible establecer una conexión con la base de datos.

El usuario debe estar previamente registrado en el sistema y correctamente autenticado.

## 4. Postcondición si éxito.

Se mostrará el contenido del mensaje seleccionado.

## 5. Postcondición si fallo.

Se mostrará un mensaje de error al usuario.

## 6. Entradas.

Identificador del mensaje que se desea visualizar.

## 7. Salidas.

Contenido del mensaje en caso de éxito o mensaje de error en caso de fallo.

## 8. Secuencia normal.

1. El usuario accede a la sección de mensajería privada.

2. El usuario selecciona la opción de la bandeja de entrada para ver todos los mensajes disponibles.

3. El usuario selecciona uno de ellos para ser visualizado.

4. Se accede a la base de datos para tomar los datos asociados a dicho mensaje.

En caso de error, pasar a S-1.

5. Se muestra al usuario una nueva vista con los datos del mensaje en cuestión.

## 9. Secuencia alternativa.

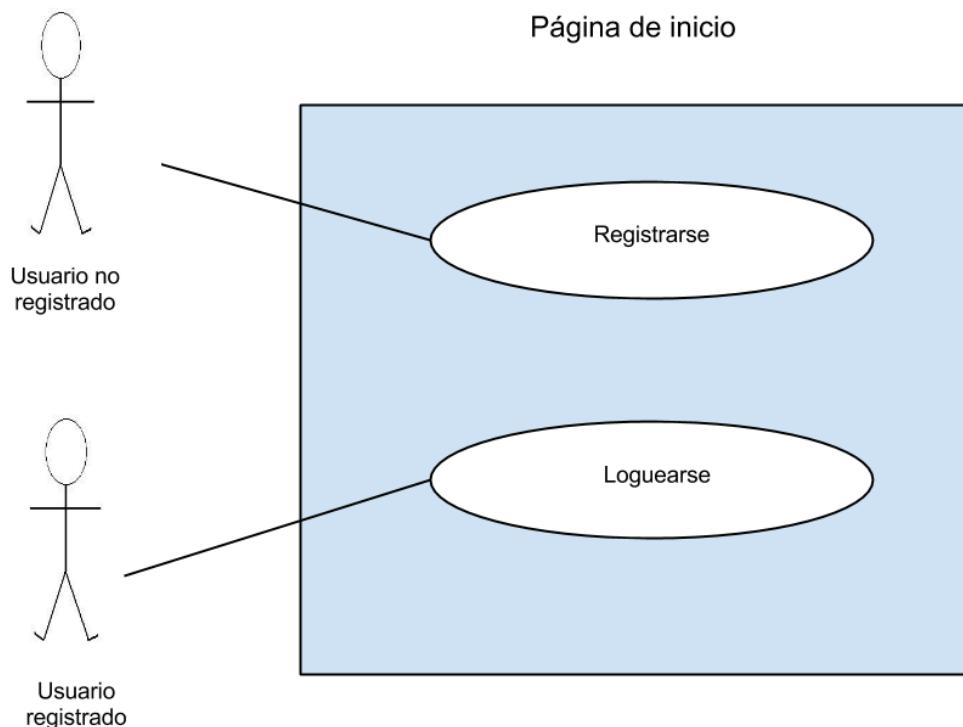
S-1. Error al recopilar los datos del mensaje. Se muestra un mensaje de error y se vuelve a la bandeja de entrada para continuar la ejecución del sistema.

# JUGADOR #12

Documentación técnica: diagramas de casos de uso

Ingeniería del Software, 2013

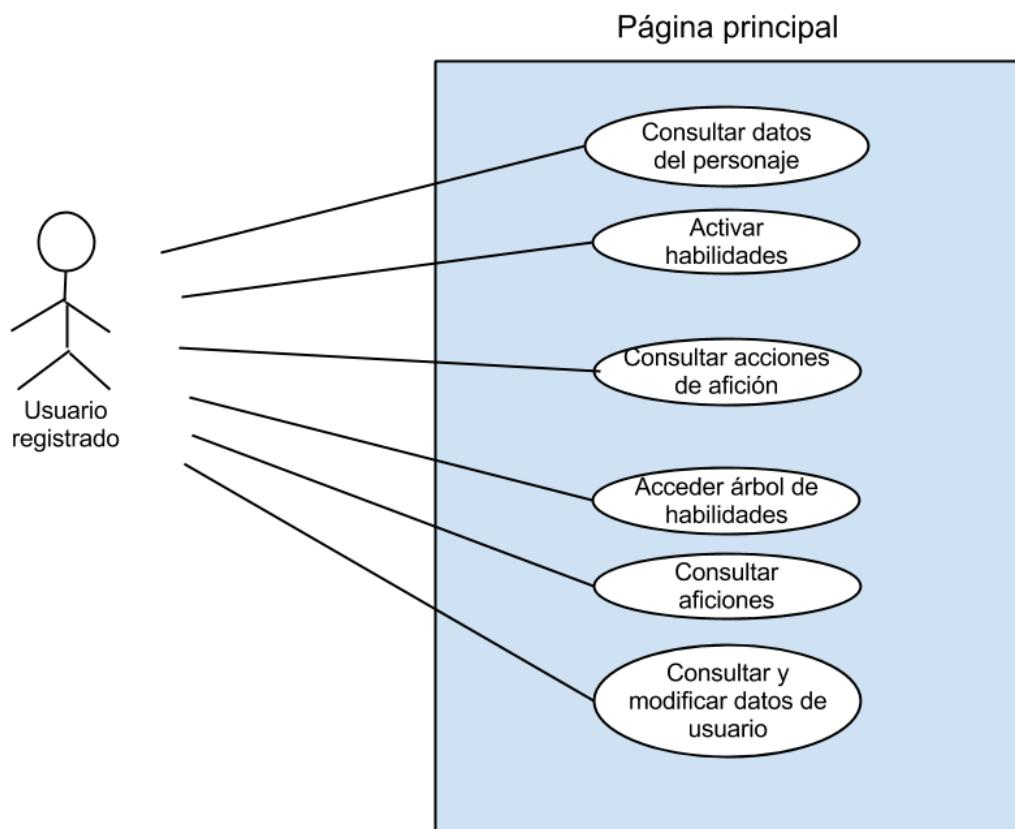
## Diagramas de casos de uso



# JUGADOR #12

Documentación técnica: diagramas de casos de uso

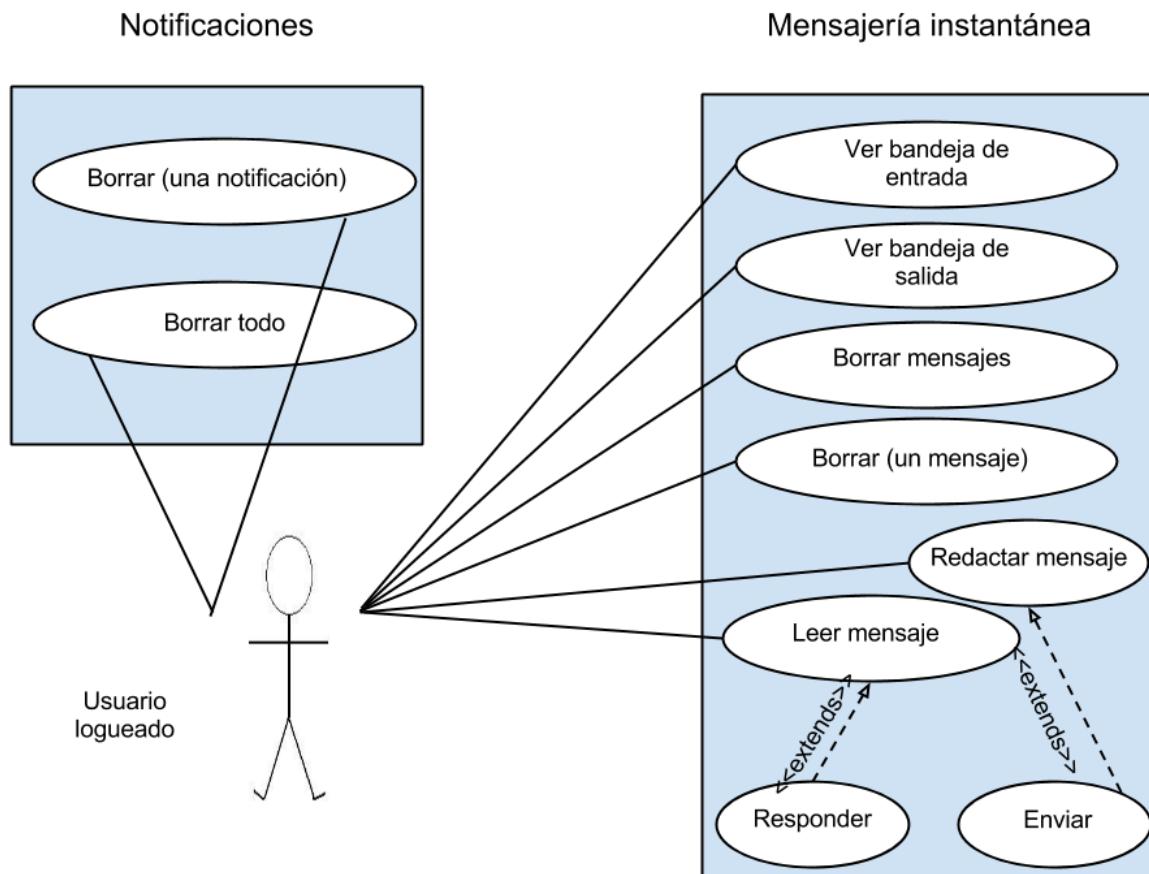
Ingeniería del Software, 2013



# JUGADOR #12

Documentación técnica: diagramas de casos de uso

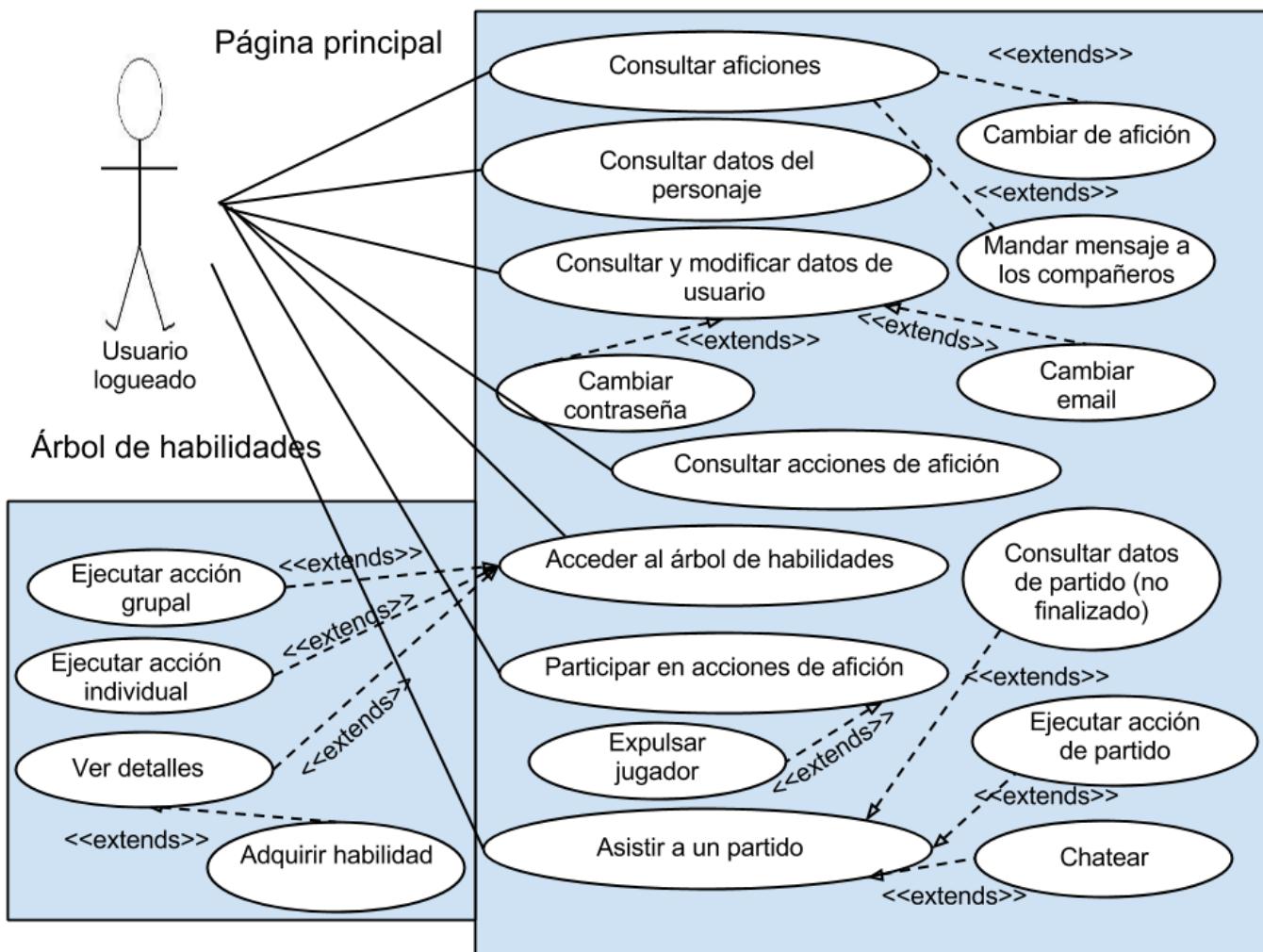
Ingeniería del Software, 2013



# JUGADOR #12

Documentación técnica: diagramas de casos de uso

Ingeniería del Software, 2013



## JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

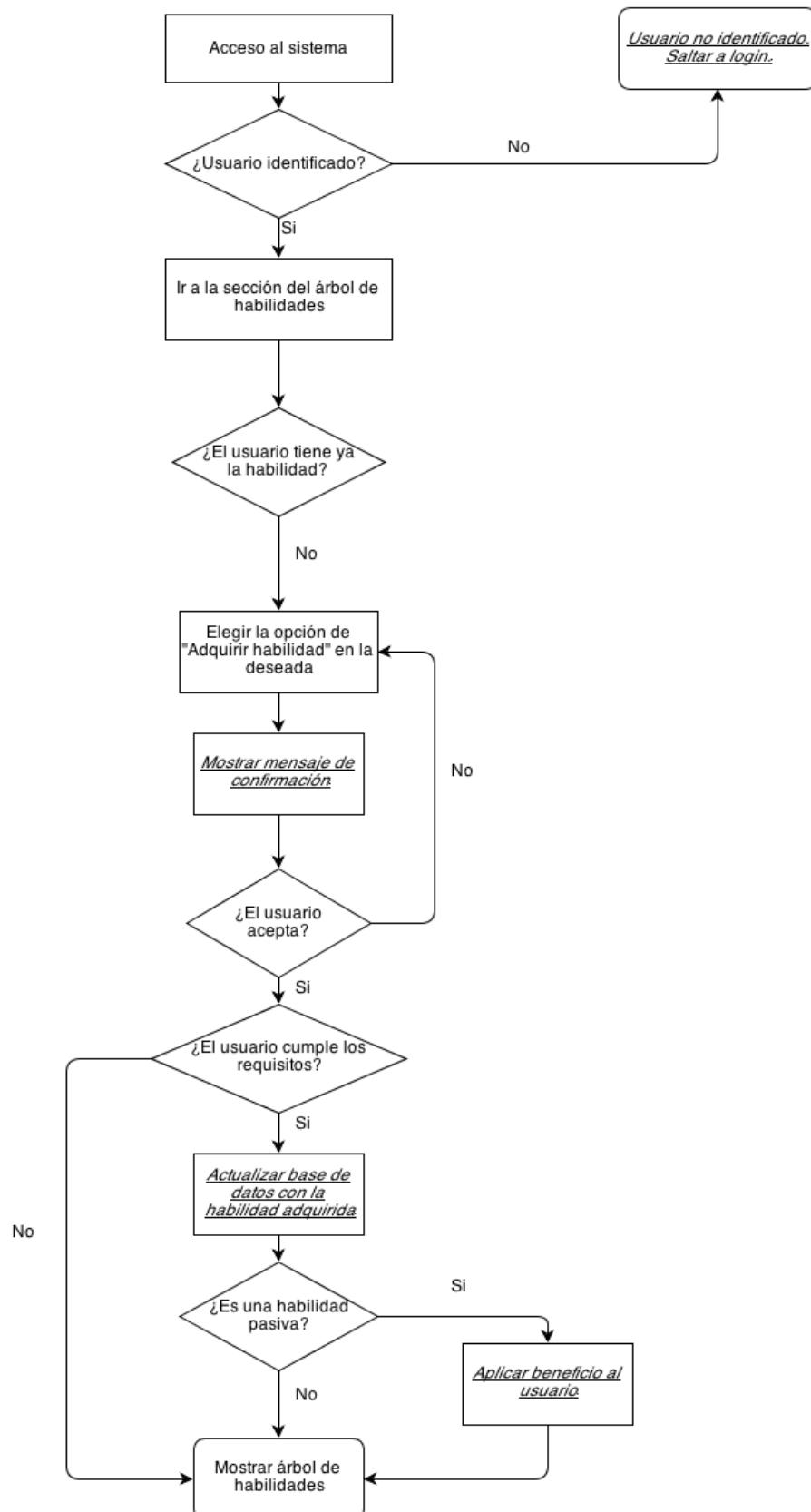
Ingeniería del Software, 2013

### Diagramas de flujo: índice

1. Adquirir habilidad
2. Borrar notificación o mensaje privado
3. Cambiar de equipo
4. Cambiar email o clave
5. Consultar datos
6. Ejecutar acción de partido
7. Ejecutar acción grupal
8. Ejecutar acción individual
9. Ejecutar acción (general)
10. Enviar mensaje privado
11. Expulsar jugador de acción grupal
12. Leer mensajes privados
13. Login
14. Participar en acción grupal
15. Registro
16. Visualizar crónica/previa/asistencia a un encuentro
17. Visualizar estado de acción grupal
18. Visualizar información de otro usuario
19. Visualizar información de un equipo
20. Visualizar información de una habilidad

# JUGADOR #12

## Adquirir habilidad

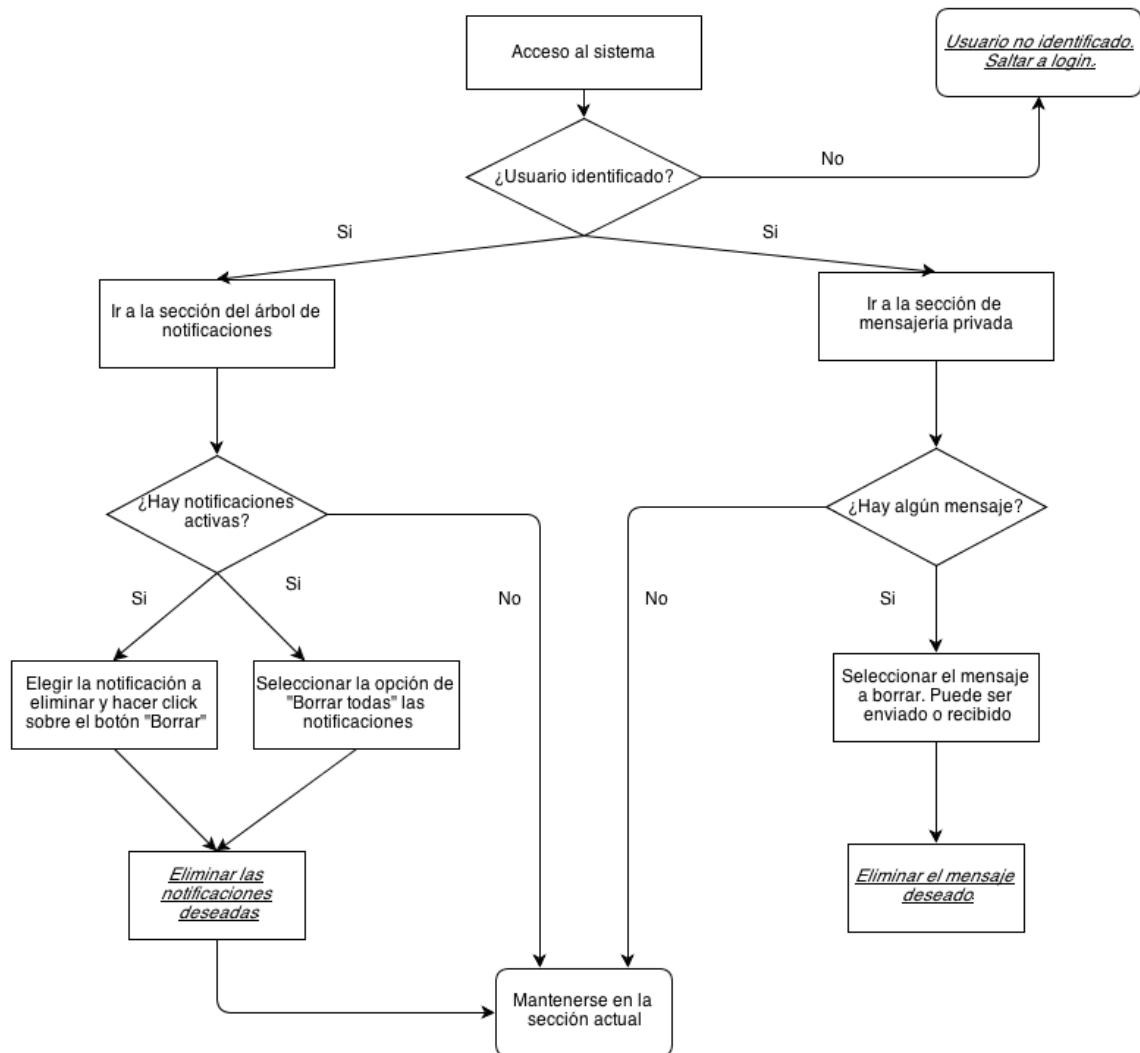


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Borrar notificación o mensaje privado

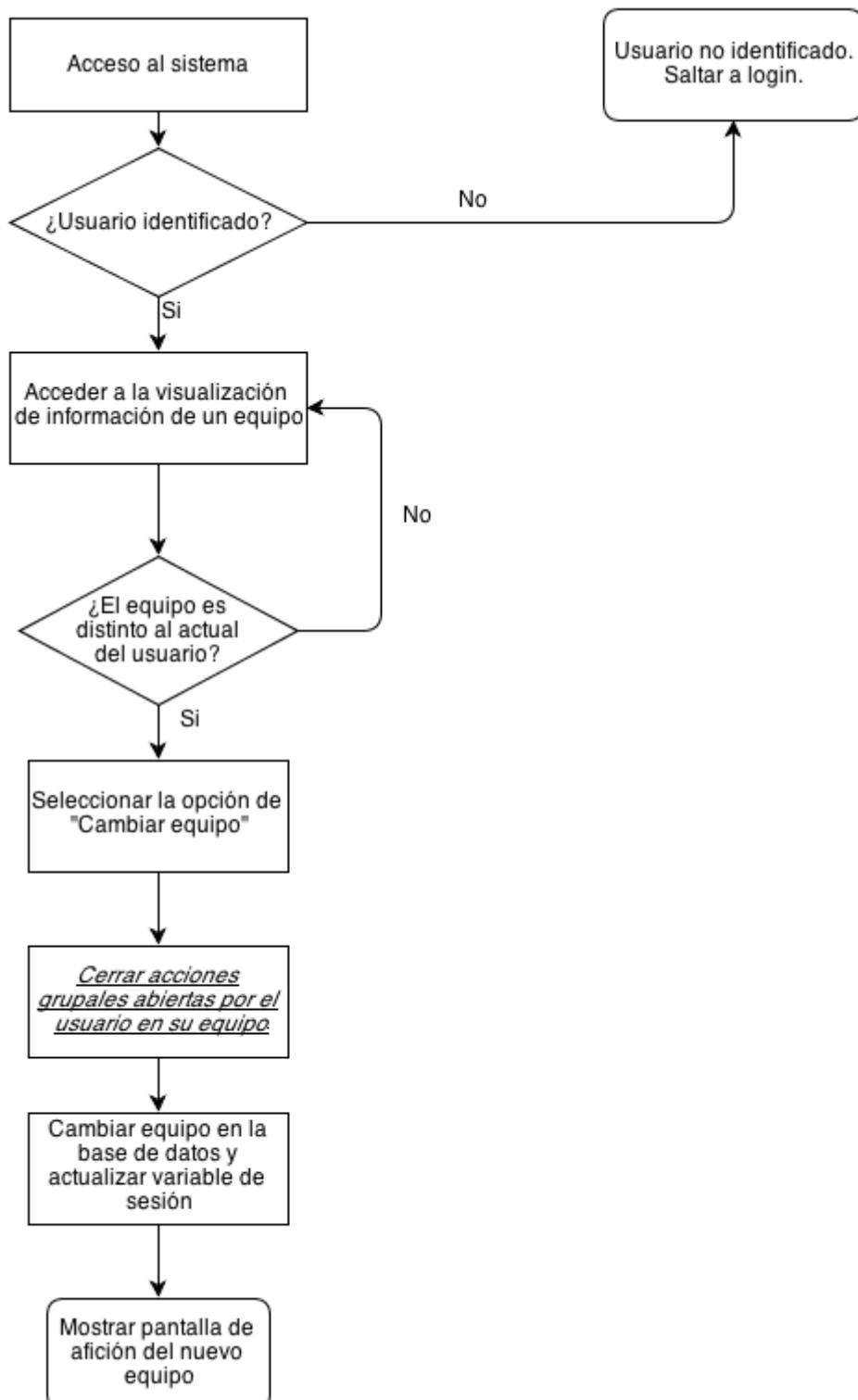


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Cambiar de equipo

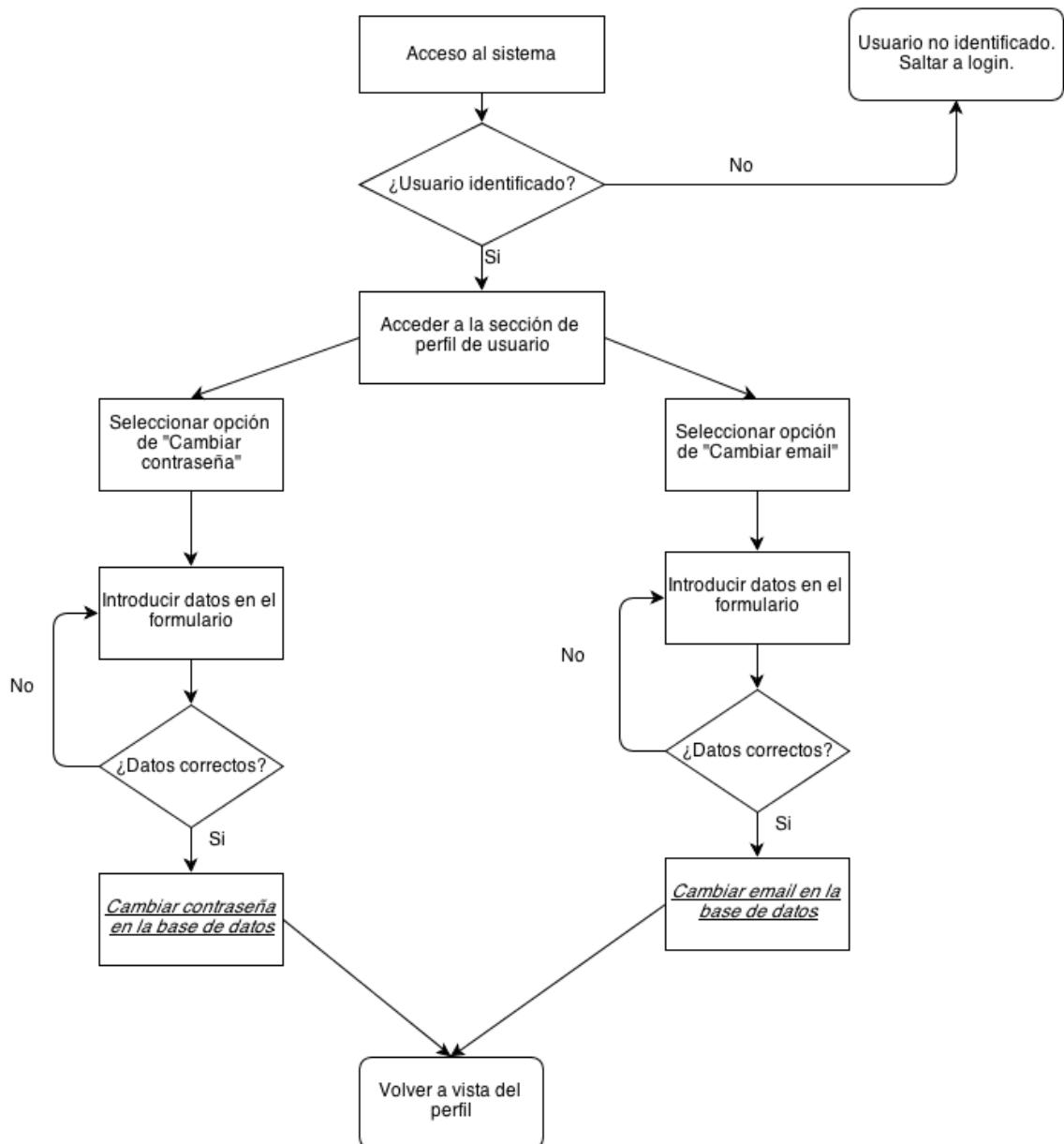


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Cambiar email o clave

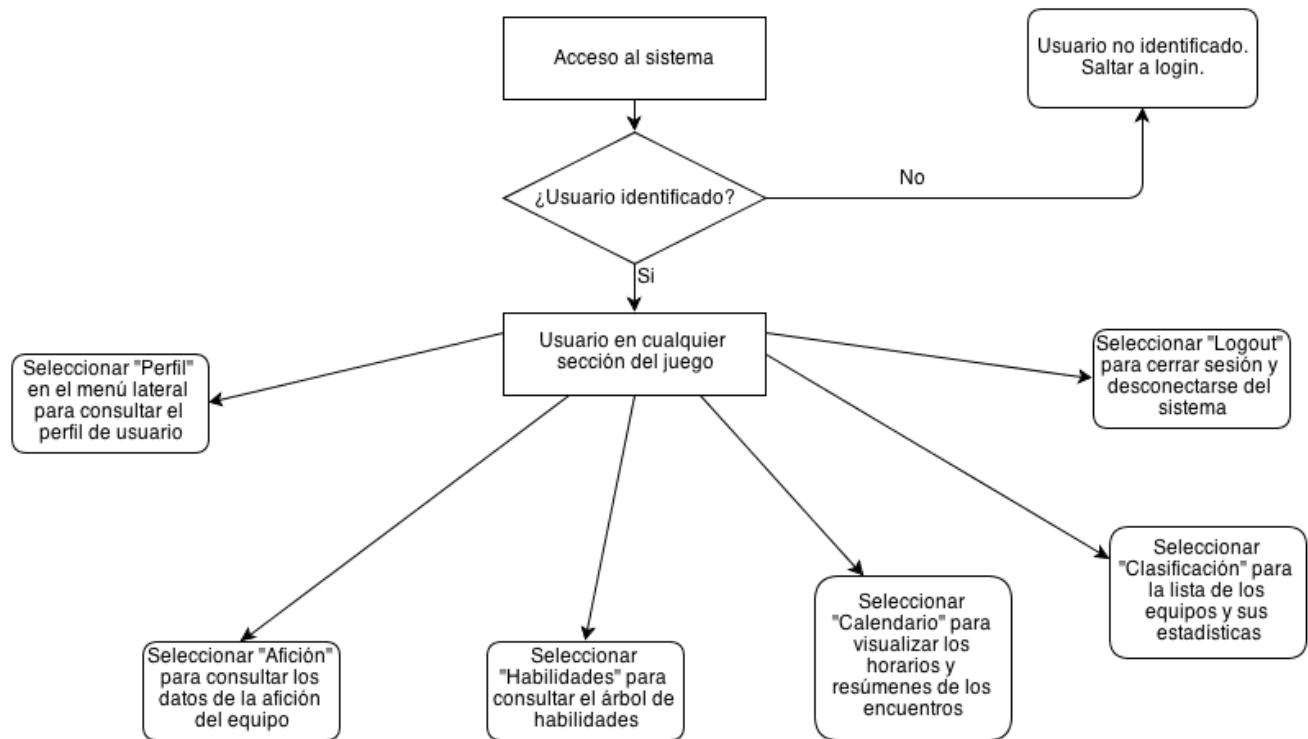


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Consultar datos

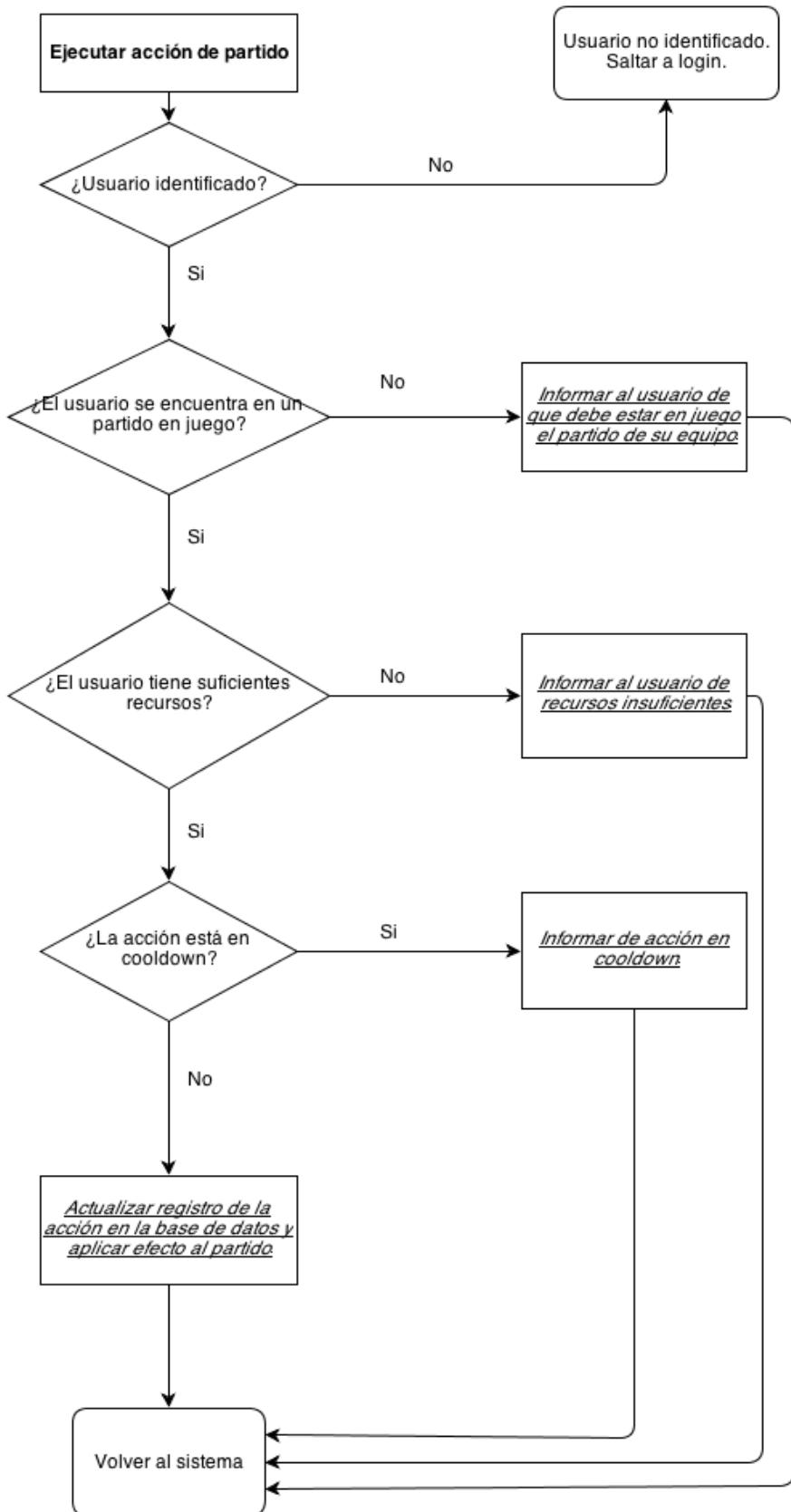


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Ejecutar acción de partido

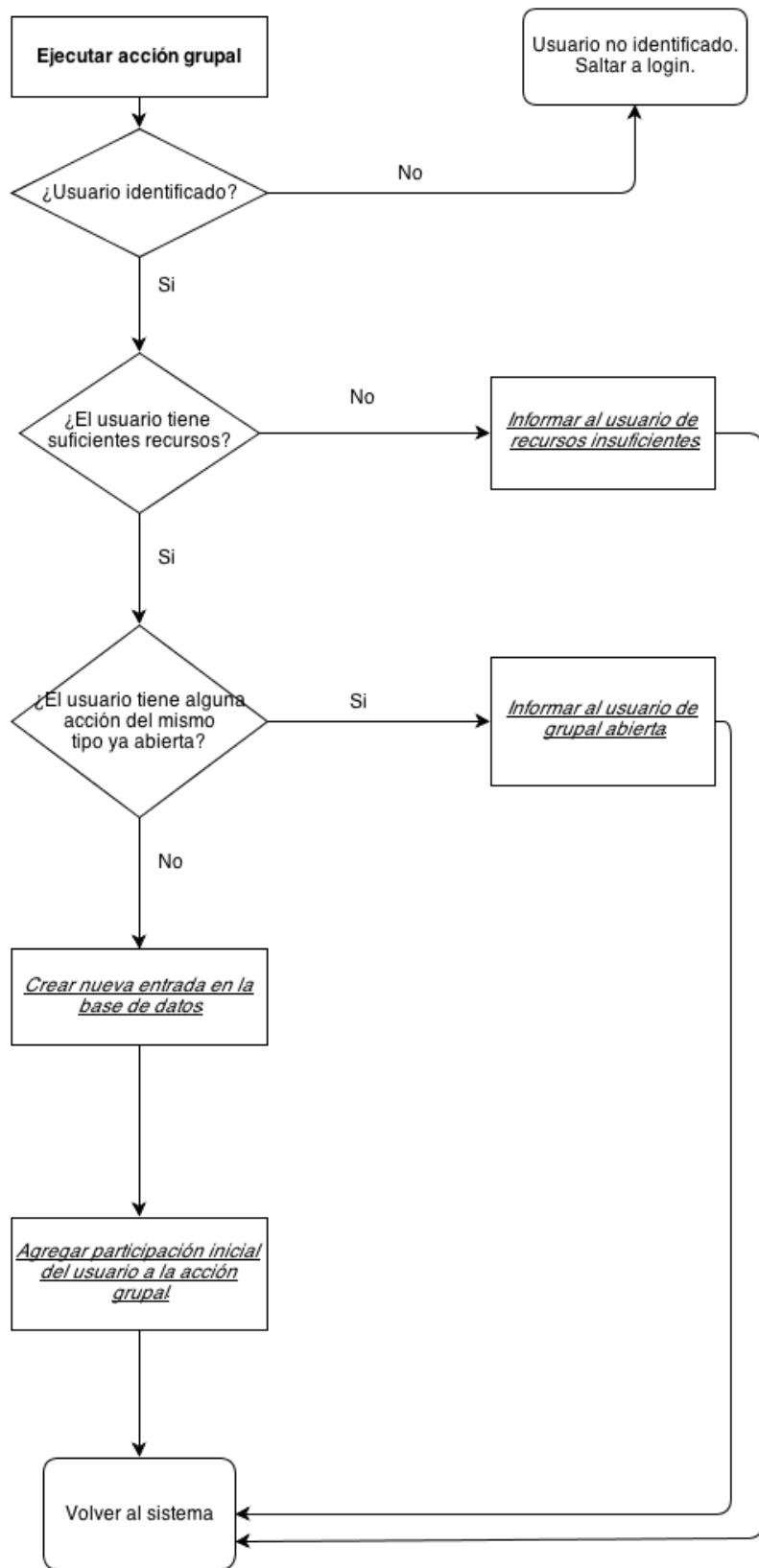


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Ejecutar acción grupal

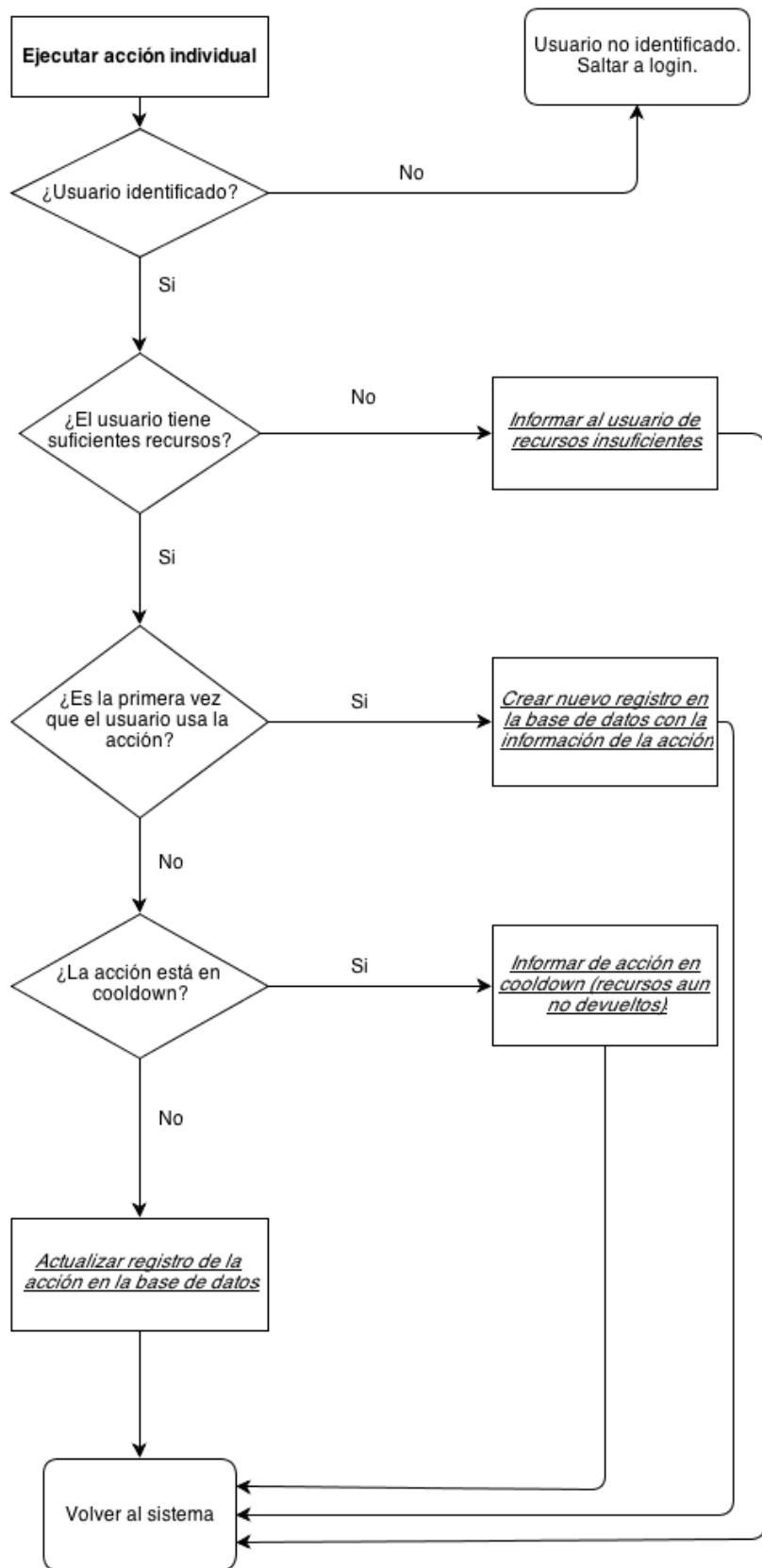


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Ejecutar acción individual

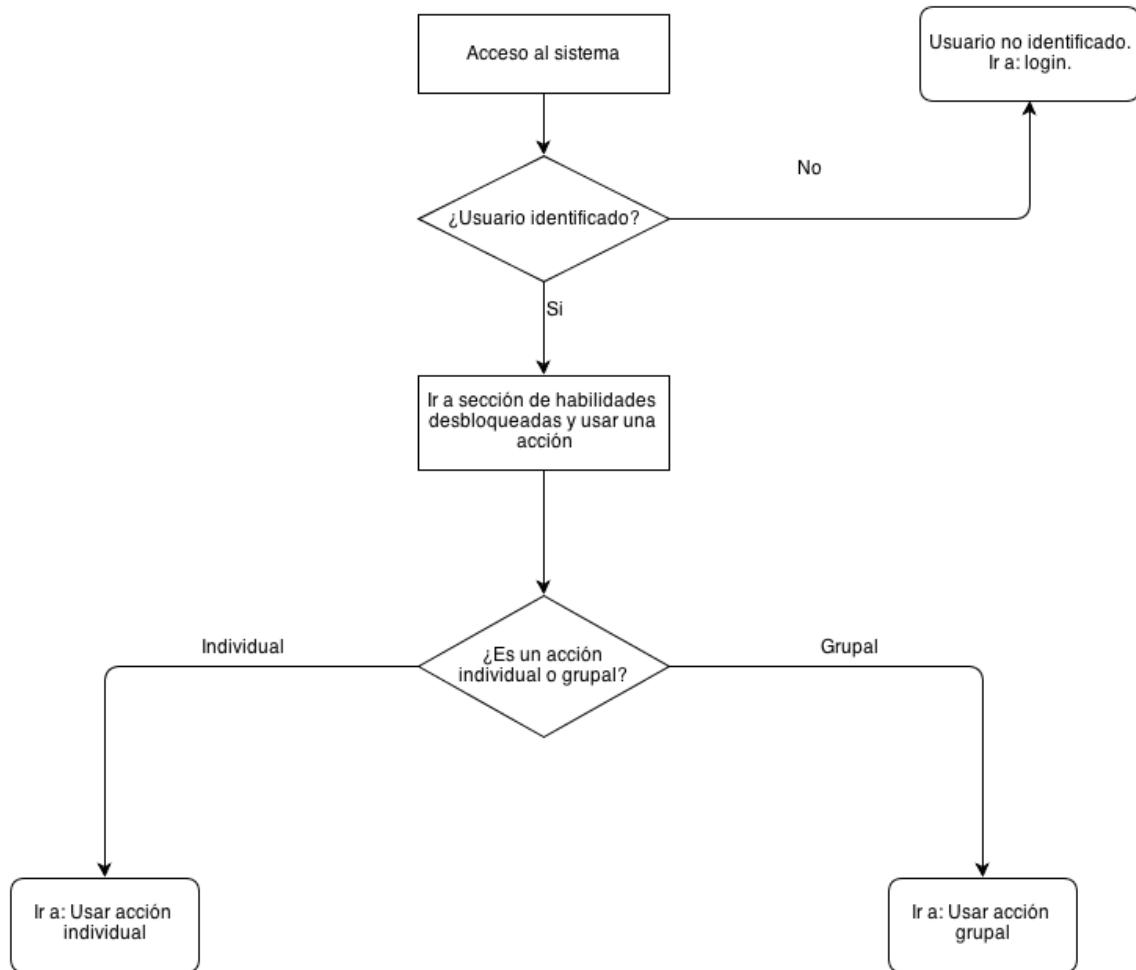


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

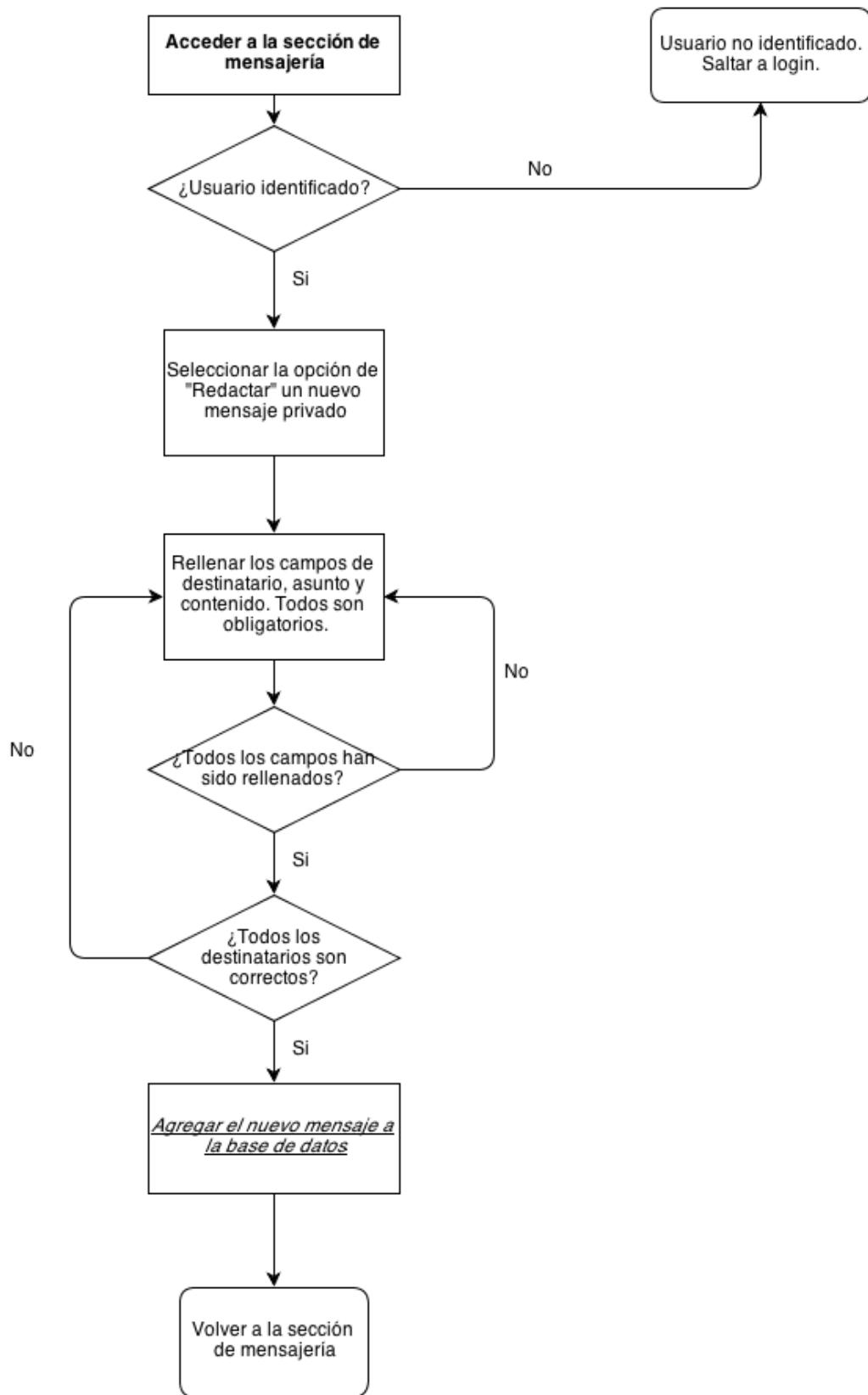
Ingeniería del Software, 2013

## Ejecutar acción (general)



# JUGADOR #12

## Enviar mensaje privado

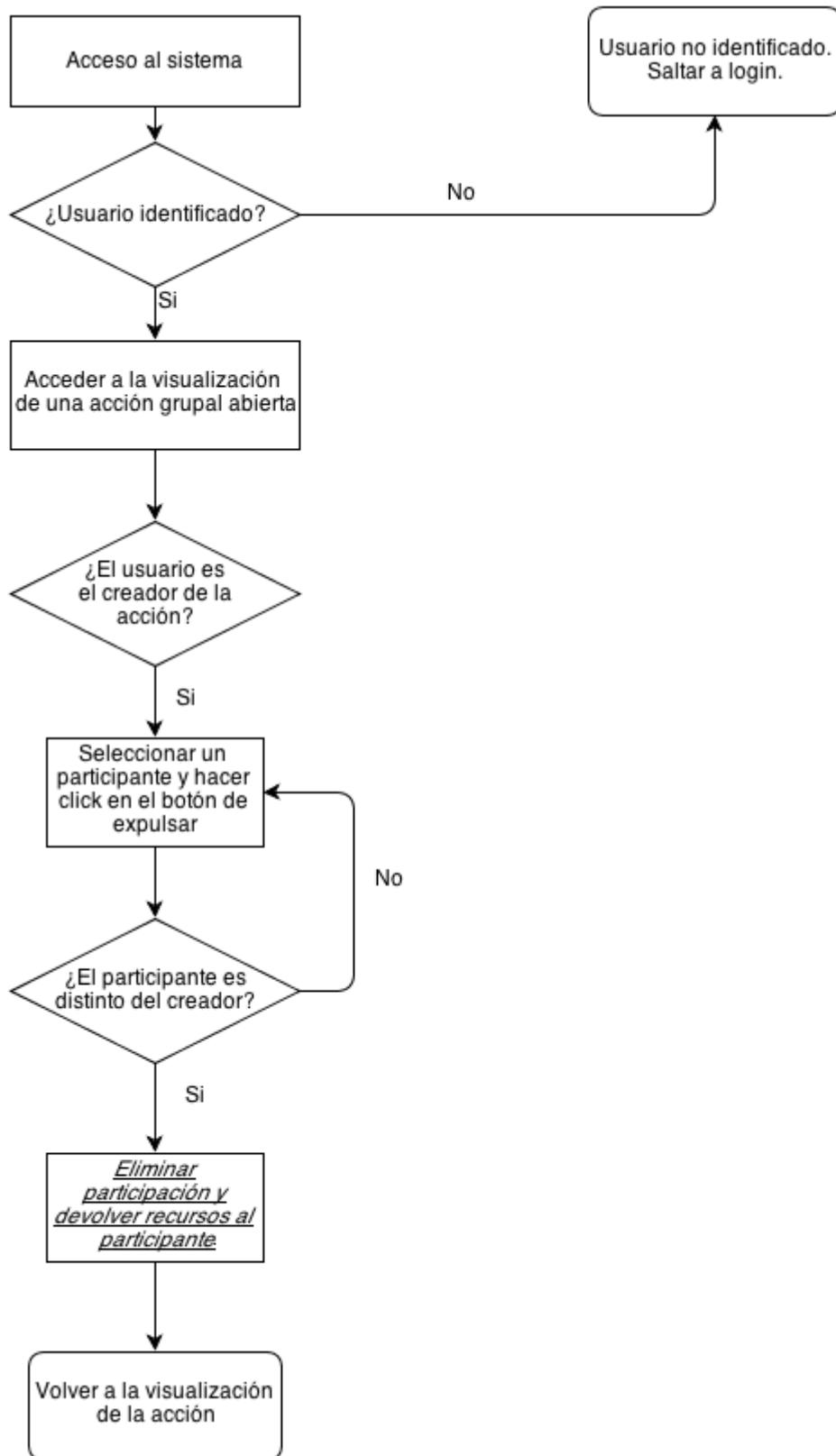


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Expulsar jugador de acción grupal

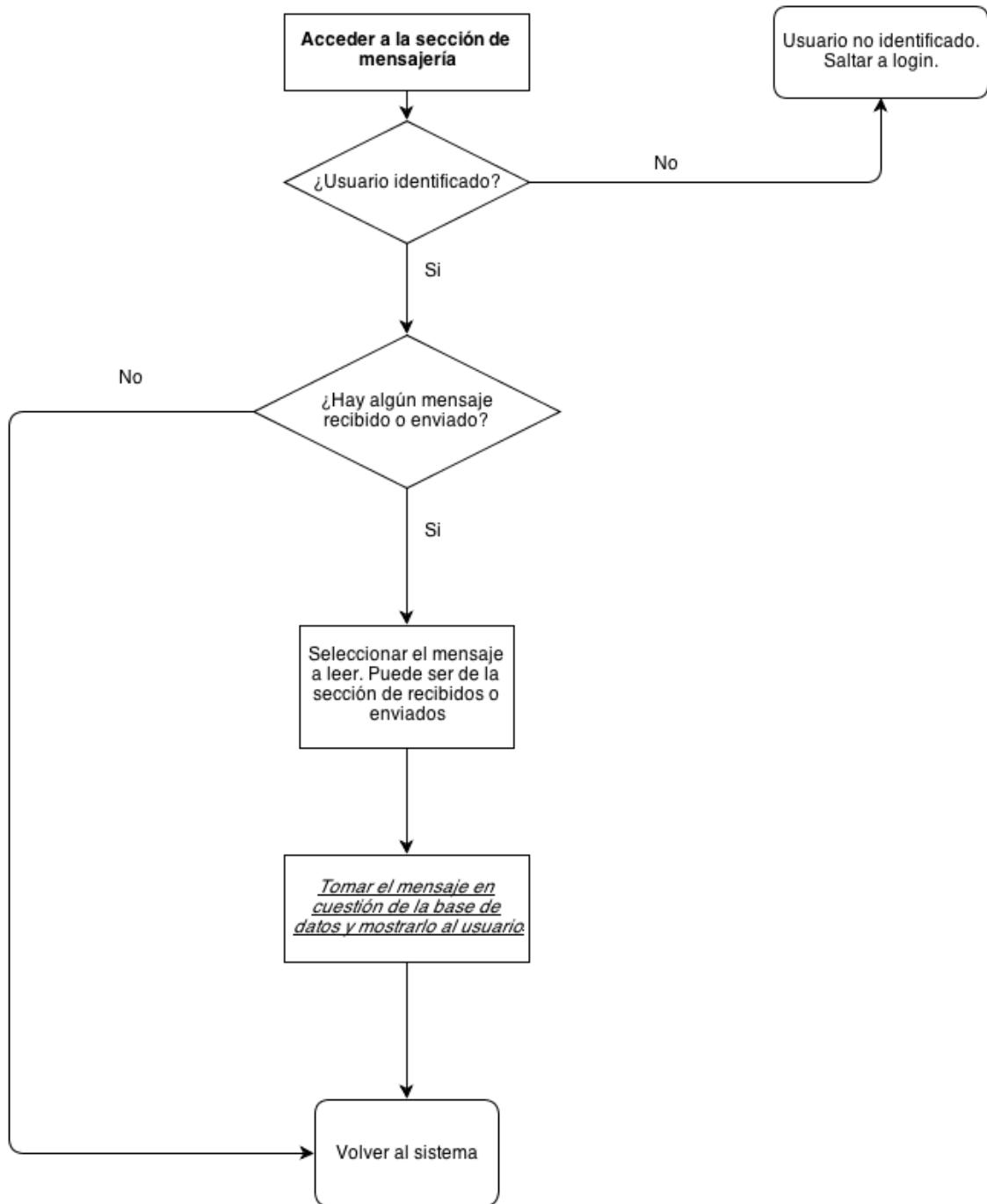


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Leer mensajes privados

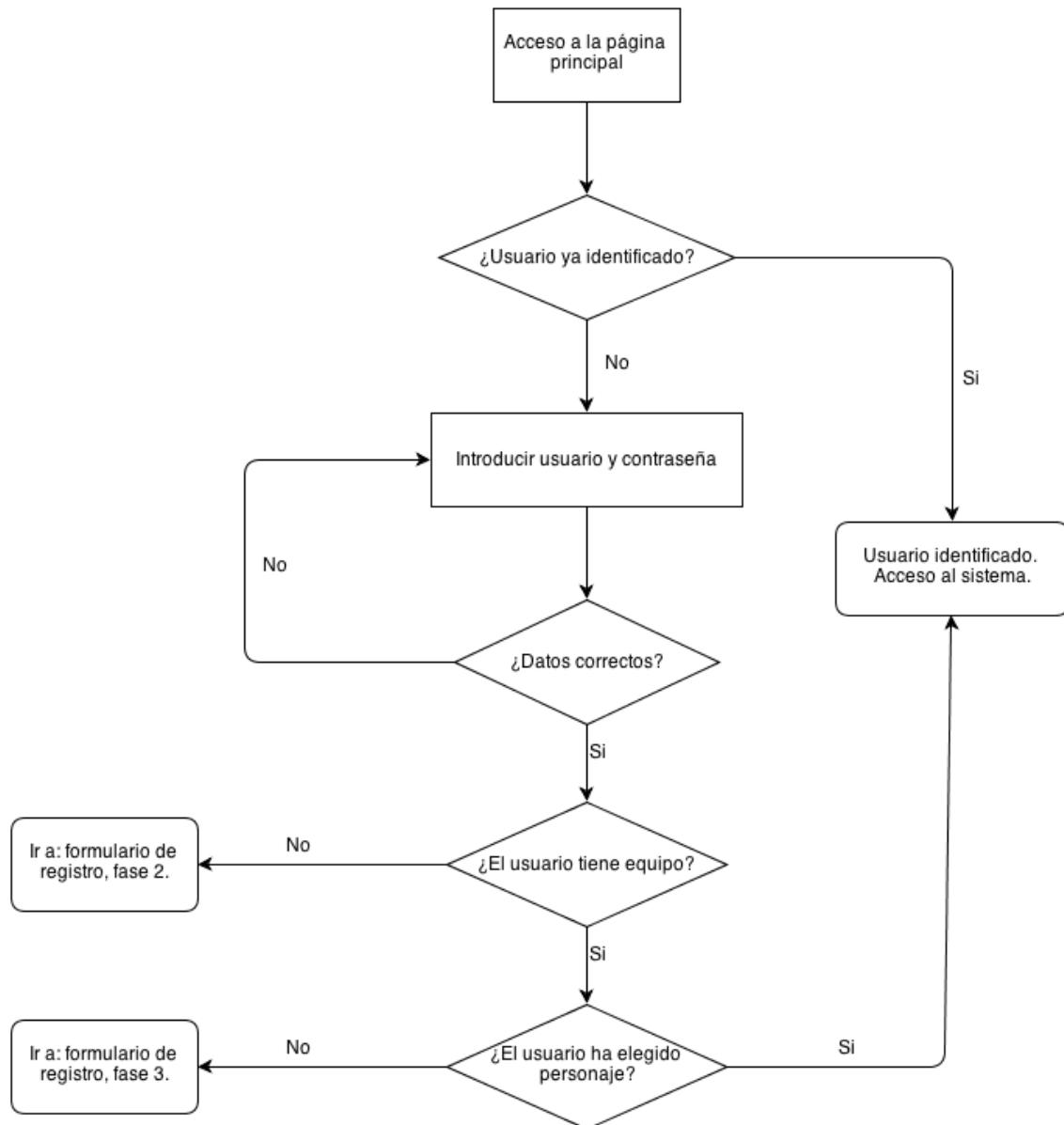


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Login

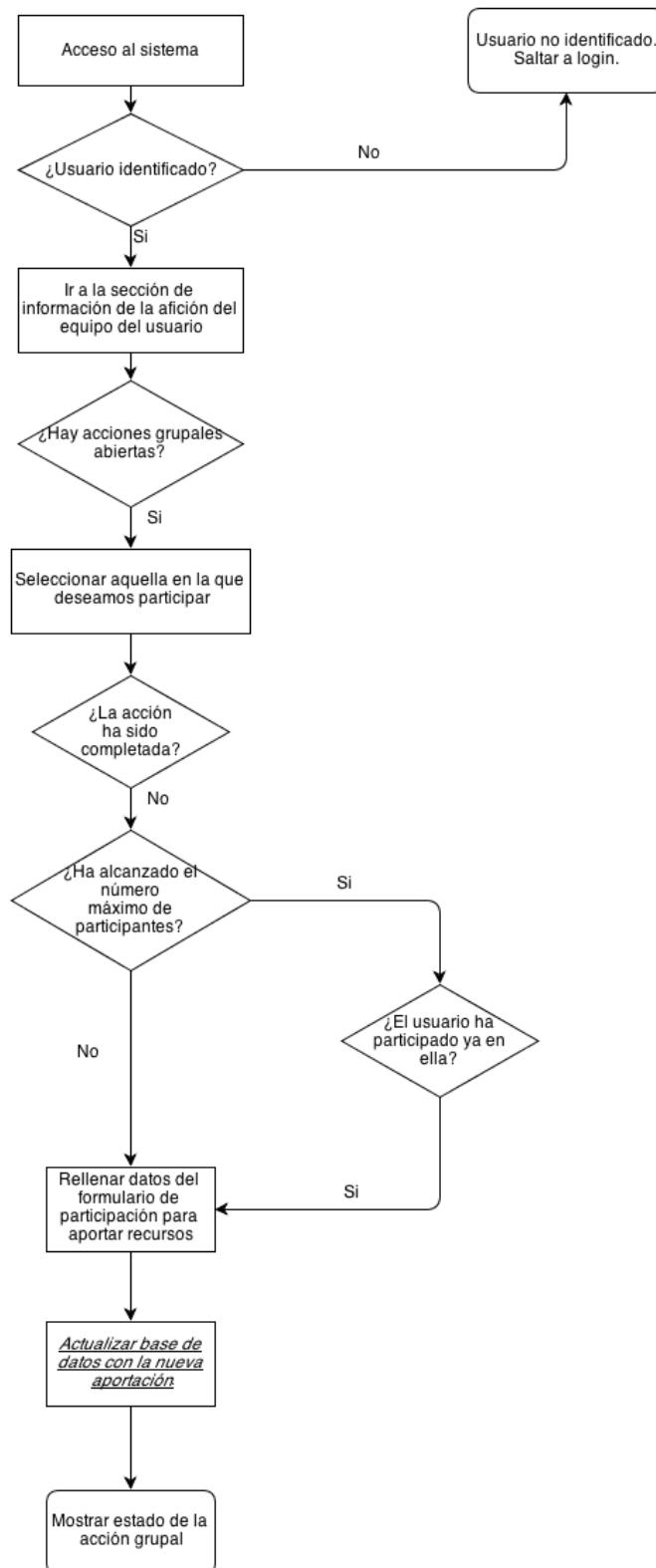


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Participar en acción grupal

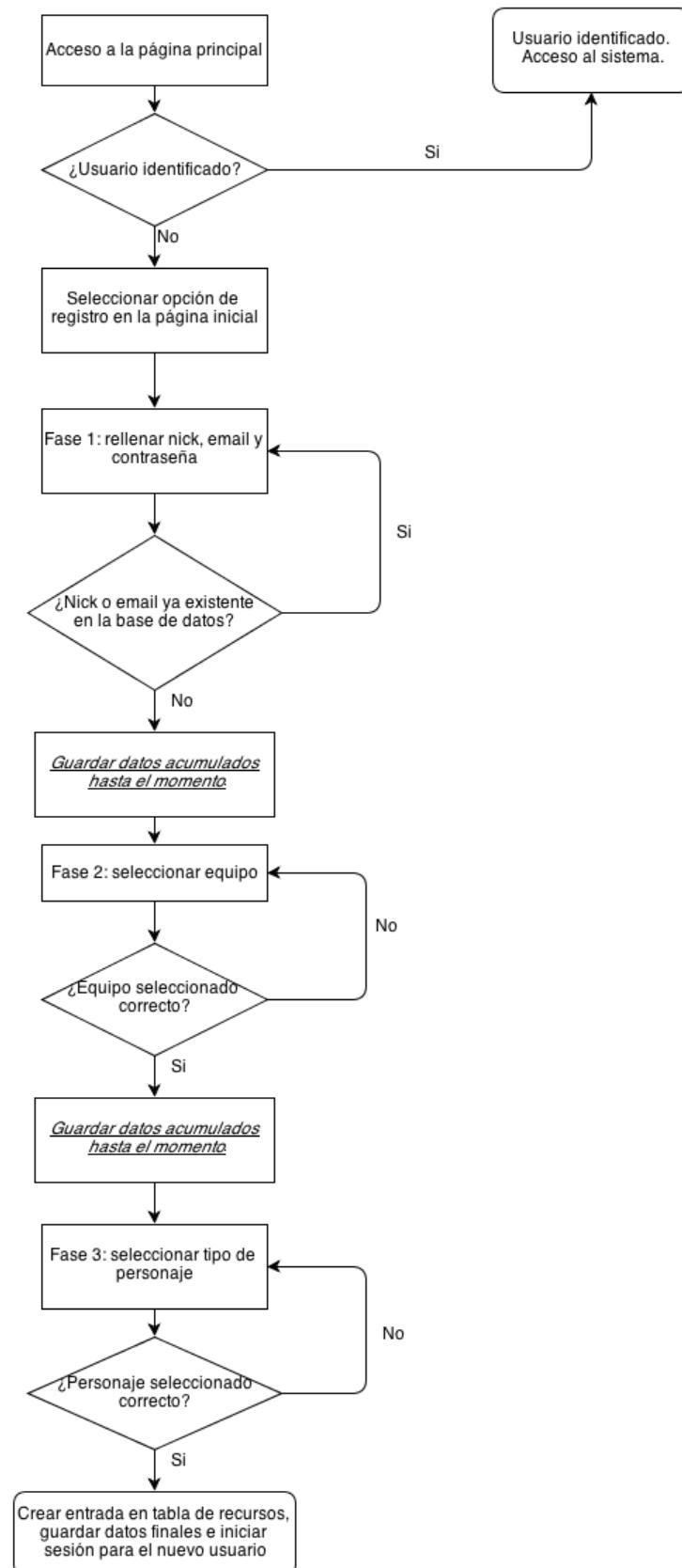


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Registro

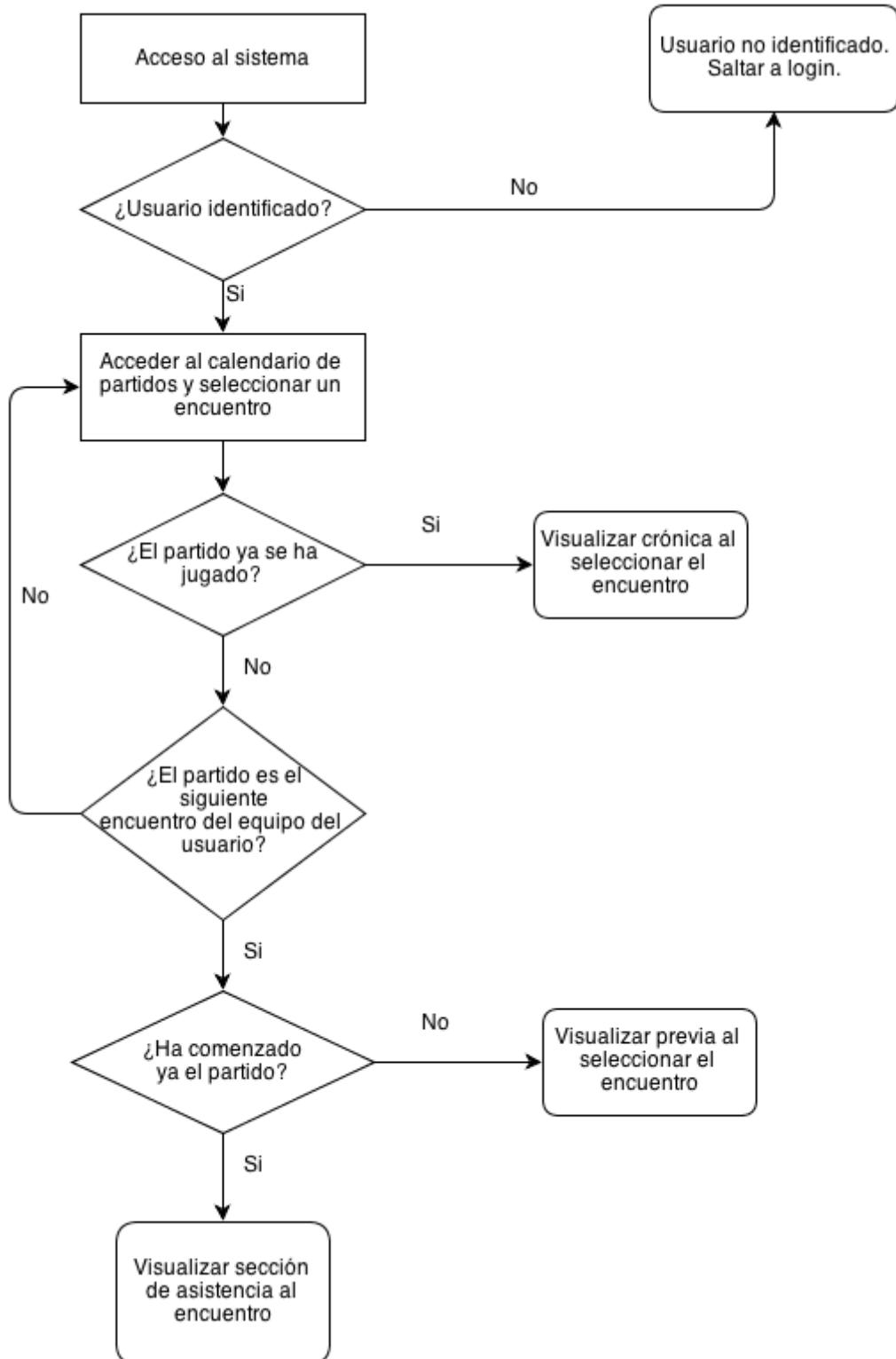


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

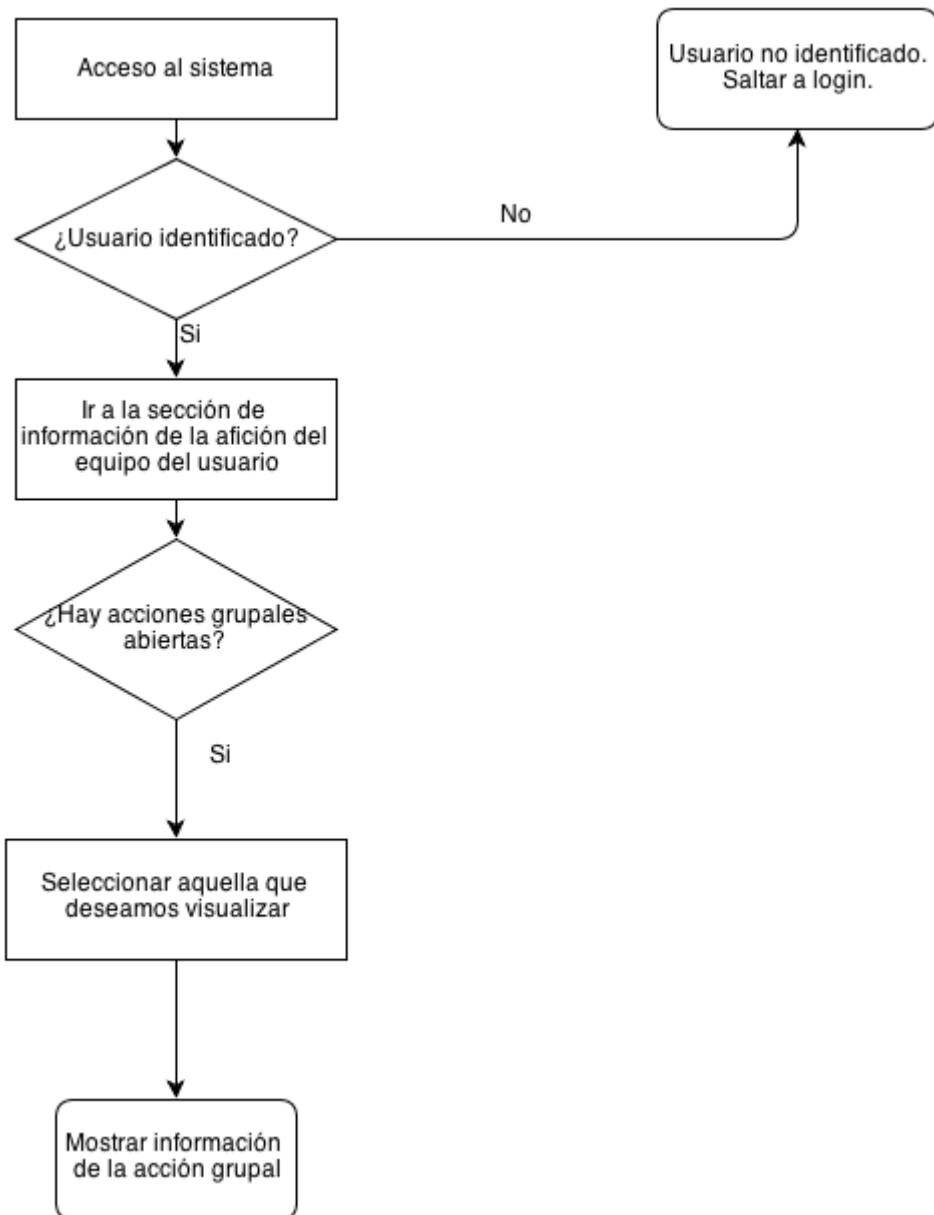
Ingeniería del Software, 2013

## Visualizar crónica/previa/asistencia a un encuentro



# JUGADOR #12

## Visualizar estado de acción grupal

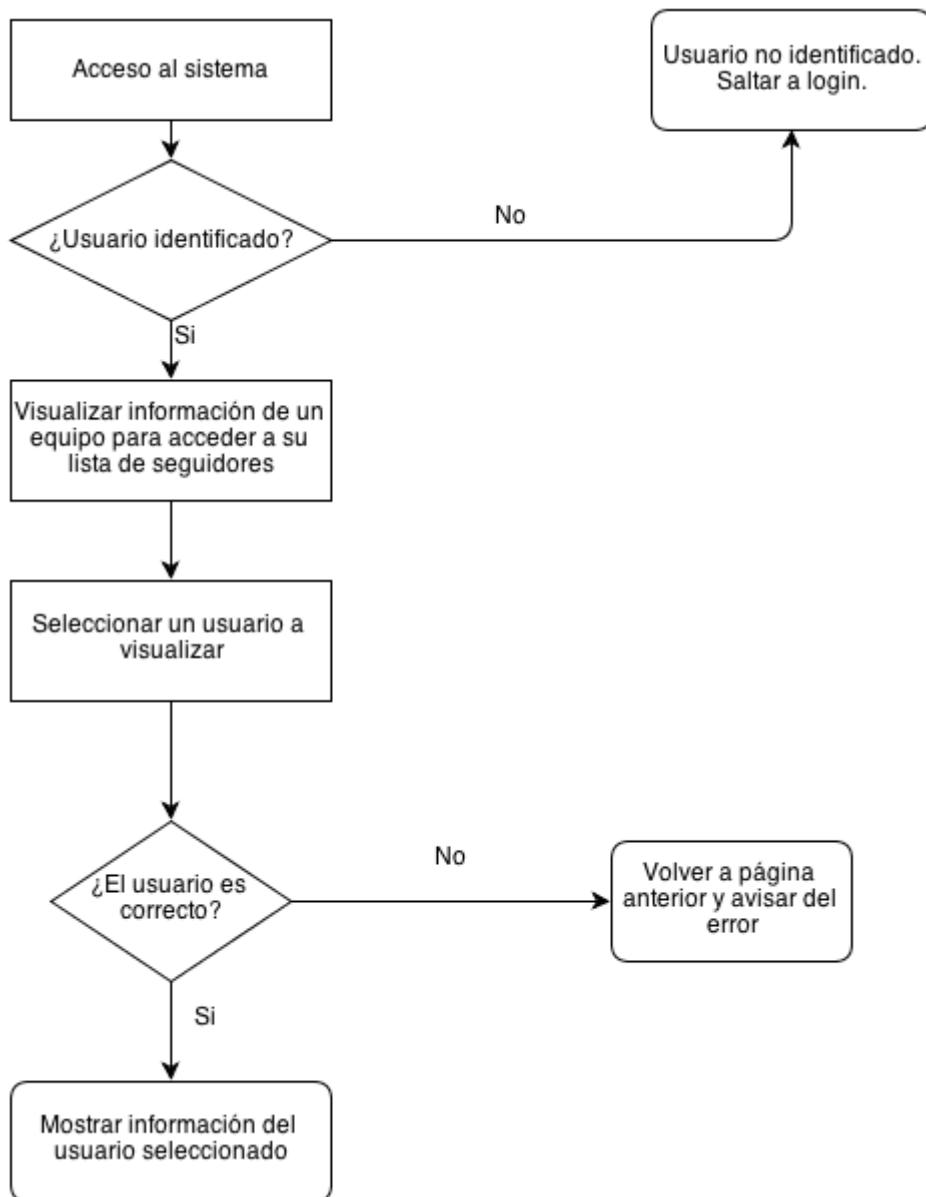


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Visualizar información de otro usuario

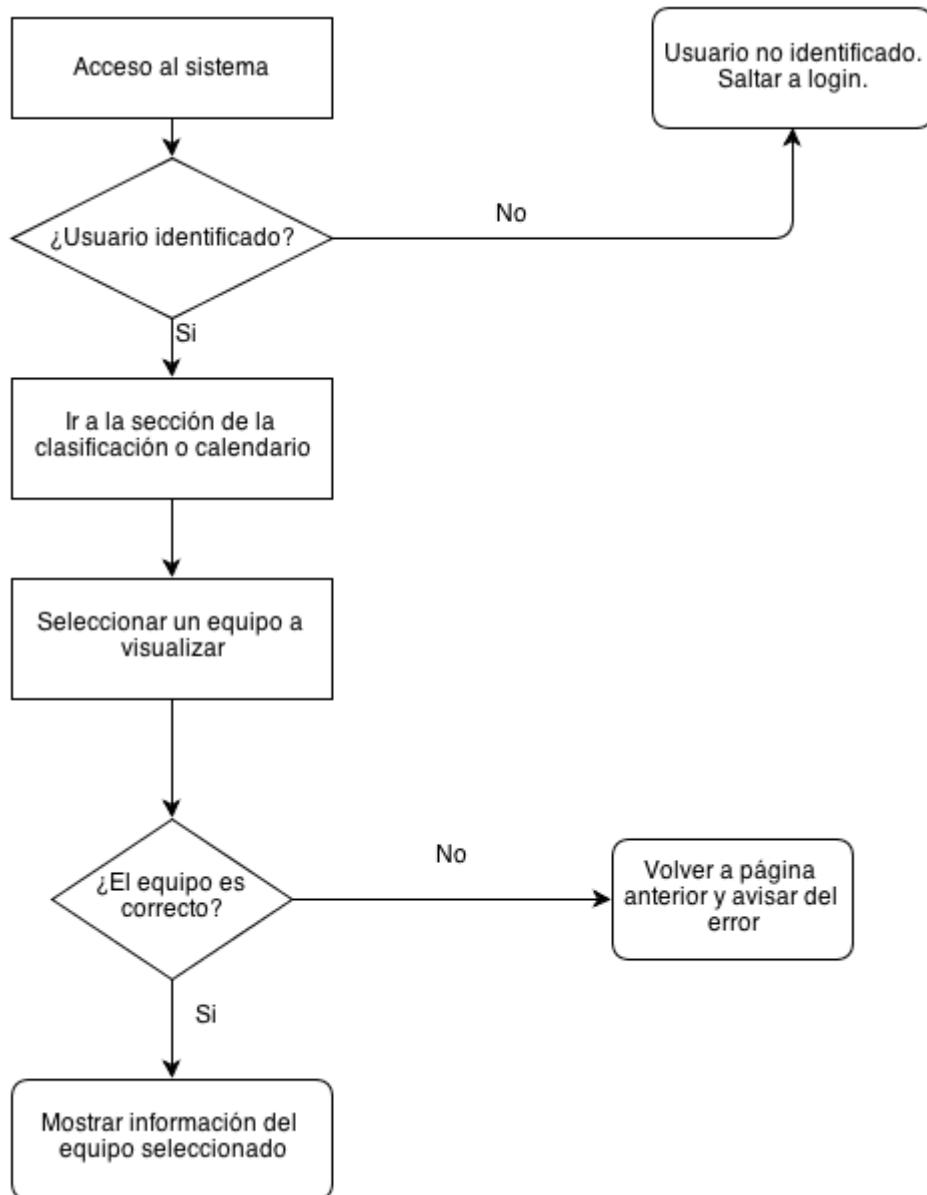


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Visualizar información de un equipo

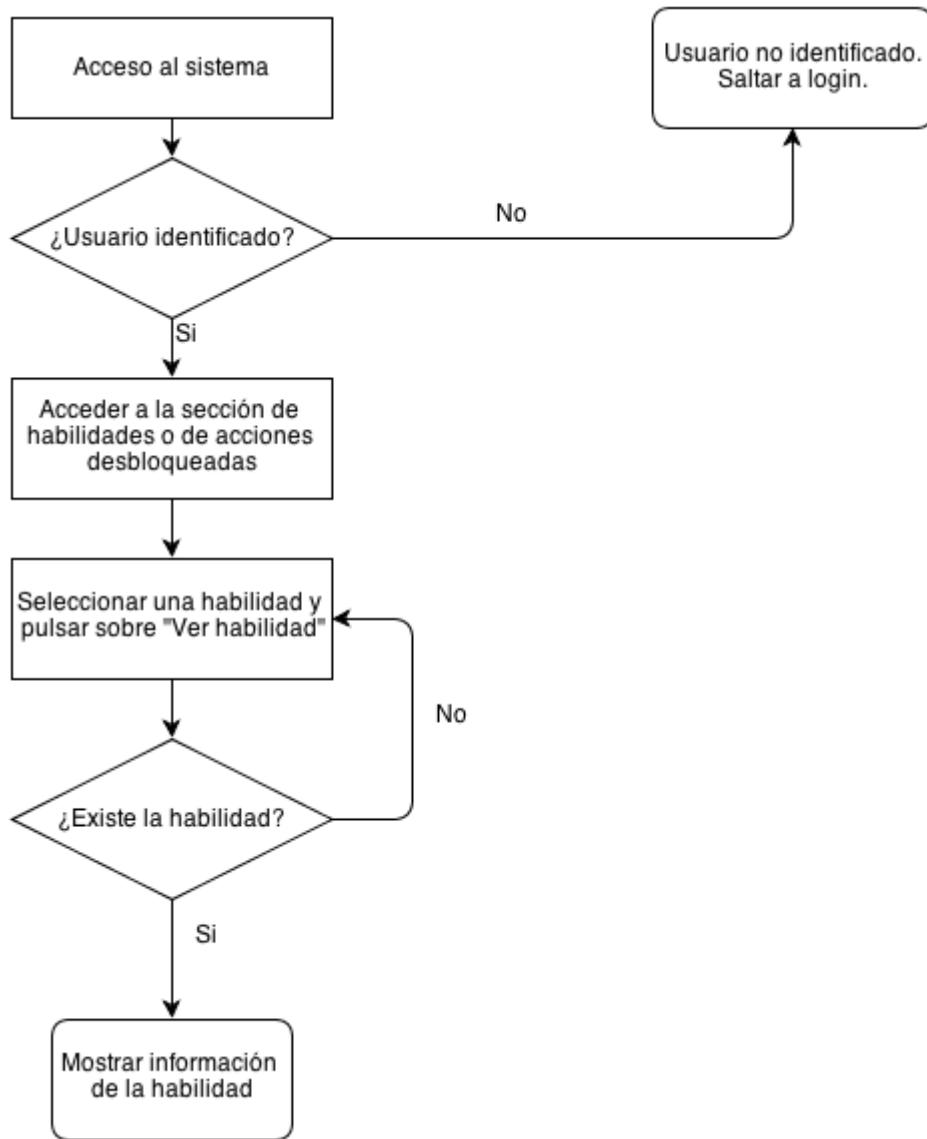


# JUGADOR #12

Documentación técnica: diagramas de flujo de ejecución

Ingeniería del Software, 2013

## Visualizar información de una habilidad



# JUGADOR #12

Documentación técnica: diagramas de clases

Ingeniería del Software, 2013

## Diagramas de clases: índice

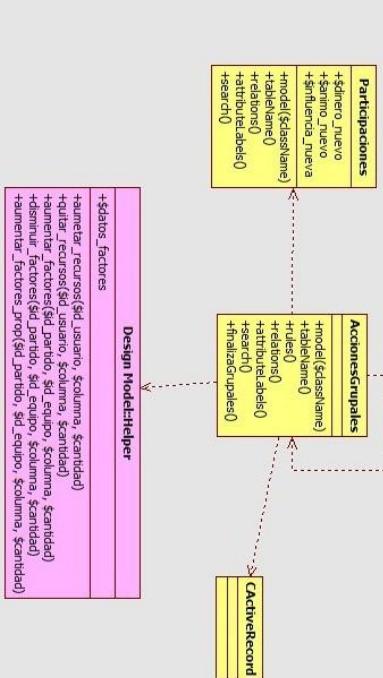
1. Modelos
2. Controladores
3. Acciones
4. Helper
5. Partido
6. Herencia

# JUGADOR #12

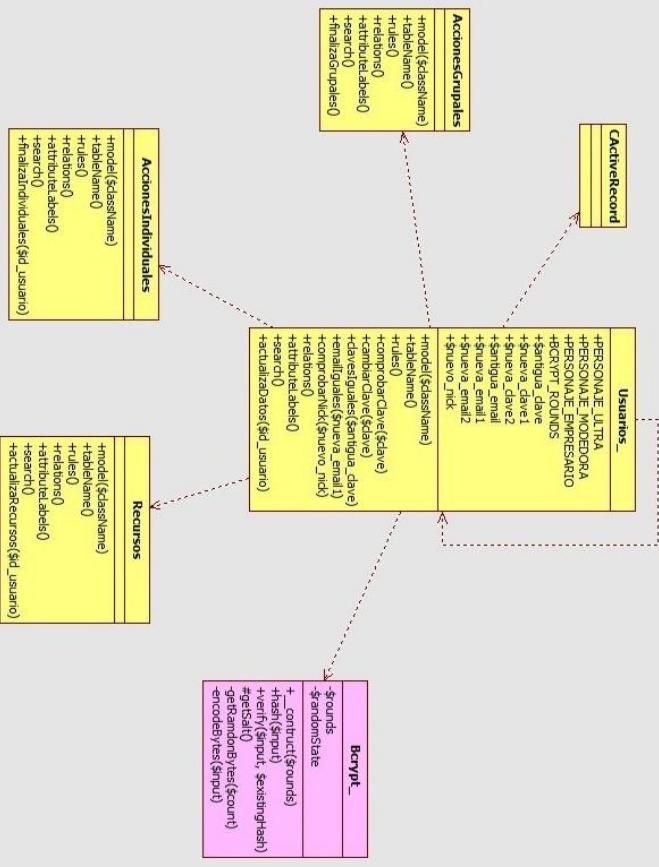
Documentación técnica: diagramas de clases

Ingeniería del Software, 2013

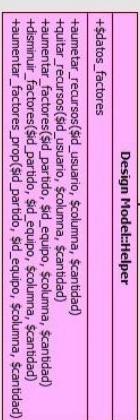
Relaciones del modelo AccionesGrupales con otras clases. Solo hay relaciones de dependencia



Relaciones del modelo Usuarios con otras clases. Solo hay relaciones de dependencia



Relaciones del modelo LoginForm con otras clases.  
Solo hay relaciones de dependencia



Relaciones del modelo Recursos con otras clases.  
Solo hay relaciones de dependencia

Éstos son solo unos ejemplos de cómo se relacionan las clases modelos del MVC con otras clases. Cómo se puede comprobar sólo hay relaciones de dependencia de uso. Suelen relacionarse con otros modelos y con clases auxiliares si lo necesitan.

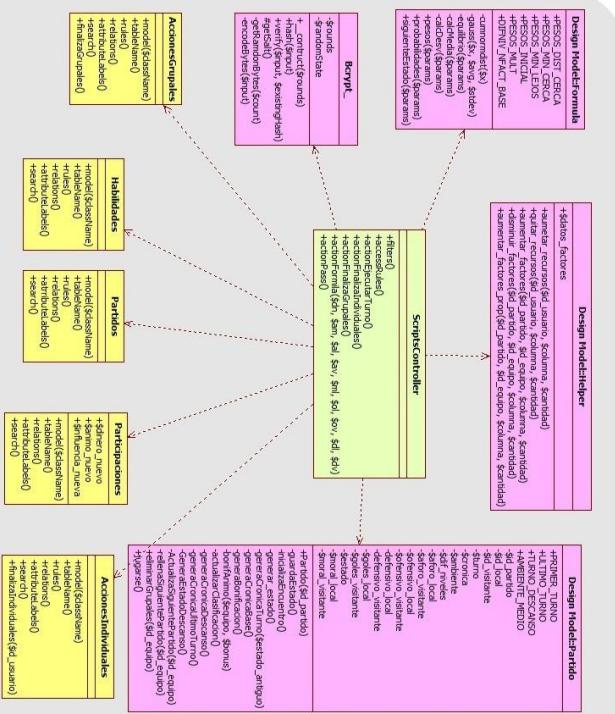
Las relaciones entre clases de las otras clases modelo son muy parecidas a éstas. Esperamos que con estos ejemplos sea suficiente para entender las relaciones de estas clases. Como dato, la clase modelo es la que con más clases se relaciona.

# JUGADOR #12

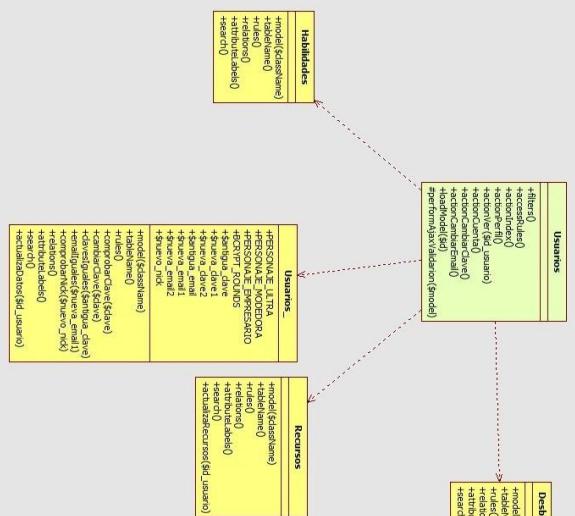
## Documentación técnica: diagramas de clases

Ingeniería del Software, 2013

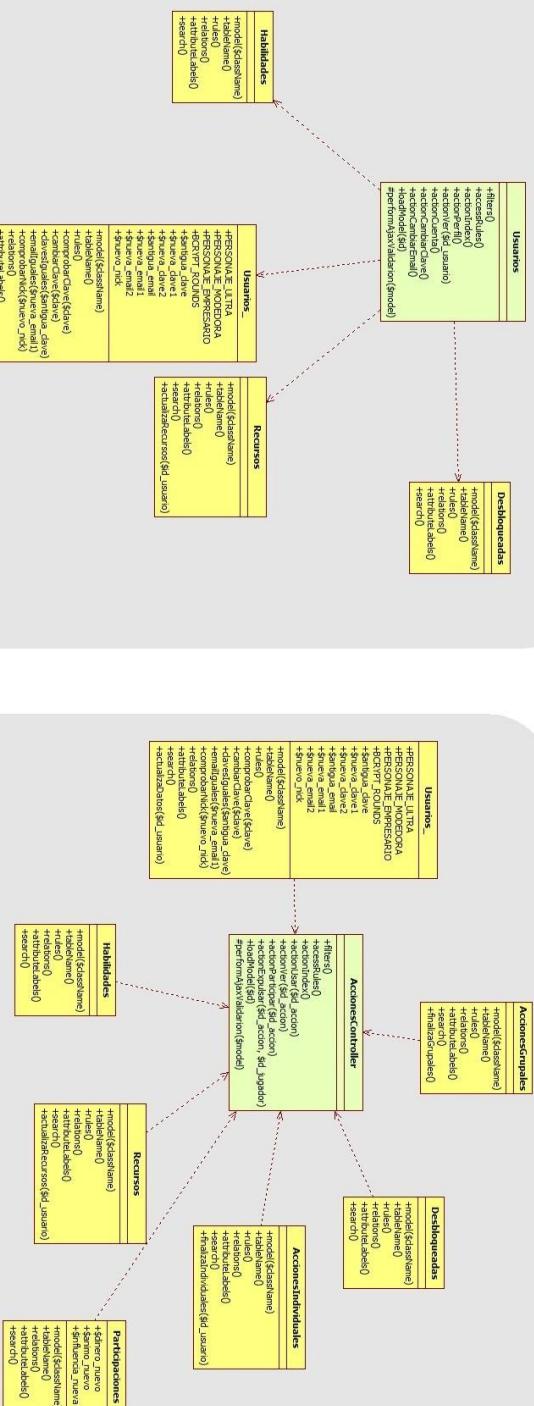
Relaciones del controlador Scripts con otras clases. Solo hay relaciones de dependencia



Relaciones del controlador Usuarios con otras clases. Solo hay relaciones de dependencia



Relaciones del controlador Acciones con otras clases. Solo hay relaciones de dependencia



Éstos son solo unos ejemplos de como se relacionan las clases controlador del MVC. Como se puede ver solo hay relaciones de dependencia.

Las clases de los ejemplos de arriba solo se relacionan con clases modelo. Todas las demás clases controlador, excepto la clase Scripts controller, se relacionan como los ejemplos superiores.

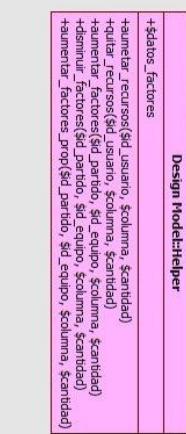
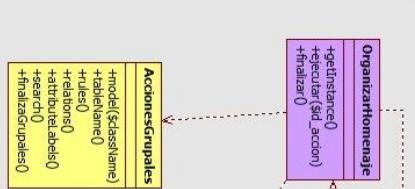
Hemos dibujado la clase ScriptsController porque es la única que no se relaciona sólo con modelos. Es una excepción.

Todas las relaciones son de dependencia de uso.

# JUGADOR #12

Documentación técnica: diagramas de clases

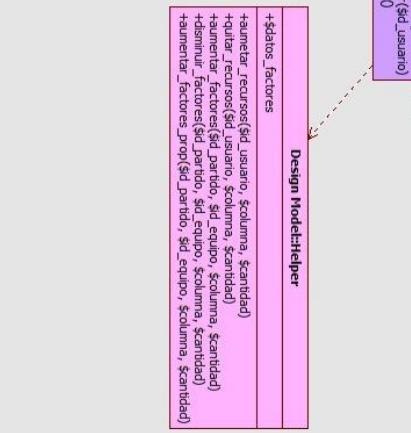
Ingeniería del Software, 2013



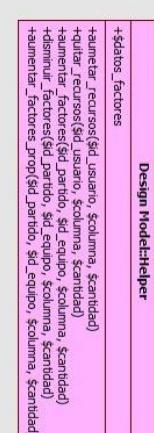
Relaciones de la acción de partido OrganizarHomenaje



Relaciones de la acción de partido IniciarOla



Relaciones de la acción individual ContratarRPP

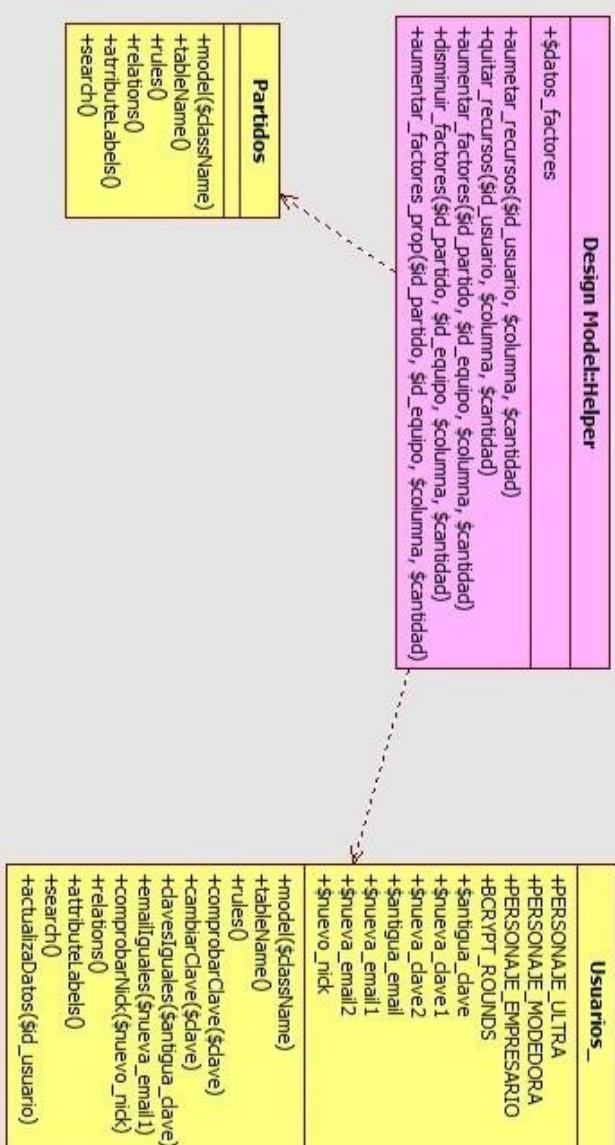


Éstos son ejemplos de cómo se relacionan las clases de las acciones con otras clases. Todas siguen un mismo patrón. Incluidas las acciones pasivas que aquí no están añadidas. Todas las relaciones son de dependencia de uso

# JUGADOR #12

Documentación técnica: diagramas de clases

Ingeniería del Software, 2013

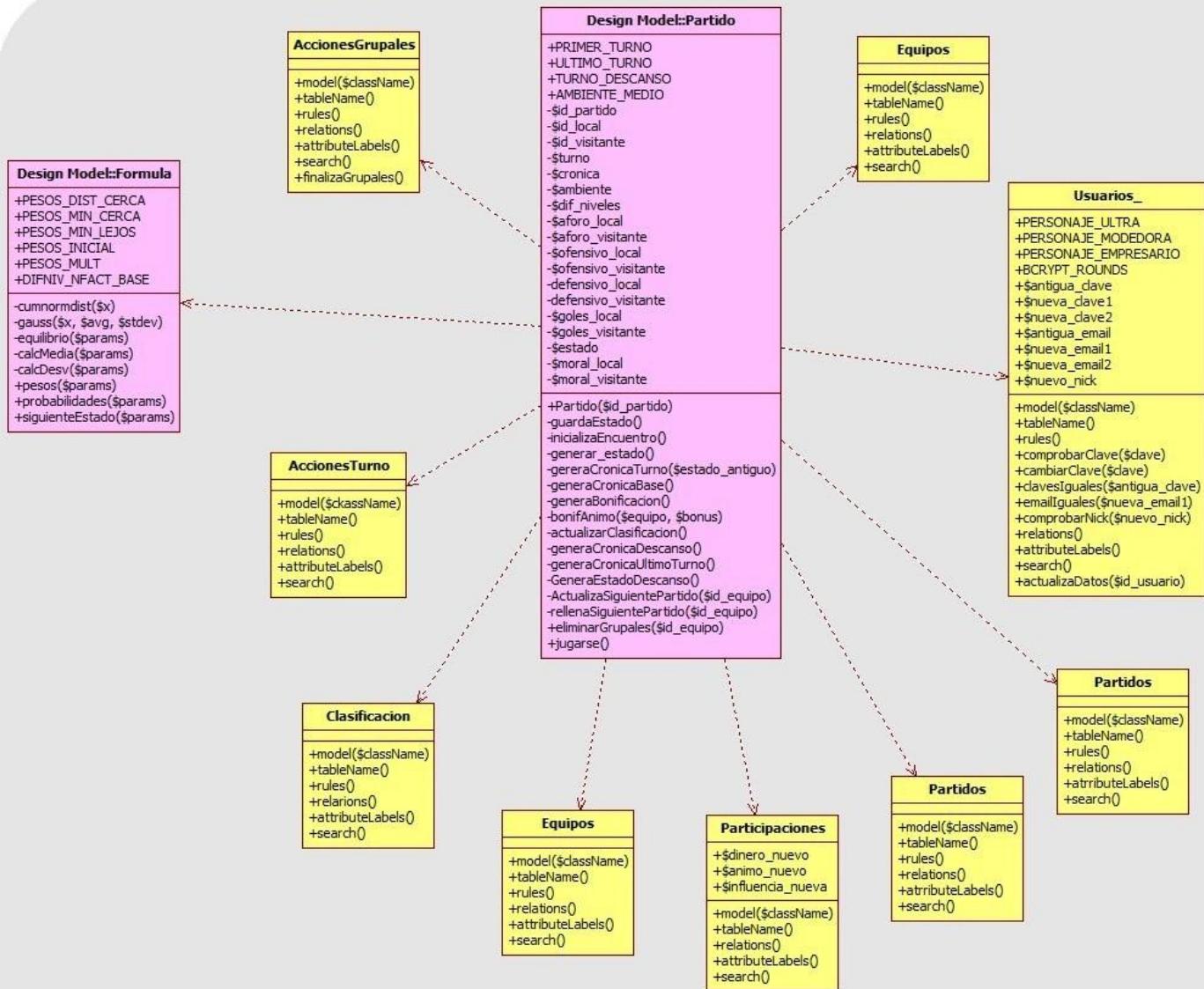


La clase Helper es una clase auxiliar que sirve de ayuda en las transacciones. Especialmente las de gasto y aumento de recursos aunque no son las únicas. Es una clase muy usada así que veamos que es interesante hacerle un diagrama a parte.

# JUGADOR #12

Documentación técnica: diagramas de clases

Ingeniería del Software, 2013

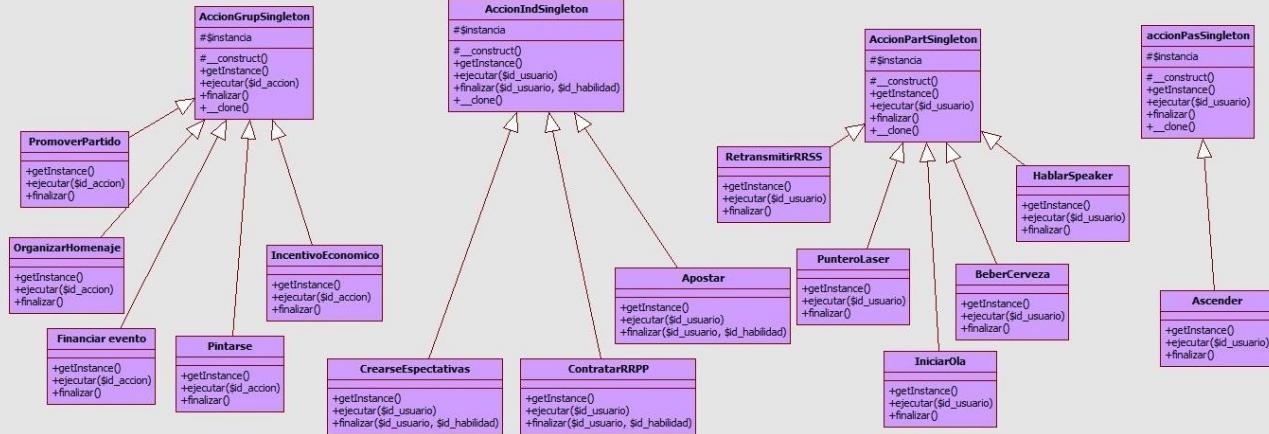


Indica cómo se relaciona la clase Partido con otras clases.  
Solo hay relaciones de dependencia.

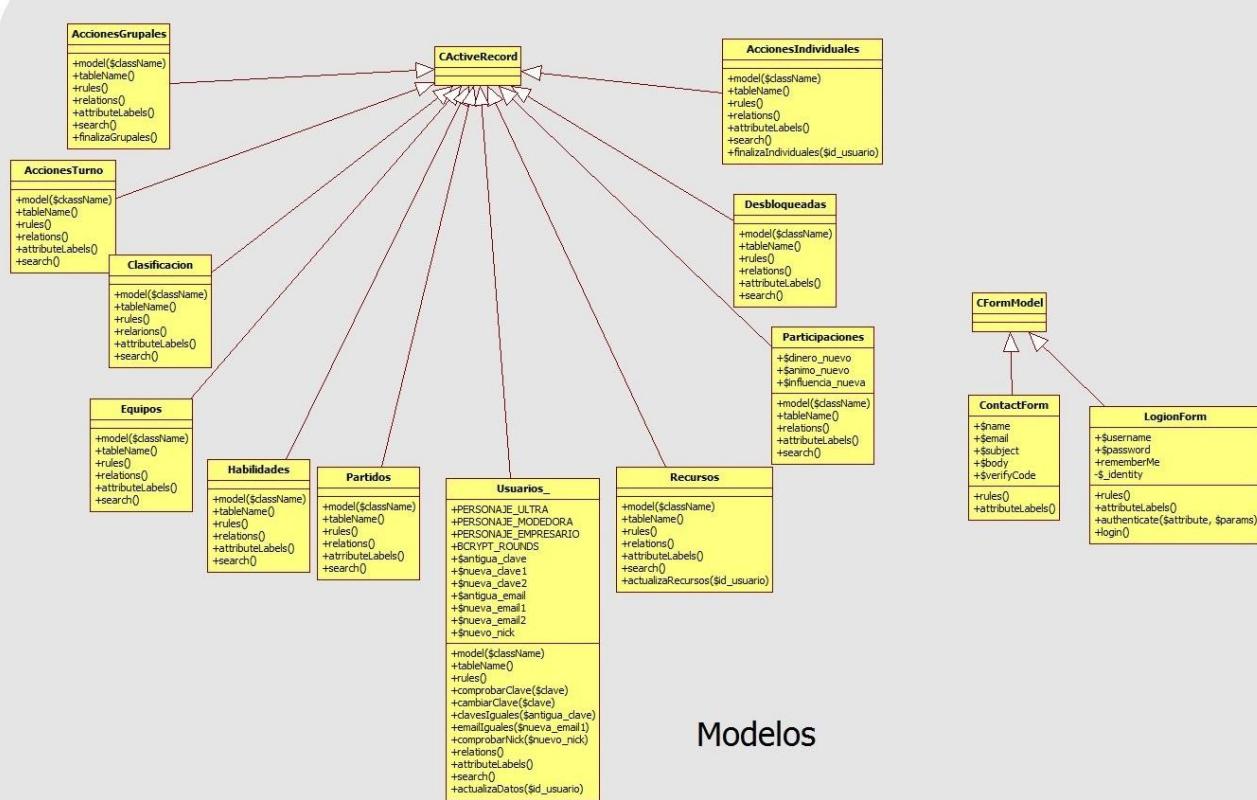
# JUGADOR #12

## Documentación técnica: diagramas de clases

Ingeniería del Software, 2013



## Acciones

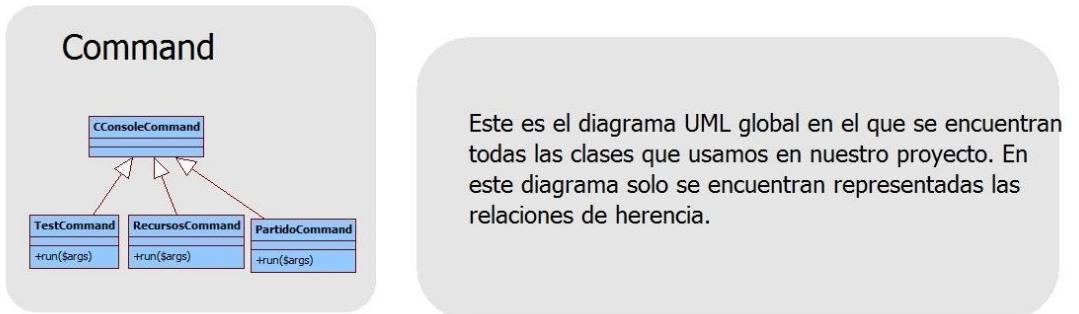
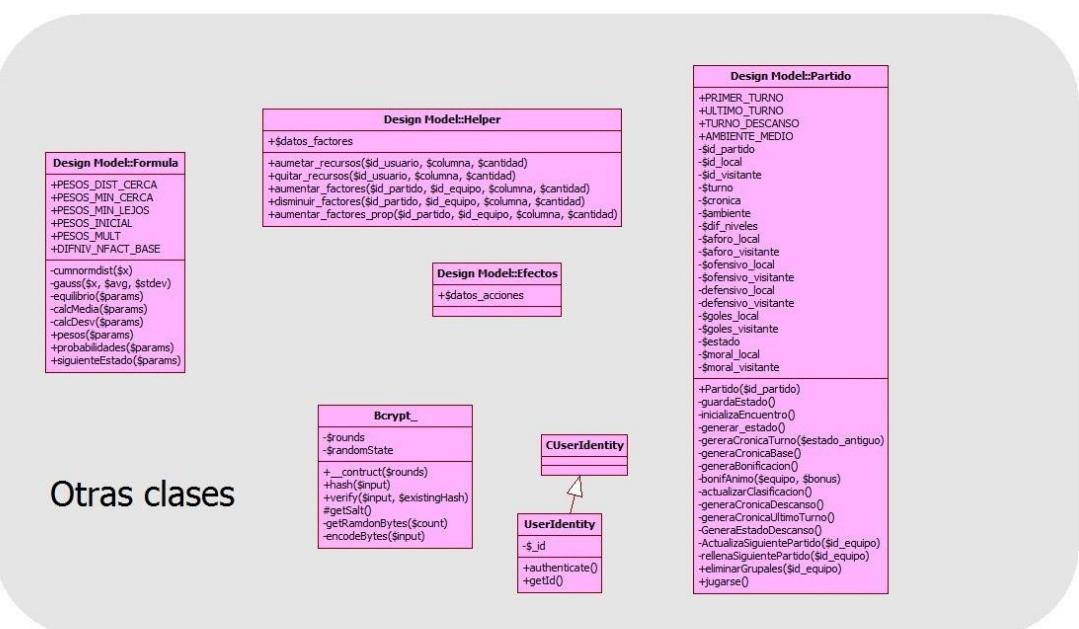
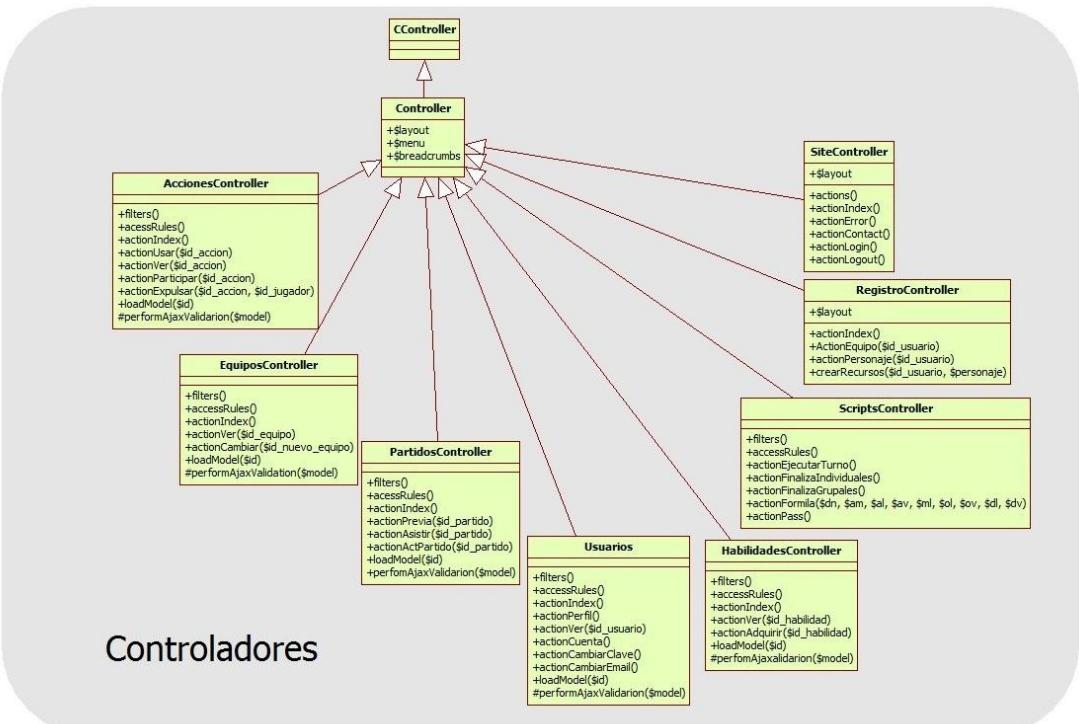


## Modelos

# JUGADOR #12

Documentación técnica: diagramas de clases

Ingeniería del Software, 2013



## Introducción

El siguiente documento es un tutorial con los puntos más destacados del framework de PHP escogido para trabajar- Yii. Está desarrollado por miembros del grupo y cuenta con doble finalidad: la primera para introducir los puntos más importantes del modelo de programación que se seguirá en la implementación de Jugador número 12; y también pretende servir como manual de referencia de las características principales de Yii.

Nota: Todos los esquemas y dibujos utilizados son completamente originales hechos expreso para el documento.

## Índice

1. Funcionamiento del MVC
  - 1.1. CRUD
  - 1.2. Rutas
2. Estructura de directorios de un proyecto Yii
  - 2.1. Archivos de configuración
  - 2.2. Controladores
  - 2.3. Modelos
  - 2.4. Vistas
    - 2.4.1. Cómo funcionan los layouts
  - 2.5. Uniéndolo todo
    - 2.5.1 Renderizar una vista
    - 2.5.2 Cargar un modelo desde un controlador
    - 2.5.3 Guardar en la base de datos
3. Gestión de usuarios
  - 3.1. El módulo de usuarios de Yii
    - 3.1.1 Clase UserIdentity
    - 3.1.2 Clase LoginForm
  - 3.2 Posibles ampliaciones
4. Formularios
  - 4.1 Widget CActiveForm

# JUGADOR #12

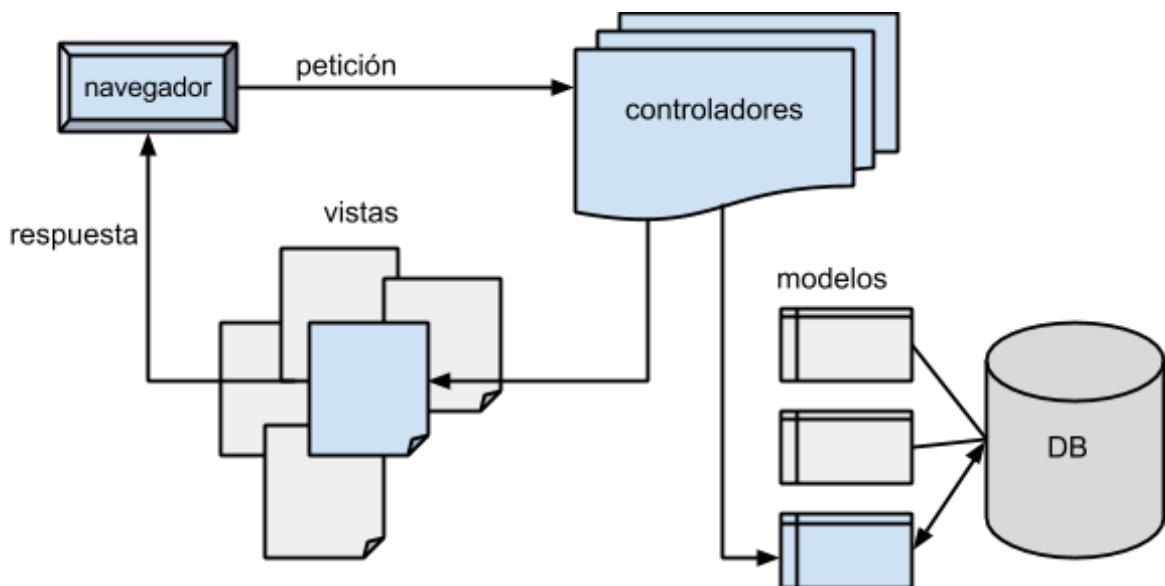
Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013

## 1. Funcionamiento del MVC

Modelo-Vista-Controlador (MVC) es un modelo de desarrollo software que separa los datos de una aplicación, la interfaz de usuario, y la lógica que gestiona los recursos en tres componentes.

1. El **modelo** es el sistema de gestión de la base de datos; se encarga de su lectura y escritura así como de validar los datos.
2. La **vista** es la página HTML que se genera como respuesta a la petición del usuario
3. El **controlador** es el responsable de coordinar los elementos; procesa las peticiones buscando y tratando los datos pedidos y renderiza las vistas para construir la respuesta final.



En una aplicación tendremos 1 controlador, 1 modelo y un conjunto de vistas para cada recurso.

### 1.1. CRUD

Para cualquier recurso existen cuatro funciones o verbos básicos: Create, Read, Update y Delete.

A lo largo de esta guía los ejemplos estarán fijados a <Posts> de un blog ficticio; por lo tanto vamos a hablar de crear nuevos posts; pedir a la aplicación posts para leerlos; modificar un post en concreto o borrar un post existente.

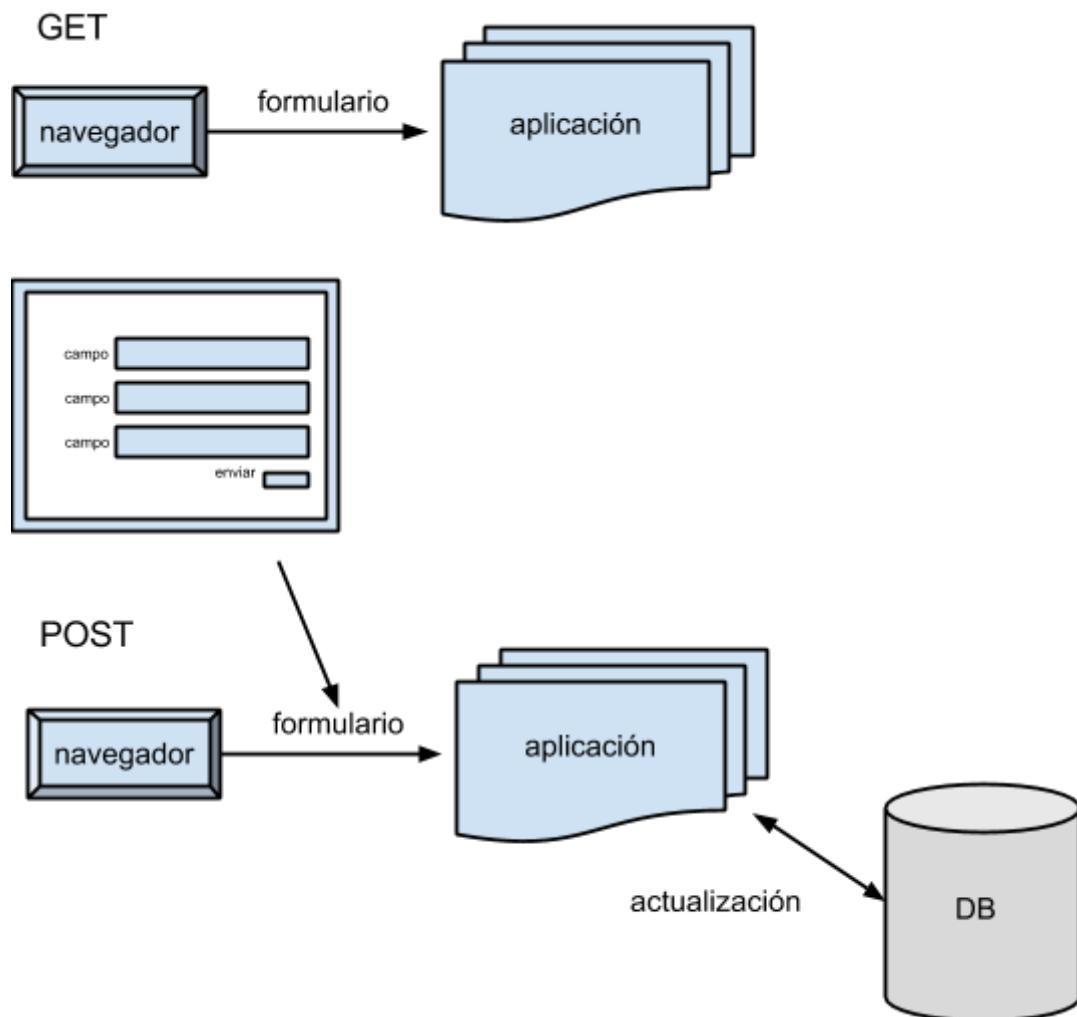
**Create:** Para crear una nueva entrada en el blog, necesitamos llenar primero los campos necesarios (por ejemplo asunto y contenido) de un formulario. Este formulario lo pediremos por GET y una vez relleno lo enviaremos a la aplicación por POST para que se cree y se guarde el nuevo post en la tabla correspondiente de la base de datos.

# JUGADOR #12

Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013

La secuencia de llamadas sería:



**Read:** En este caso sólo hacemos una petición GET de un post en concreto. La aplicación accederá a la base de datos para mostrarnos el dato pedido.

**Update:** La secuencia de llamadas es similar que para crear: primero necesitamos por GET el formulario para modificar un post y después mandaremos nuestra petición por POST

**Delete:** Accederemos (GET) al post concreto y mandaremos la petición de borrar (POST) para eliminarlo de la base de datos.

## 1.2. Rutas

Por defecto nuestras rutas serán:

```
raíz/controlador/acción/primer_parámetro/segundo_parámetro/.../ultimo_p  
arametro
```

Si nuestro controlador para Post define las siguientes acciones:

# JUGADOR #12

Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013

```
public function actionIndex  
public function actionCreate  
public function actionView($id)  
public function actionUpdate($id)  
public function actionDelete($id)
```

Tendremos las siguientes rutas disponibles

```
raíz/posts/index  
raíz/posts/create  
raíz/posts/view/{id}  
raíz/posts/update/{id}  
raíz/posts/delete/{id}
```

## 2. Estructura de directorios de un proyecto Yii

Un proyecto de Yii tiene la siguiente estructura de ficheros y directorios (remarcados los archivos)

Proyecto/

- .htaccess: Archivo de configuración del Xampp
- index.php: <<Main>> del proyecto. Define el framework que se está usando y lanza la aplicación
- assets/
- css/** Aquí se colocarán las hojas de estilo de la aplicación
  - main.css: archivo que importará la jerarquía de archivos .css
- images/ Carpeta para las imágenes estáticas
- protected/** Carpeta principal del proyecto donde escribiremos el código
  - commands/
  - components/
    - Controller.php
    - UserIdentity.php
- config/** Carpeta donde estarán los archivos de configuración
  - main.php
- controllers/** Carpeta para los controladores (un archivo por recurso)
- data/** Carpeta para el esquema de la base de datos
  - schema.mysql.sql
- extensions/
- messages/
- migrations/
- models/** Carpeta para los modelos (un archivo por recurso)
- runtime/
- tests/** Carpeta para los archivos de test
  - fixtures/

# JUGADOR #12

Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013

functional/  
unit/

**views/** Carpeta para las vistas (una subcarpeta por recurso)  
**layouts/** Carpeta con los layouts (plantillas) del proyecto  
**site/** Carpeta para las vistas generales  
**pages/** Vistas .html estáticas

Además, como trabajaremos con el editor sublime, tendremos:

.sublime-workspace: Archivo de configuración de un proyecto en Sublime  
.sublime-project: Archivo donde se registra la ruta a un proyecto de Sublime

## 2.1. Archivos de configuración

El archivo /protected/config/main.php define las variables globales de la aplicación, por ejemplo el nombre de la aplicación está definido en este archivo y podríamos acceder con

```
yii::app()->name.
```

También se carga la configuración para conectar con la base de datos con el array <db> y se define cómo recoger los errores (excepciones)

```
'errorHandler'=>array(  
    //use 'site/error' action to display errors  
    'errorAction'=>'site/error',  
,
```

## 2.2. Controladores

Los controladores se situarán todos en la carpeta protected/controllers y usarán todos el mismo modelo de nombrado, que es: **NombreControladorController.php** (importante que empiecen por mayúsculas, por ejemplo PostsController).

Internamente, dichos archivos contendrán una clase llamada igual que el archivo (PostsController) y que hereda de “Controller”, clase que se encuentra en protected/components/. Esta clase contendrá diversas funciones (acciones) que recibirán datos, los procesarán (con ayuda de los modelos) y los enviarán a una de las vistas.

Es obligatorio que estas funciones se llamen con el patrón **actionNombreacción([parámetros])**. Por ejemplo, la página de index del controlador de los posts, que sería: “localhost/blog/posts/index”, haría referencia al controlador PostsController y a la función actionIndex.

A continuación queda un breve ejemplo de controlador:

```
class PostsController extends Controller
```

## JUGADOR #12

Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013

```
{  
    public function actionIndex()  
    {  
        $dataProvider=new CActiveDataProvider('Posts');  
        $this->render('index',array(  
            'dataProvider'=>$dataProvider,  
        ));  
    }  
}
```

En protected/controllers nos encontramos con un controlador especial:

SiteController.php

Esta clase contiene funciones de uso general (sin ser de páginas específicas) como puede ser el login, logout, el formulario de contacto o la gestión de errores.

Debemos mencionar también diversas funciones de gran utilidad respecto a los controladores como por ejemplo:

- filters()  
Permite especificar una serie de filtros sobre dicho controlador. Se compone de un array de filtros, de los cuales resaltaremos accessControl que sirve para controlar el acceso a un controlador por parte de los usuarios.
- accessRules()  
Esta función será referida por el filtro accessControl y contiene una serie de reglas (allow, deny, etc) en forma de arrays que se evalúan en orden acerca de los permisos de usuarios.
  - Un usuario cualquiera se representará por el símbolo <<\*>>;
  - Un usuario autenticado por <<@>>;
  - Un usuario visitante por <<?>>;

A continuación vemos un ejemplo del uso de esta función en el que permitimos a todos los usuarios acceder al índice y vista de un post; y a un usuario registrado a hacer cualquier acción sobre los posts.

Nota: es importante dejar la última parte del “deny” por si se nos olvidase cualquier permiso.

```
public function accessRules()  
{  
    return array(  
        array('allow', //allow all users to access 'index' and  
        'view' actions.  
            'actions'=>array('index','view'),  
            'users'=>array('*'),  
        ),  
        array('allow', //allow authenticated users to access all  
        )  
    );  
}
```

```
actions
    'users'=>array('@'),
),
array('deny', //deny all users
    'users'=>array('*'),
),
);
}
```

### 2.3. Modelos

Los modelos proporcionan, como hemos visto antes, la parte de interacción con la base de datos. Al igual que los controladores tendrán un directorio y regla de nombrado fijos.

En este caso, los modelos irán situados en la carpeta protected/models y recibirán el mismo nombre que el de la tabla a la que se refieren empezando por mayúsculas. Por ejemplo, para crear el modelo de la tabla “posts”, iremos a protected/models y crearemos el archivo Posts.php.

Internamente, dichos modelos contienen una clase que hereda de CActiveRecord. Lo primero que haremos en dichas clases es llamar a la constructora del padre con:

```
public static function model($className = __CLASS__)
{
    return parent::model($className);
}
```

Tras esto nos encontraremos con diversas funciones, entre las que mencionaremos las siguientes:

- rules()  
Nos proporcionará una serie de reglas para comprobar los diversos campos de la tabla antes de, por ejemplo, modificarla.
- relations()  
Permite crear relaciones (has\_many, many\_many, etc) de esta tabla con el resto. Esto agiliza y simplifica tomar datos relacionados a través de este modelo.
- attributeLabels()  
Permite dar un nombre de forma general a cada columna de la tabla. Por ejemplo, tenemos una columna llamada id\_post y queremos nombrarla en nuestra página de forma general como “Número de post”.

Tenemos que resaltar un tipo de modelo especial que hereda de la clase CFormModel llamado LoginForm.php y que se usa para el inicio de sesión de usuarios. Dicho modelo se encuentra en la carpeta protected/models y lo explicaremos con más detalle en la sección correspondiente.

## 2.4. Vistas

Las vistas se guardarán todas en el directorio protected/views. Para ello, se creará siempre una carpeta con el nombre del controlador asociado en minúsculas. Por ejemplo, para PostsController tendríamos una carpeta en views llamada posts/. Dentro de esta carpeta habrá una serie de archivos .php que contendrán las distintas vistas a renderizar por este controlador. Estas vistas se renderizarán en un layout que viene definido en el controlador.

### 2.4.1. Cómo funcionan los layouts

Los layout son un mecanismo que emplearemos para evitar repetir código de las vistas.

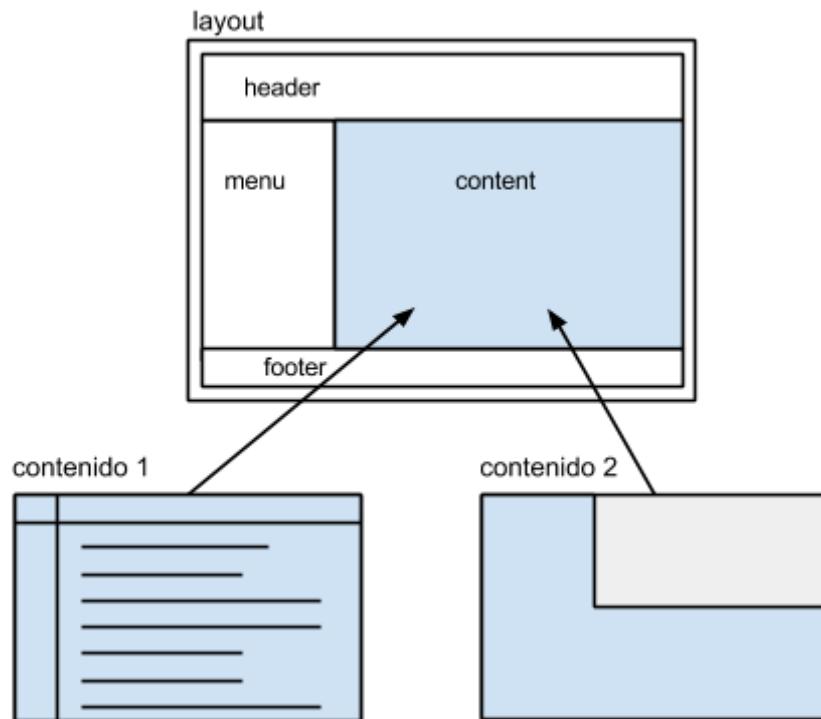
Por norma general tendremos una misma estructura básica para todas las páginas .html de nuestra página; por ejemplo la siguiente:

```
<div id="container">
    <div id="header">
        <!-- contenido para la cabecera -->
    </div>
    <div id="menu">
        <!-- barra de menu izquierdo -->
    </div>
    <div id="content">
        <!-- contenido principal de la página -->
    </div>
    <div id="footer">
        <!-- contenido para el pie de página-->
    </div>
</div>
```

## JUGADOR #12

Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013



Ahora bien, el contenido principal variará dependiendo de lo que queremos mostrar en cada página. Sin usar un layout, tendríamos que repetir el mismo código .html para cada vista. Pero por medio de este mecanismo podemos “montar” el contenido correspondiente sobre el esqueleto principal, cargando el código .html correspondiente dentro del div content.

¿Y como hacemos esto?

- Primero, decir que los layout se encuentran todos en el directorio protected/views/layouts.
- Definir el layout que emplearemos. Si entramos en la clase Controller.php veremos que se declara una variable \$layout que contendrá la ruta de dicho layout.  
Por defecto todos los controladores tomarán ese valor, pero puede ser redefinido en cualquiera de forma independiente.
- El código del layout del blog column1.php:

```
<?php /* @var $this Controller */ ?>
<?php $this->beginContent('//layouts/main'); ?>
<div id="content">
    <?php echo $content; ?>
</div><!-- content -->
<?php $this->endContent(); ?>
```

Fijándonos solo en la sección del “div”, veremos que tan sólo se hace un echo de la variable \$content. Esta variable hace referencia siempre al contenido del nivel inferior. Por ejemplo, desde PostsController llamamos a renderizar la vista index. Lo primero que se hará es tomar dicha vista y enviarla al archivo column1.php. Este archivo tomará la vista index y la añadirá por completo en la variable \$content.

## 2.5. Uniéndolo todo

Daremos unos breves ejemplos sobre cómo interconectar todo esto:

### 2.5.1 Renderizar una vista

Si desde un controlador queremos renderizar una vista, debemos hacer uso de la función

```
$this->render(ruta,array_de_parametros);
```

Por ejemplo, para renderizar la vista create pasándole un modelo guardado con anterioridad, haremos algo como:

```
$this->render('create',array(  
    'model'=>$model ));
```

### 2.5.2 Cargar un modelo desde un controlador

Para cargar un modelo concreto desde un controlador:

```
$model = Posts::model();
```

Por ejemplo, para tomar los datos de una tabla buscando una PK concreta usaremos:

```
$model = Posts::model()->findByPk($id);
```

### 2.5.3 Guardar en la base de datos

Una vez procesados los datos de un modelo desde el controlador, para guardarlos en la base de datos emplearemos la función save() suponiendo el modelo guardado en la variable \$model

```
$model->save();
```

Es importante comprobar siempre que se ha realizado correctamente viendo el valor devuelto por la función.

Si quisiésemos tomar los datos de una tabla completa, podríamos emplear clases de Yii como CActiveDataProvider. Por ejemplo, para obtener los datos de la tabla asociada a Posts, haremos algo como:

```
$dataProvider = new CActiveDataProvider('Posts');
```

## 3. Gestión de usuarios

### 3.1. El módulo de usuarios de Yii

Por defecto Yii maneja un módulo de autenticación muy básico que puede ser ampliado posteriormente. Para ver la parte básica de su funcionamiento nos centraremos en dos clases:

#### 3.1.1 Clase UserIdentity

Esta clase se encuentra en el directorio protected/components y representa la información necesaria para representar un usuario. En primer lugar, debemos crear un atributo \$\_id y sobreescibir la función getId() para que podamos determinar únicamente de alguna forma un usuario frente a otro. Para ello, añadiríamos a la clase:

```
private $_id;
public function getId()
{
    return $this->_id;
}
```

El siguiente punto importante recae sobre la función authenticate() que se encargará de validar si los datos de un usuario son válidos antes de iniciar sesión. En nuestro caso comprobaremos en la base de datos si el nombre de usuario existe y si la contraseña es correcta, quedando el código:

```
public function authenticate()
{
    $user = Usuarios::model()->findByAttributes(array('nombre'=>$this->username));
    if ($user === null)
    {
        $this->_id = 'user Null';
        $this->errorCode = self::ERROR_USERNAME_INVALID;
    }
    else if ($user->password !== $this->password)
    {
        $this-
```

# JUGADOR #12

Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013

```
>errorCode=self::ERROR_PASSWORD_INVALID;
    }
    else { // Okay!
        $this->_id = $this->username;
        $this->setState('user',$user->nombre);
        $this->errorCode = self::ERROR_NONE;
    }
    return !$this->errorCode;
}
```

Como podemos observar, primero creamos una consulta al modelo Usuarios para que nos dé las entradas que tengan como “nombre” el nombre de usuario (que ya veremos de dónde viene). A continuación comprobamos que el usuario es correcto y, en caso contrario, devolvemos los correspondientes errores.

Si las contraseñas coinciden, pasamos a autenticar a nuestro usuario en el sistema. Para ello, primero le asociamos un id a través de la variable \$\_id creada anteriormente. En este caso, suponemos que “nombre” es una clave primaria en la tabla Usuarios y no está repetida, con lo que nos aseguramos que el id será único para cada usuario.

Acto seguido, pasamos a registrar una variable de sesión - a través de la función setState - para dicho usuario (array \$\_SESSION en php) denominada ‘user’ y que tomará como valor el nombre de usuario obtenido del modelo.

## 3.1.2 Clase LoginForm

Esta clase representa la estructura para obtener los datos del formulario de login. Una vez obtenidos los datos, hace uso de dos funciones principales para realizar la autenticación:

- o authenticate()

```
public function authenticate($attribute,$params)
{
    if(!$this->hasErrors())
    {
        $this->_identity=new UserIdentity($this-
>username,$this->password);
        if(!$this->_identity->authenticate())
            $this->addError('password','Usuario o
contraseña incorrecto.');
    }
}
```

## JUGADOR #12

Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013

Esta función se encarga de comprobar, a través de la clase UserIdentity, si el usuario y contraseña introducidos son correctos. Se basa en la siguiente función.

- o rules()

```
public function rules()
{
    return array(
        // username and password are required
        array('username, password', 'required'),
        // rememberMe needs to be a boolean
        array('rememberMe', 'boolean'),
        // password needs to be authenticated
        array('password', 'authenticate'),
    );
}
```

Este fragmento se encarga de declarar las reglas acerca de los campos de login. En este caso, vemos que usuario y contraseña son requeridos, el botón de “Recordarme” debe ser booleano y la contraseña además debe ser autenticada.

- o login()

```
public function login()
{
    if($this->_identity==null)
    {
        $this->_identity=new UserIdentity($this-
>username,$this->password);
        $this->_identity->authenticate();
    }
    if($this->_identity-
>errorCode==UserIdentity::ERROR_NONE)
    {
        $duration = $this->rememberMe ? 3600*24*30 : 0; // 
30 days
        Yii::app()->user->login($this->_identity,
$duration);
        return true;
    }
    else
        return false;
}
```

## JUGADOR #12

Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013

Por último, esta función se encarga de loguear al usuario en el sistema haciendo uso de `Yii::app()->user->login($this->_identity,$duration);`. Es importante ver que devuelve un booleano indicando si se ha realizado la operación con éxito, por lo que deberemos capturarlo y actuar en consecuencia.

Una vez vistas estas dos clases gracias a las cuales podemos tener un usuario ya autenticado en el sistema, pasemos a ver un par de detalles útiles:

- Registrar variable de sesión: para ello, como hemos visto antes, hemos hecho uso de la función `setState()`:

```
$this->setState('nombreVariable',valor);
```

- Acceder a variable de sesión: para esto nos serviremos del módulo de usuarios de Yii, al que referenciaremos de la siguiente forma:

```
Yii::app()->user->nombreVariable
```

### 3.2 Posibles ampliaciones

Yii nos ofrece diversas extensiones al módulo de usuarios muy útiles. Entre ellas podemos destacar dos bastante interesantes:

- CDbHttpSession: nos ofrece una forma diferente y más cercana a PHP de trabajo con sesiones, incluyendo la opción (automática) de mantenerlas en una base de datos. Otras de las opciones que aporta son la generación de correos de verificación, recuperación de contraseñas, etc.
- CDbAuthManager: una interesante ampliación al módulo de usuarios básico que añade la posibilidad de creación de un sistema de roles, tareas y operaciones. Para ello hace uso de una base de datos creada automáticamente en la que guarda el árbol de relación entre estas entidades. Ofrece también la posibilidad de crear “reglas de negocio” que servirán para aplicar los permisos bajo parámetros adicionales. Esto muy útil y sencillo de utilizar para detalles tan importantes como permitir que un usuario tan sólo pueda modificar sus propios datos.

## 4. Formularios

A la hora de trabajar con formularios se nos presentan dos opciones posibles:

- Podemos realizar los formularios al estilo tradicional, esto es, crear el formulario Html a mano, definir su archivo de procesado por PHP y su validación por javascript/ajax+PHP.

Esto nos permite enviar parámetros por GET o POST al controlador correspondiente y tomarlos allí haciendo uso de `$_GET` y `$_POST`.

# JUGADOR #12

Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013

- Otra opción es hacer uso del widget (extensión) CActiveForm, que explicaremos a continuación.

## 4.1 Widget CActiveForm

Esta extensión nos permite trabajar de forma muy sencilla con formularios estándar. De forma predeterminada permite crear un formulario asociado a un modelo concreto. Por ejemplo, pongámonos en el caso de la tabla Posts de la base de datos. Si generamos usando Gii el código para sus vistas, en la parte de “create” y “update” se hará uso de una vista denominada \_form.php que llama a este mismo widget. Este formulario incluye campos para todas las columnas de dicha tabla (Posts), junto con sus reglas de validación.

Para llamar al widget debemos darle una serie de parámetros:

```
$form=$this->beginWidget(' CActiveForm', array(  
    'id'=>'posts-form',  
    'enableAjaxValidation'=>false,  
) );
```

En primer lugar le daremos el nombre del widget ( CActiveForm). Tras esto, dos parámetros necesarios para dicho widget: el “id” que se asociará al formulario html y la opción de habilitar la validación de los datos mediante Ajax.

Lo importante es que no sólo debemos adaptarnos al formulario predeterminado, sino que podemos lo podremos modificar fácilmente. Por ejemplo, imaginemos que necesitamos añadir la fecha a las entradas del blog. No queremos que el usuario pueda introducirla a través del formulario, por lo que eliminamos el textField y la sección correspondiente de la fecha. A continuación, pasamos al controlador, donde añadiremos la capacidad de añadir esos datos internos:

```
public function actionCreate()  
{  
    $model = new Posts();  
  
    // Uncomment the following line if AJAX validation is  
    // needed  
    // $this->performAjaxValidation($model);  
  
    if(isset($_POST['Posts']))  
    {  
        $model->attributes = $_POST['Posts'];  
        $model->fecha = time();  
        $model->usuario = Yii::app()->user->user;  
        if($model->save())  
    }
```

## JUGADOR #12

Documentación referencial: guía del framework Yii

Ingeniería del Software, 2013

```
        $this->redirect(array('view','id'=>$model->id));
    }

    $this->render('create',array(
        'model'=>$model,
    ));
}
```

Como podemos ver, el widget pasa todos los campos del formulario en un array llamado igual que el modelo (Posts en este caso) empleando el método POST. Por ello, para comprobar que estos datos existan y emplearlos haremos uso de `$_POST['Posts']`.

Acto seguido pasamos a añadir los datos del array directamente al modelo. Para ello nos servimos de la propiedad `attributes` del modelo que nos permite hacerlo directamente (toma los valores del array Posts y los añade al modelo).

Ahora, como lo que nos interesaba era modificar la fecha, tan solo debemos crear manualmente dicho campo en nuestro modelo y darle valor. Lo mismo hemos hecho en este caso con el nombre de usuario, que será tomado de la variable de sesión 'user'.

## Instalación y uso de XAMPP

Para aquellos que decidan montar un servidor “todo en uno” sin instalar Apache, MySQL, etc. por separado, dejo este tutorial sobre XAMPP.

### 1. WINDOWS

- Descargar XAMPP del siguiente enlace:  
<http://www.apachefriends.org/download.php?xampp-win32-1.8.1-VC9-installer.exe>
- Ejecutar el instalador y elegir el directorio donde se instalará XAMPP. Por temas de permisos en W7 y tal, siempre lo instalo en C:/xampp para evitar problemas.
- Al abrir XAMPP nos saldrá un panel de control con diversas funciones. Las más importantes serán Apache (servidor local) y MySQL (base de datos). Al darle a “Start” ambos comenzarán a ejecutarse y podremos usarlos. Cuando terminemos, pulsaremos sobre “Stop”.

**Nota:** recomiendo añadir al path las siguientes rutas -> C:\xampp\php y C:\xampp\mysql\bin para poder usar los comandos de php y mysql desde la consola de Windows en caso de ser necesario.

En la carpeta C:\xampp\htdocs estarán los elementos de nuestro servidor local (<http://localhost/>). Para comprobar que se ha instalado todo correctamente, iniciar Apache y acceder a la dirección anterior, en la cual debería mostrarse la página local de XAMPP.

### 2. LINUX

- Descargar la versión de Linux desde:  
<http://www.apachefriends.org/download.php?xampp-linux-1.8.1.tar.gz>
- Ir a un terminal y acceder como administrador (root) (ejecutar “su”).
- Extraer los archivos en /opt ejecutando “tar xvfz xampp-linux-1.8.1.tar.gz -C /opt”.
- XAMPP quedará instalado en el directorio /opt/lampp.
  - Postinstalación:
    - la carpeta htdocs (donde va a ir el proyecto) sólo está disponible para root. Para cambiar los permisos de acceso a la carpeta, añadimos a nuestro usuario local como owner. Para ello hay que ejecutar el comando (como root):  
"chown miUsuarioLocal /opt/lampp/htdocs/"
    - Además es recomendable tener un acceso directo desde /home por si está en una partición diferente, para ello hay que ejecutar (como root)  
"ln -s /opt/lampp/htdocs /home/miUsuarioLocal/ProyectosXampp".
  - Configuración de seguridad:
    - Ejecutar el comando (como root): "/opt/lampp/lampp security"

## JUGADOR #12

Documentación referencial: instalación y uso de XAMPP

Ingeniería del Software, 2013

A continuación veremos que nos hace las siguientes preguntas:

**1 - Your XAMPP pages are NOT secured by a password**

Informa que las páginas de XAMPP no están protegidas por una contraseña, esto quiere decir que se puede acceder a las páginas de configuración de XAMPP. Para proteger esta parte vital de nuestro servidor, conviene decirle que sí (Y) e ingresar la clave dos veces.

**2 - MySQL is accessible via network**

Aquí nos dice que MySQL es accesible por la red, mejor deshabilitar el acceso por red.

**3 - The MySQL/phpmyadmin user pma has no password set!!!**

Aquí nos dice que el usuario por defecto que es pma para phpmyadmin, no tiene una contraseña, responde que sí (Y.) y añade un password.

**4 - MySQL has not root password set!!!**

Igual que en el caso anterior, pero para Mysql, responde que sí e introduce contraseña.

**5 - The FTP password for user 'nobody' is still set to 'lamp'.**

Igual que en el caso anterior, pero para proFTPD, responde que sí e introduce contraseña.

- Para iniciar XAMPP, ejecutar en un terminal “/opt/lampp/lampp start”. Esto iniciará Apache y MySQL.  
Además inicia proFTPD, por lo que no es conveniente usar este comando si queremos prescindir de usar ese FTP. Es mejor usar el comando siguiente (como root):  
“/opt/lampp/share/xampp-control-panel/xampp-control-panel”, que inicia un panel de control con GUI, que permite seleccionar los componentes que quieras iniciar (Apache y MySQL).
- Acceder desde un navegador a <http://localhost> y comprobar que todo funciona correctamente.

### 3. MAC OS X

- Descargar XAMPP en: <http://www.apachefriends.org/download.php?xampp-macosx-1.7.3.dmg>
- Abrir el DMG-Image y arrastrar el directorio XAMPP en el directorio de Aplicaciones.
- Abrir el panel de control de XAMPP.
- Acceder desde un navegador a <http://localhost> y comprobar que todo funciona correctamente.

## Introducción

El siguiente documento es un tutorial o manual hecho por miembros del grupo para contar con un manual de referencia para usar el repositorio elegido (Git).

## Índice

1. ¿Qué es Git?
  - 1.1. Sistema de control de versiones (SCV)
  - 1.2. Sistema centralizado vs. sistema distribuido
  - 1.3. Los tres estados
2. Guía de instalación de Git
  - 2.1. Windows
  - 2.2. Linux
3. Configuración básica inicial
  - 3.1. Identificación
  - 3.2. Clave RSA
    - 3.2.1. Windows
    - 3.3.2. Linux
4. Acciones básicas en Git
  - 4.1. Añadir ficheros al índice (add)
  - 4.2. Borrar ficheros (rm)
  - 4.3. Comprobar el estado de los ficheros (status)
  - 4.4. Guardar los cambios en el repositorio local (commit)
  - 4.5. Deshacer cambios (reset)
  - 4.6. Subir cambios al repositorio remoto (push)
  - 4.7. Bajar cambios del repositorio remoto (pull)
  - 4.8. Trabajo con ramas (checkout)
  - 4.9. Juntar la rama actual con otra (merge)
  - 4.10. Modificar el punto de partida de una rama (rebase)
  - 4.11. Más información
5. Formas de trabajo
  - 5.1. Ramas remotas
  - 5.2. Ramas locales
  - 5.3. Commits organizados
  - 5.4. Ramas y niveles de estabilidad

## 1. ¿Qué es Git?

### 1.1. Sistema de control de versiones (SCV)

El control de versiones es un sistema que registra los cambios en un conjunto de archivos a través del tiempo para que pueda recuperar versiones específicas más tarde. Esto es Git exactamente. Gracias a esto podremos trabajar de forma concurrente sin preocuparnos de estropear el proyecto o el trabajo de los demás.

### 1.2. Sistema centralizado vs. sistema distribuido

Existen dos tipos de SCV que se diferencian en la forma en la que almacenan los archivos: sistemas centralizados y sistemas distribuidos.

- **Sistemas centralizados:** Utilizan un único servidor en el que se almacenan todos los archivos versionados. Todos los usuarios deben acceder al servidor para trabajar con los ficheros, puesto que solamente se encuentran ahí. Si la conexión con el servidor se perdiera o el usuario no tuviera acceso a internet, ningún cambio podrá registrarse, y se deberá esperar a tener acceso de nuevo. Acceder al servidor conlleva además una cierta latencia en las operaciones.
- **Sistemas distribuidos:** Cada usuario posee una copia del repositorio en su propia máquina. El concepto de servidor central no existe desde el punto de vista del SCV, aunque comúnmente se establece un servidor remoto como *central* para poder coordinar a un cierto equipo. En caso de no disponer de una conexión con el servidor, el usuario puede realizar cambios localmente y *subirlos* más adelante, sin que esto afecte a su forma de trabajar. Además, como todos los usuarios disponen de una copia del repositorio, en caso de pérdida de datos en el servidor, el proyecto es fácil de recuperar.

# JUGADOR #12

Documentación referencial: manual de GIT

Ingeniería del Software, 2013

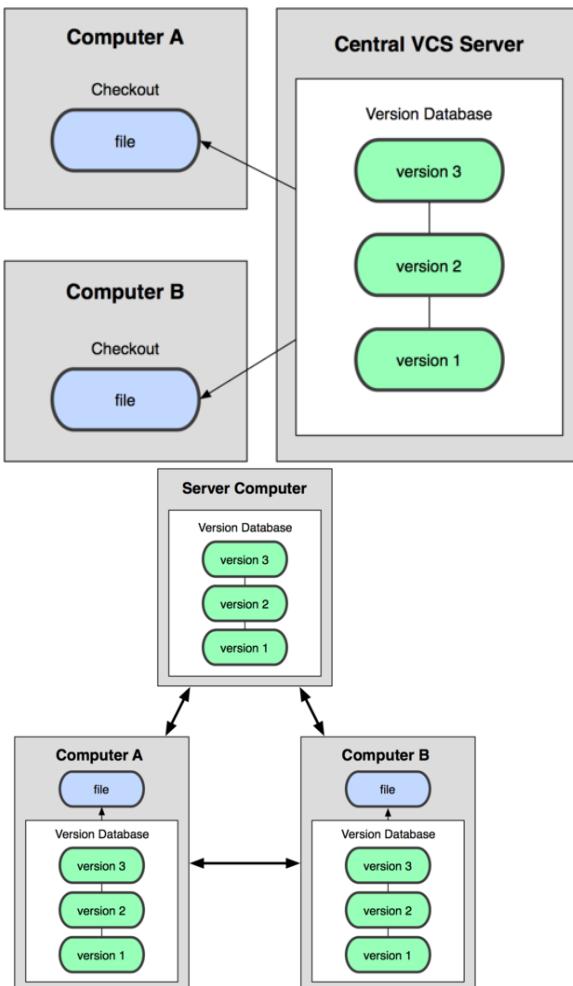


Diagrama de un sistema centralizado (izquierda) y un sistema distribuido (derecha).

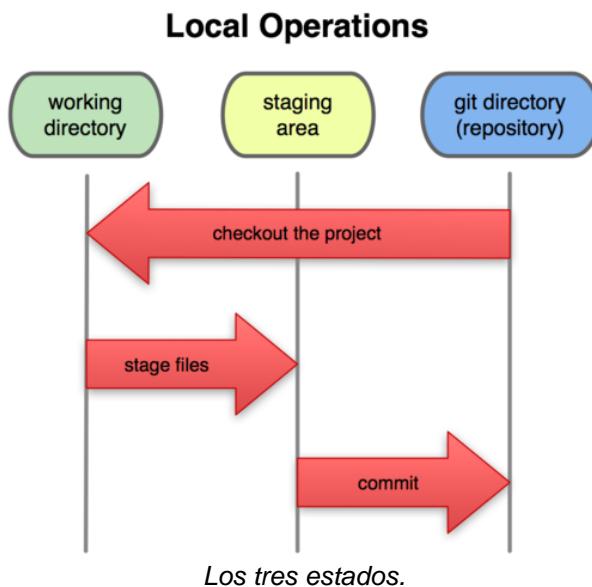
### 1.3. Los tres estados

Cada archivo gestionado mediante Git puede estar en uno de los siguientes estados:

- **Committed:** Fichero almacenado completamente en el repositorio en su última versión
- **Modified:** Fichero modificado en el directorio de trabajo, de cuyos cambios Git aún no sabe nada.
- **Staged:** Fichero modificado en el directorio de trabajo, cuyos cambios han sido añadidos al *índice* (staging area) para formar parte del siguiente *commit*.

En ocasiones un fichero puede encontrarse a la vez en los estados *modified* y *staged* si sólo hemos añadido al *índice* parte del fichero o si lo hemos modificado tras añadirlo.

# JUGADOR #12



## 2. Guía de instalación de Git

### 2.1. Windows

En primer lugar, descargamos Git del siguiente enlace: <http://git-scm.com/download/win>

Una vez descargado, ejecutamos el instalador y personalizamos la instalación. Los puntos donde tenemos elección son los siguientes:

- Cómo usar Git desde la línea de comandos: Podemos elegir si queremos usar Git únicamente desde Bash (primera opción), si queremos poder utilizarlo también desde la consola de Windows (segunda opción), y si además queremos instalar utilidades Unix extra (tercera opción). Para no complicar las cosas, seleccionamos la primera opción.
- Cómo se tratan los finales de línea en los ficheros de texto: Debido a que Windows utiliza una combinación de caracteres para representar el final de línea diferente a la de los sistemas POSIX como Linux o Mac, Git debe saber cómo tratarlos. Aquí podemos elegir entre tres opciones, de las cuales nos interesa la primera. Git convertirá todos los saltos al formato de Windows en el directorio de trabajo, y al formato de POSIX en el repositorio. De esta forma el repositorio trabajará de forma uniforme con el formato POSIX, pero cada uno usará al editar los ficheros el formato que mejor le convenga.

# JUGADOR #12

Documentación referencial: manual de GIT

Ingeniería del Software, 2013



Ajuste de la terminal en la que se ejecuta Git.



Configuración del salto de línea.

## 2.2. Linux

La instalación en Linux es muy simple, tan solo necesitamos un comando.

Dependiendo de la distribución usada, el comando es diferente. En el siguiente enlace está la lista con todos los comandos: <http://git-scm.com/download/linux>

## 3. Configuración básica inicial

### 3.1. Identificación

# JUGADOR #12

Documentación referencial: manual de GIT

Ingeniería del Software, 2013

Para que nuestros *commits* aparezcan con nuestra información, debemos configurar en Git nuestro nombre de usuario y correo electrónico:

```
$ git config --global user.name "Your  
Name"  
$ git config --global user.email  
email@gmail.com
```

## 3.2. Clave RSA

Para poder trabajar con GitHub sin tener que introducir nuestro usuario y contraseña cada vez que realicemos un *push* o *pull*, debemos configurar una clave RSA que nos identificará.

### 3.2.1. Windows

Generamos una nueva clave RSA:

```
$ ssh-keygen -t rsa -C "your_email@youremail.com"  
(/c/Users/you/.ssh/id_rsa): #Generating public/private rsa key pair.  
Enter file in which to save the key [Press enter]
```

Tras esto nos pedirá una contraseña. Podemos elegir dejarla en blanco, aunque no es recomendable.

```
Enter passphrase (empty for no passphrase): [Type a passphrase]  
Enter same passphrase again: [Type passphrase again]
```

Si todo va bien, debería salir algo parecido a esto

```
Your identification has been saved in /c/Users/you/.ssh/id_rsa.  
Your public key has been saved in /c/Users/you/.ssh/id_rsa.pub.  
The key fingerprint is:  
#01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db  
your_email@youremail.com
```

A continuación introducimos el siguiente comando, que copiará al portapapeles la clave pública:

```
$ clip <  
~/.ssh/id_rsa.pub
```

Ahora hay que añadir la clave en nuestra cuenta de GitHub. Para ello nos dirigimos a la configuración de nuestra cuenta, y en la sección *SSH Keys* damos al botón de añadir nueva clave y pegamos la clave. El nombre de la clave es irrelevante.

Comprobamos que todo es correcto:

## JUGADOR #12

Documentación referencial: manual de GIT

Ingeniería del Software, 2013

```
$ ssh -T  
git@github.com
```

Aparecerá el siguiente mensaje:

```
The authenticity of host 'github.com (207.97.227.239)' can't be  
established.RSA key fingerprint is  
16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.  
Are you sure you want to continue connecting (yes/no)? [Type yes]
```

Finalmente aparecerá este mensaje, que nos indica la cuenta con la que acabamos de identificarnos.

### 3.3.2. Linux

Generamos una nueva clave RSA:

```
$ ssh-keygen -t rsa -C "your_email@youremail.com"  
(/c/Users/you/.ssh/id_rsa): #Generating public/private rsa key pair.  
Enter file in which to save the key [Press enter]
```

Tras esto nos pedirá una contraseña. Podemos elegir dejarla en blanco, aunque no es recomendable.

```
Enter passphrase (empty for no passphrase): [Type a passphrase]  
Enter same passphrase again: [Type passphrase again]
```

Si todo va bien, debería salir algo parecido a esto.

```
Your identification has been saved in ~/.ssh/id_rsa.  
Your public key has been saved in ~/.ssh/id_rsa.pub.  
The key fingerprint is:  
#01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db  
your_email@youremail.com
```

A continuación introducimos los siguientes, que copiará al portapapeles la clave pública:

```
$ sudo apt-get install xclip #Downloads and installs xclip  
$ xclip -sel clip < ~/.ssh/id_rsa.pub #Copies the contents of the  
id_rsa.pub file to your clipboard
```

Ahora hay que añadir la clave en nuestra cuenta de GitHub. Para ello nos dirigimos a la configuración de nuestra cuenta, y en la sección *SSH Keys* damos al botón de añadir nueva clave y pegamos la clave. El nombre de la clave es irrelevante.

Comprobamos que todo es correcto:

```
$ ssh -T  
git@github.com
```

Aparecerá el siguiente mensaje:

```
The authenticity of host 'github.com (207.97.227.239)' can't be  
established.RSA key fingerprint is  
16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.  
Are you sure you want to continue connecting (yes/no)? [Type yes]
```

Finalmente aparecerá este mensaje, que nos indica la cuenta con la que acabamos de identificarnos.

## 4. Acciones básicas en Git

A continuación se explican los comandos básicos de Git y su uso. Para cualquier duda podemos consultar la ayuda de git, mediante git help <comando> o git <comando> --help.

### 4.1. Añadir ficheros al índice (add)

Para poder realizar *commits*, antes debemos añadir las modificaciones al índice (*staging area*). Para ello utilizaremos el comando git add <ficheros>. Como parámetros pasamos los nombres de los ficheros o directorios que queremos añadir.

Una versión muy útil de este comando es git add -u <ficheros>, que añadirá todos los cambios realizados en los ficheros de los que Git ya tenga algún registro.

Git incluye una herramienta para añadir ficheros al índice de forma interactiva, usable desde git add -i <ficheros>. Su uso puede consultarse aquí: <http://git-scm.com/book/en/Git-Tools-Interactive-Staging>

### 4.2. Borrar ficheros (rm)

Tanto si deseamos borrar un fichero de un repositorio como si queremos que un fichero que ha desaparecido de nuestro directorio de trabajo figure como eliminado, necesitamos usar el comando git rm <ficheros>.

Este comando eliminará el fichero si existe y lo marcará como borrado en el repositorio. Si lo que queremos es quitar un fichero del repositorio conservando la versión existente en el directorio de trabajo, debemos usar la versión git rm --cached <ficheros>.

## 4.3. Comprobar el estado de los ficheros (status)

Antes de realizar cualquier acción sobre el repositorio, es conveniente saber previamente en qué estado se encuentra. Para ello utilizaremos el comando git status o su versión de listado reducido git status -s.

Este comando nos informará de todos aquellos ficheros que no estén presentes o hayan sufrido cambios desde el último *commit*. Los ficheros actualizados no aparecerán en el listado de este comando.

## 4.4. Guardar los cambios en el repositorio local (commit)

El centro de todo el sistema de versiones es el *commit*, el cual podemos realizar mediante el comando git commit. Un *commit* almacenará en el repositorio todos los cambios del índice introducidos con otros comandos (particularmente con git add), y los almacenará en forma de *delta*, es decir, modificaciones desde el último *commit*.

Existen algunas opciones dignas de mención para este comando:

- -m "mensaje": Se usará el mensaje indicado como mensaje de *commit*. Si no incluimos esta opción, Git lanzará el editor por defecto (vim en Windows).
- -a: Añade automáticamente todos los ficheros de los que Git tiene registro al índice antes de realizar el *commit*. Esta opción equivale a ejecutar el comando git add -u justo antes de realizar el *commit*.

## 4.5. Deshacer cambios (reset)

Este comando es demasiado grande y complejo como para que os lo explique porque, sinceramente, no termino de saber qué narices hace. Consultad la página del manual (<http://www.kernel.org/pub/software/scm/git/docs/git-reset.html>) y la sección de Pro Git (<http://git-scm.com/book/en/Git-Basics-Undoing-Things>).

## 4.6. Subir cambios al repositorio remoto (push)

Para compartir todos los cambios que realicemos con el resto del mundo, debemos realizar un *push*. La sintaxis de este comando es git push <remote> <branch>. En nuestro caso, la forma concreta de este comando será git push origin develop.

Mediante este comando, enviamos todos los commits de una cierta rama a un repositorio remoto, a su rama correspondiente. Esto sólo podremos hacerlo si la rama remota y la local comparten una base, es decir, *no se han realizado cambios en la rama remota desde que la obtuvimos*. Si esto no ocurre, Git nos informará y deberemos realizar un *pull* para incluir los cambios antes de nada.

## 4.7. Bajar cambios del repositorio remoto (pull)

Para poder ver los cambios realizados en el proyecto por otros usuarios, debemos descargar esos cambios mediante el comando git pull. Al descargar los cambios en una rama ya existente, Git automáticamente realizará un *merge*, lo cual puede llevar a conflictos si se ha modificado el mismo fichero varias veces.

## JUGADOR #12

Documentación referencial: manual de GIT

Ingeniería del Software, 2013

### 4.8. Trabajo con ramas (checkout)

Para organizar distintas versiones paralelas de trabajo, Git funciona en lo que se denominan *ramas*. Una rama no es más que un conjunto de cambios que llevan a un estado del proyecto. Las ramas tienen generalmente una base común, a partir de la cual divergen.

Mediante el comando `git checkout <rama>`, podemos cambiar de rama en cualquier momento. Al cambiar de rama, todos los ficheros del directorio de trabajo se actualizarán a los de la nueva rama, permitiendo modificarla.

La variante `git checkout -b <rama>` nos permite crear una nueva rama. La nueva rama será idéntica a la rama desde la cual ejecutamos el comando, y Git automáticamente cambiará a esta nueva rama.

d0Hay que tener en cuenta que Git no nos dejará cambiar de rama inmediatamente si hay cambios sin guardar tanto en el directorio de trabajo como en el índice. Antes de cambiar de rama, debemos realizar un *commit* o descartar los cambios.

### 4.9. Juntar la rama actual con otra (merge)

Probablemente el elemento más importante de Git, por su potencia y su uso, es el *merge*. Este comando nos permite mezclar dos ramas diferentes en una, juntando todos los cambios.

Para ello usamos `git merge <rama>`. Esto introducirá todos los cambios de la rama especificada en nuestra rama actual. *Nótese que la única rama modificada es la actual, nunca la especificada en el comando.* Tras ello, Git automáticamente guarda los cambios introducidos por *merge* con un nuevo *commit*, y deja nuestro espacio de trabajo limpio y listo para seguir trabajando. Es de notar que, de la misma forma, *merge* se quejará si intentamos ejecutarlo en una rama con cambios sin almacenar en un *commit*.

En ocasiones podemos necesitar revisar y editar los cambios de un *merge* antes de realizar el *commit* final, o incluso podemos querer editar el mensaje de dicho *commit*. Para ello podemos usar la opción `--no-commit`. Nunca se nos debe olvidar realizar el *commit* manualmente tras esto.

Generalmente, Git es lo suficientemente *listo* como para mezclar incluso aquellos ficheros que han sido editados en ambas ramas en múltiples ocasiones. Sólo cuando la misma línea se ha editado en ramas diferentes, Git entenderá que hay un conflicto y dejará en manos del usuario su resolución.

Tanto si existe un conflicto como si elegimos no realizar el *commit* automáticamente, Git se quedará en un estado intermedio en el que todos los cambios se han introducido en el espacio de trabajo pero no se han guardado. En el caso de un conflicto, debemos resolverlo, añadir el fichero conflictivo al índice mediante `git add`, y

## JUGADOR #12

Documentación referencial: manual de GIT

Ingeniería del Software, 2013

posteriormente realizar un *commit*. En ambos casos, es recomendable realizar un *commit* inmediatamente después de revisar y corregir posibles errores de mezcla y conflictos, y no realizar ningún otro cambio.

### 4.10. Modificar el punto de partida de una rama (rebase)

Con rebase podemos modificar el commit original del cual partió la rama. La sintaxis del comando es git rebase <rama>, lo cual modificará la base de la rama actual por el commit de la rama indicada.

Su uso principal es el de integrar en la rama actual los cambios de la rama de partida como si se hubieran realizado *antes* de los commits de nuestra propia rama. Esto también se puede conseguir mediante git merge, aunque el resultado es diferente.

Más información y un montón de diagramas en <http://git-scm.com/book/en/Git-Branching-Rebasing>.

### 4.11. Más información

Para más información y mayores referencias y datos, es recomendable leerse el libro on-line Pro Git. Cualquier cosa que no haya quedado clara de este mini-manual, viene mejor explicada en el libro con total seguridad.

## 5. Formas de trabajo

### 5.1. Ramas remotas

El repositorio remoto contará en todo momento con dos ramas fijas, master y develop, así como de una serie de ramas que se irán creando según sean necesarias para coordinar a subgrupos del grupo que estén trabajando en una misma característica.

La rama master no debe modificarse bajo ningún concepto: de ello se encargará una única persona designada en el grupo, que periódicamente incorporará los cambios estables a tiempo para las entregas.

La rama develop será la que modifiquemos, directa o indirectamente, al realizar cambios en el proyecto. Sólo debe modificarse directamente para cambios menores como corrección de *bugs* o pequeños ajustes realizados por una sola persona que no encajen en ramas específicas creadas posteriormente.

Las *ramas específicas* se crearán cuando sea necesario, y serán los puntos de coordinación de todo el grupo. Cada vez que se decida implementar una cierta característica en el proyecto, se creará una de estas ramas, y los que estén asignados a estas tareas trabajarán en esta nueva rama. Alguien, una sola persona dentro de cada rama, se encargará de incorporar los cambios en develop, cuidando de no subir código que no compile o rompa el resto de la aplicación.

## 5.2. Ramas locales

Aunque cada uno es libre de trabajar en casa como quiera, es recomendable seguir una metodología de trabajo concreta.

Cada vez que vayamos a ponernos a trabajar, descargaremos todos los cambios de la rama específica que tengamos asignada. A partir de aquí, crearemos una rama para cada nueva tarea que realicemos. Trabajaremos en esta nueva rama durante todo o parte del día, y al finalizar, mezclaremos los cambios en nuestra rama específica y procederemos a subirla de nuevo al servidor. La rama local que creamos inicialmente ya no sirve para nada y podemos proceder a borrarla usando git branch rm <rama> sin problema.

De esta manera estamos consiguiendo varias cosas:

- Si alguno de nuestros compañeros actualiza el código, lo hará en la rama específica, y no en la rama local nueva. Esto puede evitar conflictos si actualizamos frecuentemente la rama específica para ver los cambios del compañero.
- Si el código que estamos escribiendo no nos sirviera o quisieramos descartarlo, basta con borrar la rama.
- Es posible que queramos avanzar en una cierta característica mientras arreglamos un bug que acabamos de detectar. En lugar de corregir el bug en nuestra rama local y no incorporarlo hasta que acabemos, podemos volver a la rama específica, crear otra rama extra, corregir el bug y subir la corrección. Todo ello con nuestro código a medias, en una rama aparte, de la que nadie sabe nada. Posteriormente podemos incorporar los cambios a esta rama con un *merge* o un *rebase*, para tener el error corregido.

## 5.3. Commits organizados

Para que todos podamos saber en todo momento qué está pasando en la otra punta del proyecto, conviene mantener un orden y una organización en los *commits*. Esta organización se consigue con los siguientes aspectos:

- **Agrupar cambios similares en un *commit*, separar cambios diferentes en varios *commits*.**

Aunque pueda parecer una tontería, es conveniente agrupar los cambios en los *commits* por su lógica y no por su fecha. Si el mismo día, incluso a diferentes intervalos, hemos trabajado en dos partes distintas del proyecto, es conveniente integrarlas en el repositorio con *commits* separados.

Incluso si dentro del mismo fichero tocamos partes diferenciadas de código, conviene guardarlas en varios *commits*. Recordemos que Git trae una herramienta que permite añadir trozos de ficheros al índice en lugar de ficheros completos.

- **Evitar los *supercommits*.**

## JUGADOR #12

Documentación referencial: manual de GIT

Ingeniería del Software, 2013

Debemos evitar siempre realizar *commits* con cientos de cambios. En lugar de juntar todo nuestro trabajo de un día en un solo *commit*, conviene separarlo en pequeños grupos. Una forma de hacerlo es siguiendo el anterior punto, otra es realizar pequeños *commits* cada cierto tiempo.

Los *commits* no tienen por qué incluir cambios completos, depurados o testeados. Aunque sí conviene que el código esté mínimamente limpio y compile, dejar una característica a medio hacer en un *commit* no está prohibido.

- **Describir los cambios de cada commit correcta y completamente.**

Git obliga a incluir un mensaje con cada *commit*. Este mensaje debe incluir una descripción de los cambios realizados en dicho *commit*, de forma que leyendo este mensaje sepamos exactamente qué ha cambiado.

El formato que utilizaremos se compondrá de una primera línea, muy breve, que dará una idea general de los cambios, y un cuerpo donde explicaremos los cambios de forma detallada. El formato exacto del cuerpo depende de cada uno: Podemos listar los cambios uno a uno, de forma concisa, redactar una explicación de lo que se ha hecho, o ambas.

### 5.4. Ramas y niveles de estabilidad

Cada una de las categorías en las que hemos dividido el sistema de trabajo corresponde a un nivel de estabilidad:

- Las **ramas locales** corresponden a código activo y, por tanto, inestable. Cualquier cosa tiene cabida aquí.
- Las **ramas específicas** corresponden a código activo pero compartido. El código que se suba aquí no debería fallar a niveles básicos como la compilación, pero puede ser inestable a nivel de usabilidad.
- La **rama *develop*** corresponde a código en desarrollo pero con coherencia interna. El código que se suba aquí debe estar funcionando, aunque sea a nivel básico, y no romper otras funcionalidades.
- La **rama *master*** corresponde a código *estable*, susceptible de ser entregado al profesor. Todo lo que se suba a esta rama estará testeado apropiadamente y no deberá contener fallos.

## Introducción

Este documento presenta el tutorial más básico posible sobre la interacción entre PHP y MySQL.

Completamente original por Manuel Artero Anguita. Se ha usado la página web *PasteBin* para enlazar el código.

## Índice

1. Planteamiento
2. Qué recursos tenemos
3. Conectar con la base de datos
4. Hacer una consulta a la base de datos
5. Tratar los resultados obtenidos

# JUGADOR #12

Documentación referencial: conectar PHP y MySQL

Ingeniería del Software, 2013

## 1. Planteamiento

Imaginemos que en nuestra página los usuarios han escrito comentarios y nosotros los hemos ido guardando en una base de datos.

Para esta primera toma de contacto con PHP-MySQL vamos a imaginar que lo que queremos es mostrar todos los comentarios que tenemos almacenados por orden de creación.

## 2. Qué recursos tenemos

En la base de datos `datos` la tabla `comentarios` que tiene la siguiente pinta

`comentarios.sql`

<code>id</code>	<code>autor</code>	<code>contenido</code>	<code>fecha</code>
01	Pepe	Comentario de prueba	13-5-2012
02	Manu	NANANANA Batman	1-6-2012
02	Samu	Samu es un xixi	19-7-2012

En el archivo `comentarios_index.php` queremos mostrar como hemos dicho todos los comentarios almacenados.

## 3. Conectar con la base de datos

Lo primero que tenemos que hacer conectarnos con la base de datos para poder pedirle información, lo hacemos en dos pasos:

- Nos conectamos con MySQL
- Seleccionamos la base de datos que queremos, en nuestro caso dijimos que se llamaba **data**

El script php para conectarnos con una base de datos es el siguiente:

<http://pastebin.com/7TApPyMP>

## 4. Hacer una consulta a la base de datos

Ahora ya podemos pedir a la base de datos que nos de todos los comentarios ordenados por fecha (por ejemplo).

Esto es ya saber hacer una consulta en MySQL, pero a grandes rasgos le vamos a decir: "de la tabla comentarios, dame todas las filas ordenadas por fecha"

La consulta MySQL es la siguiente:

<http://pastebin.com/878d9ZBc>

## JUGADOR #12

Documentación referencial: conectar PHP y MySQL

Ingeniería del Software, 2013

Para llamar al script que nos conecta con la base de datos vamos a usar la función `require`. El código quedaría:

<http://pastebin.com/0t273SJA>

### 5. Tratar los resultados obtenidos

En la variable `$query` tenemos entonces los resultados

La forma va a ser recorrer cada fila de resultados y almacenarla en un array. Podremos acceder a las posiciones del array.

<http://pastebin.com/2ygm7rFU>

## Introducción

En este documento se detallan diversos enlaces a tutoriales tanto en inglés como en español sobre las tecnologías que se van a usar en el proyecto. Se han seleccionado los que han parecido mejor presentados y completos. Si alguien tiene especial problema con el inglés en alguno de ellos que avise y se tratará de hacer un tutorial simplificado.

Se recomienda, eso sí, realizar los tutoriales en el orden que se exponen aquí ya que hay elementos que pueden depender de otros. Por ejemplo, lo correcto es mirar primero HTML y después CSS para aplicar estilos a dichos HTML.

## Índice

1. Tutoriales
  - 1.1. HTML
  - 1.3. Javascript
  - 1.4. MySQL
  - 1.5. PHP
2. Herramientas
  - 2.1. XAMPP
  - 2.2. DB Designer
3. Referencias

## 1. Tutoriales

### 1.1. HTML

- [HTML básico \[1\]](#): Tutorial en español de una página bastante sencillita que contiene las etiquetas básicas (con ejemplos de uso) y aspectos avanzados de las mismas. Muy útil para recurrir a él y ver cómo se crean determinados elementos.
- [HTML medio \[2\]](#): Otro tutorial parecido al anterior. También en español y que contiene usos básicos y avanzados de HTML.

### 1.2. CSS

- [CSS básico \[3\]](#): Un sencillo tutorial de CSS en español que contiene elementos muy básicos. Lo importante en este apartado es aprender un poco como funciona un archivo CSS, no conocerse todos los atributos de estilo posibles. En caso de que sea necesario algo más avanzado, añadiré nuevos tutoriales.

### 1.3. Javascript

- [Javascript medio \[4\]](#): Tutorial Javascript en inglés. De todos los que vi fue el que me llamó más la atención. Si alguien necesita uno en español trataré de buscar uno decente. Con Javascript ocurre algo parecido a CSS: lo usaremos en puntos muy concretos (por ejemplo una primera validación de formularios) pero no será una pieza fundamental de la página.

### 1.4. MySQL

- [Página principal de MySQL \[5\]](#) : Enlace a la página de MySQL en español. Contiene un tutorial muy sencillito y muchísima información acerca de MySQL en general. Es una muy buena referencia en cuanto al tema.
- [Tutorial paso a paso \[6\]](#): Un tutorial paso a paso en español. No me gusta mucho porque no tiene índice ni una organización muy buena, pero es posible que sea de gran ayuda para seguirlo paso a paso.
- [Referencias directas \[7\]](#): Tutorial en inglés que junta los conceptos más usuales de MySQL en secciones y puede ser una referencia más directa a ciertos apartados (si buscamos algo en concreto).

### 1.5. PHP

- [Página oficial de PHP \[8\]](#): Al igual que ocurría con MySQL, ofrece un pequeño tutorial y ejemplos así como gran cantidad de información general de PHP. Lo bueno es que debajo de cada explicación teórica hay multitud de comentarios de usuarios que pueden ayudar a esclarecer las dudas.
- [PHP básico \[9\]](#): Tutorial muy sencillito en español con los elementos más básicos de PHP. Puede ser un buen punto de partida.
- [PHP medio \[10\]](#): Tutorial en inglés de PHP. Contiene apartados más básicos y avanzados (como PHP+Ajax). No es el más recomendable para comenzar.

## 2. Herramientas

### 2.1. XAMPP

XAMPP es una herramienta gratuita que resulta muy apropiada para tener un servidor personal en el ordenador sobre el que ir probando archivos. El programa incluye ya servicio para PHP, MySQL, etc. lo cual nos evita tener que instalarlo por separado. Contiene también servicios como PHPMyAdmin que nos permite gestionar la base de datos de forma gráfica (Esto no excluye en absoluto la necesidad de tener que conocer los comandos MySQL. Solo se debe usar para facilitar el uso de la BBDD). Tiene versión de Windows, Linux y MAC.

No es obligatorio pero me parece la forma más sencilla de montar un servidor de pruebas personal. Otra opción (como dije antes) sería [instalar un servidor paso a paso \[11\]](#).

### 2.2. DB Designer

Se trata de un [programa open source para diseño de BBDD \[12\]](#) especialmente optimizado para MySQL. Es capaz de definir el esquema de una base de datos a partir del diagrama entidad-relación.

## 3. Referencias

- [1]: Esther Mª Pulido Alonso, Octavio Domínguez Arteaga y Cristina Jiménez Peñate.  
Universidad de Las Palmas de Gran Canaria.  
[http://gias720.dis.ulpgc.es/Gias/Cursos/Tutorial\\_html/indice.htm](http://gias720.dis.ulpgc.es/Gias/Cursos/Tutorial_html/indice.htm)
- [2]: Página web de Virtualnauta, <http://www.virtualnauta.com/html-introduccion>
- [3]: Página web de Virtualnauta, <http://www.virtualnauta.com/css-introduccion>
- [4]: Página web de W3Schools, [http://www.w3schools.com/js/js\\_intro.asp](http://www.w3schools.com/js/js_intro.asp)
- [5]: Página web oficial de documentación MySQL,  
<http://dev.mysql.com/doc/refman/5.0/es/tutorial.html>
- [6]: Página web de “Programación en castellano”, autoría de los hermanos Carrero,  
[http://www.programacion.com/articulo/tutorial\\_basico\\_de\\_mysql\\_189/1](http://www.programacion.com/articulo/tutorial_basico_de_mysql_189/1)
- [7]: Página web de TutorialsPoint, <http://www.tutorialspoint.com/mysql/index.htm>
- [8]: Página web oficial de PHP, <http://php.net/manual/es/tutorial.php>
- [9]: Alfonso E. Martínez de Castro, 2007, <http://tutorialphp.net/>
- [10]: Página web de W3Schools, [http://www.w3schools.com/php/php\\_intro.asp](http://www.w3schools.com/php/php_intro.asp)
- [11]: Página web de Apache friends, <http://www.apachefriends.org/es/xampp.html>
- [12]: Página oficial de la empresa FabForce. Sección de DBDesigner  
<http://www.fabforce.net/dbdesigner4/>

## Introducción

La guía de estilo de la documentación nace de la necesidad de hacer una documentación uniforme y unificada. No se puede presentar una documentación en la que los documentos cambien de estilo (formato, forma de escritura, estructuración etc) a medida que es leída. Esta guía sienta unas bases que todos respetaremos a la hora de hacer documentación.

## Índice

1. Normas generales de estilo
  - 1.1 Normas sobre el drive
  - 1.2. Normas de documentación
2. Estilos
3. Párrafos
4. Las mayúsculas
5. El índice
6. Numeración de los epígrafes y sus partes
  - 6.1. Otras subdivisiones menores
7. La cursiva
8. La negrita
10. Comillas
11. Uso del paréntesis
12. Referencias
13. Enlaces
14. Fechas
15. Números
16. Unidades
17. Imágenes y tablas

# JUGADOR #12

Documentación referencial: guía de estilo de documentación Ingeniería del Software, 2013

## 1. Normas generales de estilo

### 1.1 Normas sobre el drive

- Todo documento no acabado se situará en la parte de desarrollo. Si un archivo todavía no tiene el formato que indica esta guía de estilo se entenderá que no está acabado.
- No habrá elementos repetidos. Cuando un documento pase a la zona de documentación se borrará de la de desarrollo.
- Por petición del profesor, todos los documentos tendrán extensión google docs. Hay una opción de convertir automáticamente cualquier documento a esta extensión en el navegador de carpetas. Por favor, activadla.
- Por petición del profesor, todas las carpetas y los documentos tendrán una descripción de lo que hay dentro. La descripción de un documento se pone en el navegador de carpetas, seleccionando la carpeta o el documento, en la opción de vista previa.

### 1.2. Normas de documentación

A la hora de redactar, deben tenerse en cuenta algunos principios básicos que facilitarán tanto la elaboración como la lectura de los artículos:

- Deben respetarse las normas del idioma español contenidas en las obras académicas de la Asociación de Academias de la Lengua Española.
- El texto de todos los documentos se escribirán en letra Arial, tamaño 10, color negro.
- No repetiremos el título del documento en la primera línea del contenido. Ya se ve arriba a la izquierda de la pantalla.
- Se mantendrá el estilo formal en todos los documentos, aunque el contenido no sea académico o nos dirijamos entre nosotros.
  1. La **estructura** de la exposición puede seguir el siguiente modelo en la mayoría de los casos: introducción de la idea, desarrollo de los diferentes aspectos a considerar y, finalmente, conclusión.
  2. Desarrollar **una idea central** en cada párrafo.
  3. Siempre que sea posible, escribir **oraciones cortas**, convenientemente separadas por puntos. Es habitual ver oraciones que se alargan innecesariamente tres, cuatro y más líneas sin una pausa principal (punto).
  4. Redactar las oraciones siguiendo su **orden lógico**: sujeto, verbo y complementos. Alterar esa estructura o abusar de pasivas añade complejidad, que puede ser innecesaria.
  5. No abusar de **enlaces** coordinados ni subordinados. El exceso de coordinación o subordinación, alargando las oraciones varias líneas, puede hacer que perdamos de vista la idea principal.
  6. Evitar los **circunloquios** —rodeos de palabras para dar a entender algo que hubiera podido expresarse más brevemente—. La oración «el rey llamó a su hermano, que acudió a la cita, y le propuso un trato» puede expresarse más concisamente como «el rey propuso un trato a su hermano» si no son relevantes los hechos de que lo llamase y aquel acudiese. El contexto suele proporcionar datos sobre la relevancia. «Estos animales suelen encontrarse la mayoría de las

# JUGADOR #12

Documentación referencial: guía de estilo de documentación Ingeniería del Software, 2013

veces en lugares aislados de la luz donde el grado de humedad es más alto de lo normal» es un circunloquio que expresa lo mismo que «estos animales se encuentran casi siempre en lugares oscuros y húmedos».

## 2. Estilos

Para todo el texto se usará letra Arial y el color negro.

Los apartados principales — introducción, índice, apéndices y anexos, referencias y bibliografía — se escribirán con *título 1*. Para los títulos de los apartados se usará el *título 2*. Para los subapartados se usará el *título 3*. Todos los títulos se alinearán a la izquierda.

Para el texto normal se usará tamaño 11. Google docs se configura inicialmente tamaño de letra 11. En el apartado de estilos se puede poner por defecto que el texto normal use letra tamaño 11. El texto se alineará a izquierda.

## 3. Párrafos

Los párrafos se separarán con una línea en blanco entre ellos y usarán un interlineado de 1,15. Todas las líneas de un párrafo estarán sangradas a la misma altura y se situarán debajo de título.

Cuando un documento tenga más de cuatro o cinco párrafos largos, es recomendable que esté dividido en secciones. Entre dos secciones del texto habrá dos líneas en blanco.

## 4. Las mayúsculas

Los títulos de las carpetas, documentos y secciones deben estar en minúscula excepto la primera letra de cada frase, con excepción de las palabras que requieran ir en mayúscula de acuerdo con las reglas ortográficas.

Uso incorrecto:

- Repercusión sobre El Planeta Tierra
- Discos Publicados En Español

Uso correcto:

- Repercusión sobre el planeta Tierra
- Discos publicados en español

## 5. El índice

El índice de contenido debe ir siempre en las primeras páginas del libro; justo después del título, y la introducción. Deberán aparecer de los epígrafes y cuantas unidades de texto menores consideremos oportunas siempre y cuando sean divisiones formales del texto. Si el texto de índice no cabe en una sola línea, deberá componerse en párrafo francés, es decir, todas las líneas llevarán una pequeña sangría excepto la primera.

En el índice normalmente aparece la página del documento donde comienza el epígrafe, pero esta opción no está disponible en el google docs. En vez de eso, enlazaremos el índice con su epígrafe mediante un hipervínculo. Cuando hagamos el enlace sólo enlazaremos el texto, los números «1.», «1.1.», «1.2.», etc no formarán parte del enlace.

# JUGADOR #12

Documentación referencial: guía de estilo de documentación Ingeniería del Software, 2013

## 6. Numeración de los epígrafes y sus partes

Se numerarán los epígrafes mediante cifras arábigas correlativas separadas cada una de la siguiente por un punto. La numeración arábiga decimal deberá empezar por el número uno; éste podrá subdividirse en otras partes que siempre empezarán por el número uno. Entre el último punto de la numeración y el texto habrá dos espacios

- 1.
- 2.
- 1.1.
- 2.1.
- 2.1.1

El sentido común deberá guiarnos para saber cuántas subdivisiones son necesarias en un texto, y hasta qué punto las subdivisiones cumplen su misión de facilitar la comprensión de un texto o realmente la dificultan.

El google docs no permite que éste tipo de numeración se haga de forma automática. Hay que hacerlo a mano y las sangrías también.

### 6.1. Otras subdivisiones menores

Para otras listas y enumeraciones usamos las listas con viñetas. Las enumeraciones simples han de introducirse con dos puntos (:)

Ejemplo de enumeración:

- Patatas
- Frutas
  - Manzanas
  - Peras
- Harina

Si no te saliese el símbolo que deseas para las listas puedes cambiarlo en Formato->Estilos de lista

## 7. La cursiva

Usa cursiva para los títulos de las obras literarias y artísticas, como álbumes musicales, libros, películas, pinturas, series de televisión, nombres científicos y dedicatorias. También los términos y locuciones extranjeras no admitidos por la RAE.

En documentos informatizados la cursiva siempre sustituye al subrayado.

## 8. La negrita

El uso de la negrita debe restringirse a los títulos y subtítulos que encabezan un apartado. De manera excepcional podrá admitirse el uso de la negrita dentro del texto en enumeraciones o para destacar unidades menores que la palabra que, por su tamaño, de otro modo no destacarían.

Existen cuatro tipos de triángulos:

- El triángulo **rectángulo** es el que tiene un ángulo de 90 grados.
- El triángulo **isósceles** es aquel que tiene dos lados iguales y uno desigual.
- El triángulo **escaleno** es aquel que tiene los tres lados desiguales.
- El triángulo **equilátero** es aquel que tiene los tres lados iguales y por lo tanto sus tres ángulos miden 60 grados.

## 10. Comillas

Las comillas son un signo ortográfico doble del cual se usan diferentes tipos en español:

- las comillas angulares, también llamadas latinas o españolas (« »)
- las inglesas (“ ”)
- las simples (‘ ’)

En los textos impresos e informáticos, en oposición a los manuscritos, el diccionario panhispánico de dudas<sup>1</sup> recomienda utilizar en primera instancia las comillas angulares, reservando los otros tipos para cuando deban añadirse comillas a partes de un texto ya previamente entrecomiñado. En este caso, las simples se emplean en último lugar: «Antonio me dijo: “Vaya ‘cacharro’ que se ha comprado Julián”».

## 11. Uso del paréntesis

Hay gran cantidad de artículos que, al insertar incisos, comentarios y aclaraciones entre paréntesis, no hacen más que dificultar la lectura. En muchos casos, tales incisos pueden ir entre comas; en otros, simplemente no son necesarios. En su lugar, si son imprescindibles, pueden utilizarse bien notas a pie de página. Otra opción, en algunos casos más correcta que la utilización de paréntesis, es el uso de guiones largos ( — ), aunque deberían usarse con la misma prudencia que los paréntesis. El código unicode del guión largo es 2014.

## 12. Referencias

Las referencias son fundamentales a la hora de redactar. Si pusiéramos toda la información citada en el texto, éste se volvería muy tedioso de leer. Por ello, las referencias se reseñan en una sección aparte en el texto. Al final del documento, antes que la bibliografía. Cada vez que se haga una reseña en el texto se pone un número entre corchetes [1] para señalar el número de referencia.

# JUGADOR #12

Documentación referencial: guía de estilo de documentación Ingeniería del Software, 2013

## 13. Enlaces

Crea enlaces solo donde sean relevantes para el contexto: no es útil y puede ser muy molesto marcar todas las palabras posibles como hipervínculo. Los enlaces no deben desmerecer el artículo haciéndolo más difícil de leer. Una alta densidad de enlaces puede apartar la atención de los enlaces de valor elevado que querrías que tus lectores siguieran.

Los enlaces irán en azul y sin subrayar. Para quitar el subrayado por defecto selecciona el enlace y quita el subrayado de formal normal.

## 14. Fechas

El formato de las fechas dentro del texto será el siguiente: 15 de enero 1985. Los números de los años nunca llevan punto y los meses deben ir en minúsculas.

## 15. Números

Un real decreto (1317/1989 de 27 de octubre)<sup>2</sup> regula en España la expresión de las unidades legales de medida. Según esta ley, debe utilizarse la coma para separar la parte entera de la parte decimal en los números.

25,50      \*25'50      \*25.50

Aunque el uso del punto para separar los millares está generalizado; para facilitar la lectura de cantidades que tienen muchas cifras, la ley recomienda dividir los números en grupos de tres cifras, hacia la izquierda y hacia la derecha de la coma, mediante un espacio.

12 324 370,234 431

Esta separación en grupos se utiliza para los números que tienen cinco o más cifras a cada lado de la coma.

## 16. Unidades

Se usarán las unidades y símbolos del Sistema Internacional de Unidades (SI). La memoria se indica en las unidades de almacenamiento bit (símbolo **bit**) o byte (símbolo **B**), con sus correspondientes múltiplos decimales: kilobit (**kbit**), megabit (**Mbit**), kilobyte (**kB**), megabyte (**MB**), gigabyte (**GB**), terabyte (**TB**), etc.

La velocidad de transferencia de datos se indica en las unidades bit por segundo (símbolo **bit/s**) o byte por segundo (símbolo **B/s**), con sus correspondientes múltiplos: kilobit por segundo (**kbit/s**), megabit por segundo (**Mbit/s**), kilobyte por segundo (**kB/s**), megabyte por segundo (**MB/s**), etc.

## 17. Imágenes y tablas

Las imágenes y las tablas las pondremos centradas en la hoja. Tendrán siempre un pie de imagen centrado debajo de la imagen con una letra tamaño 9. El pie será una descripción simple del contenido de la imagen.

# JUGADOR #12

Documentación referencial: guía de estilo de documentación Ingeniería del Software, 2013

## Referencias

1. [Diccionario panhispánico de dudas](http://lema.rae.es/dpd/): <http://lema.rae.es/dpd/>
2. [Real decreto](http://www.boe.es/diario_boe/txt.php?id=BOE-A-1989-25841): [http://www.boe.es/diario\\_boe/txt.php?id=BOE-A-1989-25841](http://www.boe.es/diario_boe/txt.php?id=BOE-A-1989-25841)

## Actas de reuniones

### 1. Acta de la primera reunión

Semana 22 - 26 de octubre

Votación abierta en: <http://doodle.com/qkctiint7trzn5n9?>

Fecha decidida: **martes 23, 18:00**

#### 1.1. Datos generales

- Asistencia: presentes todos los miembros del grupo.
- Duración aproximada: 2 horas.

#### 1.2. Propósito de la reunión

- Toma de contacto de los diez miembros del grupo.
- Primera definición del proyecto.

#### 1.3. Decisiones

##### 1.3.1. Definición del juego

- Definición desde cero de la estructura inicial del juego (explicación de los elementos que teníamos claros para partir todos desde una misma base).
- Discusión de diversos puntos de la dinámica del juego como la estructura de las acciones o eventos y la distinción de estos en:
  - Acciones duraderas en el tiempo
  - Acciones con efecto hasta el partido
  - Acciones a realizar durante el partido
- Decisión sobre la implementación de partidos dinámicos mediante “flancos de tiempo” prefijados.

##### 1.3.2. Organización del grupo

- Reparto de trabajo: redactar en grupos de dos, documentación para un apartado concreto del juego (resumen general, árbol de habilidades, recursos...).
- Fijados los martes a las 18:00 como reuniones habituales del grupo
- Fijada la fecha de la segunda reunión.

# JUGADOR #12

Documentación de organización: actas de reuniones

Ingeniería del Software, 2013

## 2. Acta de la segunda reunión

Semana 29 octubre - 4 de noviembre

Fecha decidida: **Lunes 29, 17:00**

### 2.1. Datos generales

- Asistencia: presentes todos los miembros del grupo salvo Roberto Marín Fernández por causa justificada.
- Duración aproximada: 3 horas.

### 2.2. Propósitos de la reunión

- Revisión de los documentos de definición del juego.
- Elegir entre las distintas versiones propuestas.
- Aclarar dudas sobre la definición del juego.

### 2.3. Decisiones

#### 2.3.1. Recursos

- Se han diferenciado los recursos como se propuso tras la primera reunión:
  - El dinero se queda como estaba, regeneración fija y acumulación ilimitada.
  - La influencia tiene dos valores, la total y la libre. La utilizada se queda bloqueada hasta terminar la acción. Además el total puede oscilar (en torno al 10%) por el éxito de tus acciones.
  - El "fervor" tiene un valor máximo (el fanatismo por tu equipo), y se invierte como valores fijos. Pero la recuperación depende del porcentaje total invertido y del éxito de la acción.
- Se plantea (una vez funcionando el juego), complicar el fervor para que los efectos de la acción dependan del porcentaje invertido.

#### 2.3.2. Organización del grupo

- Reparto de trabajo:
  - Estudiar las ventajas e inconvenientes de Yii vs Silex.
  - Crear el mockup.
  - Diseñar los árboles de habilidades.
  - Redactar tutorial sobre interacciones tecnológicas de las herramientas.

# JUGADOR #12

Documentación de organización: actas de reuniones

Ingeniería del Software, 2013

## 3. Acta de la tercera reunión

Semana del 5 al 11 de noviembre

Fecha decidida: **martes 6, 18:00**

### 3.1. Datos generales

- Asistencia: presentes todos los miembros del grupo.
- Duración aproximada: 2 horas.

### 3.2. Propósitos de la reunión

- Tomar decisión sobre qué framework de PHP vamos a usar (Yii o Silex).
- Explicar la estructura de las carpetas del Google Drive.
- Establecer fecha límite para realizar tutorial de Git.
- Definir el juego de las cartas y el simulador en Java para la próxima entrega.

### 3.3. Decisiones

#### 3.3.1. Decisión del Framework

- Se ha tomado la decisión de usar Yii como framework de PHP ya que obliga a estructurar el proyecto de una forma determinada mientras los proyectos en Silex tienen una estructura más libre y nos conviene tener una estructura fija para poder organizarnos mejor.

#### 3.3.2. Estructura de las carpetas del Google Drive

- El Google Drive se ha organizado de la siguiente manera: habrá dos carpetas con la misma estructura, una de desarrollo donde se irán guardando los documentos sin finalizar, y una de documentación donde se guardarán todos los archivos finalizados y que tengan el visto bueno. Se ha acordado que todos los documentos sigan una guía de estilo que hemos creado.

#### 3.3.3. Juego de cartas y simulador en Java

- Vamos a hacer un juego de cartas y un programa en Java que simulen una partida. El juego de cartas consiste en realizar una partida como si fuera un juego de mesa. Las cartas simularán las acciones que se podrán realizar antes y durante el partido. El programa en Java se encargará de simular la partida en función de las acciones que elija el jugador.

#### 3.3.4. Acciones de cada jugador

- Movedora de masas
  - a. Acciones antes del partido
    - i. Promocionar al equipo a través de las redes sociales (aumenta el aforo).
    - ii. Organizar una cena con los jugadores del equipo (mejora el nivel del equipo).
    - iii. Hackear la página oficial del equipo contrario (todas las acciones rivales serán penalizadas).
    - iv. Organizar un homenaje a un jugador leyenda (aumenta el aforo).

## JUGADOR #12

Documentación de organización: actas de reuniones

Ingeniería del Software, 2013

- b. Acciones durante el partido
  - i. Entrevista a un jugador durante el descanso (mejora el nivel del equipo en la segunda mitad)
  - ii. Organizar un evento durante el descanso (mejora el nivel del equipo en la segunda mitad).
- Empresario
  - a. Acciones antes del partido
    - i. Incentivo económico para los jugadores (mejora el nivel del equipo).
    - ii. Mejorar las gradas del estadio (aumenta el aforo).
    - iii. Financiar un evento (aumenta el aforo).
    - iv. Sobornar al juez de línea (empeora la fuerza de la afición del equipo rival).
  - b. Acciones durante el partido
    - i. Hacer apuestas (aumento de dinero).
    - ii. Llamar al juez de línea sobornado para que empiece a amañar el partido (disminuye la fuerza de la afición del equipo rival).
- Ultra
  - a. Acciones antes del partido
    - i. Preparar pancarta para el partido (aumenta la fuerza de la afición).
    - ii. Pintarte con los colores del equipo (aumenta la fuerza de la afición).
  - b. Acciones durante el partido
    - i. Organizar una ola (aumenta la fuerza de la afición).
    - ii. Bufandeo (aumenta la fuerza de la afición).
    - iii. Tirar una bengala (disminuye la fuerza de la afición contraria).
    - iv. Salir de espontáneo al campo desnudo (iguala las fuerzas de la afición de los equipos y expulsa al jugador automáticamente del campo).

### 3.3.5. Organización del grupo

- Reparto de trabajo:
  - Crear los mock-ups.
  - Realizar un juego con cartas y un programa en Java que simule el juego web.
  - Organizar unas clases sobre Yii y Git para coger los conceptos básicos.

# JUGADOR #12

Documentación de organización: actas de reuniones

Ingeniería del Software, 2013

## 4. Acta de la cuarta reunión

Semana del 12 al 18 de Noviembre

Fecha decidida: **Martes 13, 18:00**

### 4.1. Datos generales

- Asistencia: presentes todos los miembros del grupo.
- Duración aproximada: 2 horas y 20 minutos.

### 4.2. Propósitos de la reunión

- Explicar el funcionamiento de la herramienta YII. Instalación, uso y cómo vamos a trabajar con ella.
- Explicar el funcionamiento de la herramienta Git. Instalación uso y cómo vamos a trabajar con ella.
- El propósito es que todos comencemos con una base común sobre la que trabajar.

### 4.3. Decisiones tomadas

#### 4.3.1. Herramienta YII

Manuel Artero y Xabi López dieron una pequeña charla sobre la herramienta YII. Explicaron su funcionamiento y comunicaron cómo se ha decidido que trabajemos todos con la herramienta. Para ello crearon un documento dónde resumen lo que explicaron<sup>1</sup>.

#### 4.3.2. Estructura de las carpetas del Google Drive

Daniel Escoz dio una pequeña charla sobre la herramienta Git. Explicó su funcionamiento y comunicó cómo se ha decidido que trabajemos todos con la herramienta. Para ello creó un documento donde resume lo que explicó<sup>2</sup>.

## Referencias

1. Manuel Artero y Javier López, “Documento de la herramienta YI”
2. Pedro Morgado y Daniel Escoz, “Documento de la herramienta Git”

# JUGADOR #12

Documentación de organización: actas de reuniones

Ingeniería del Software, 2013

## 5. Acta de la quinta reunión

Semana del 19 al 25 de Noviembre

Fecha decidida: **Lunes 19, 16:00**

### 5.1. Datos generales

- Asistencia: presentes todos los miembros del grupo salvo Roberto Marín Fernández y Javier López Gómez por causa justificada.
- Duración aproximada: 3 horas.

### 5.2. Propósitos de la reunión

- Probar el juego de cartas junto con el simulador de Java.

### 5.3. Decisiones

#### 5.3.1. Conclusiones sobre el prototipo

- Partidas excesivamente largas y con demasiados recursos sobre la mesa
- Simulador de partidos en Java con resultados dispares: corregir este comportamiento
- Finalidad del prototipo alcanzada al permitirnos probar la dinámica del juego
- Crear documento en el que se recogen todas las conclusiones que se hagan a partir de distintas partidas con el prototipo

#### 5.3.2. Organización del grupo

- Reparto de trabajo:
  - Realizar el documento de especificación de requisitos software.
  - Realizar documento de la previsión de riesgos.
  - Realizar un Javadoc del simulador de Java.
  - Terminar el documento de la definición del juego.
  - Realizar un calendario del año.

# JUGADOR #12

Documentación de organización: actas de reuniones

Ingeniería del Software, 2013

## 6. Acta de la sexta reunión

Semana del 26 de Noviembre al 2 de Diciembre

Fecha decidida: **Martes 27, 18:00**

### 6.1. Datos generales

- Asistencia: presentes todos los miembros del grupo.
- Duración aproximada: 2 horas.

### 6.2. Propósitos de la reunión

- Redefinir la especificación de los árboles de habilidades.
- Redefinir la especificación de peñas y aficiones.
- Especificar los casos de uso y discriminar los que pertenecen a la arquitectura y los que no.

### 6.3. Decisiones

#### 6.3.1. Árboles de habilidades

El profesor determinó que no habría 3 árboles de habilidades diferentes sino que habría uno solo común para los 3 personajes. Esto supone redefinir cómo son los árboles de habilidades. El árbol de habilidades —lo llamamos árbol por motivos históricos— tendrá forma de grafo circular y conexo, y dependiendo del tipo de personaje que use el jugador (ultra, movedora de masas, empresario) se empezará en un punto del árbol (círculo) u otro. Para más información sobre cómo se determinó que sería consulte el archivo de «Definición del juego»[1](#).

#### 6.3.2. Peñas y aficiones

El profesor determinó que no habría distinción entre peñas y aficiones. Se hablaron sobre todos los aspectos del juego dónde afectaría este cambio. Vimos, que no suponía un gran cambio en el concepto del juego, sin embargo, implicaba un gran trabajo el cambiar los documentos de la documentación en los que se reflejaba el cambio.

## Referencias

1. Marina Bezares, “[Definición del juego](#)”.

## 7. Acta de la séptima reunión

Semana del 10-16 de diciembre

Fecha decidida: **martes 11 de diciembre**, 18:00

### 7.1. Datos generales

- Asistencia: presentes todos los miembros del grupo.
- Duración aproximada: 2 horas.

### 7.2. Propósitos de la reunión

- Explicar la organización decidida para abordar la implementación del proyecto
- Explicar el punto en el que se encuentra el proyecto en la planificación general
- Dejar el calendario detallado para las vacaciones de navidad

### 7.3. Decisiones

#### 7.3.1. Organización de la programación

Se ha explicado cómo qué organización se va a llevar desde este momento en el proyecto. Se ha sustituido la organización del grupo en diagramas por una alternativa mucho más práctica en el índice del repositorio.

También se ha hablado de la visión general del producto.

## 8. Acta de la octava reunión

Semana del 7 - 13 de enero del 2013

Fecha decidida: **martes 8 de enero; 17:00**

### 8.1. Datos generales

- Asistencia: presentes todos los miembros del grupo.
- Duración aproximada: 3 horas y 30 minutos.

### 8.2. Propósito de la reunión

- Hablar sobre el estado del proyecto pasadas las navidades
- Poner en conjunto el código escrito durante las vacaciones
- Organización del grupo de cara a la entrega de enero

### 8.3. Decisiones

#### 8.3.1. Puesta a punto del código

Se ha comenzado a unir las funcionalidades implementadas. Se ha planificado el reparto de tareas para completar los hitos puestos para la entrega de enero; todo indicado en el [documento de organización del grupo del repositorio](#).

# JUGADOR #12

Documentación de organización: actas de reuniones

Ingeniería del Software, 2013

## 9. Acta de la novena reunión

Semana del 14 - 20 de enero del 2013

Fecha decidida: **martes 15 de enero; 17:00**

### 9.1. Datos generales

- Asistencia: presentes todos los miembros del grupo.
- Duración aproximada: 2 horas y 30 minutos.

### 9.2. Propósito de la reunión

- Reunión del código implementado y revisión del mismo
- Implementación durante la reunión en busca de posibles problemas o atascos

# JUGADOR #12

Documentación de organización: actas de reuniones

Ingeniería del Software, 2013

## 10. Acta de la décima reunión

Semana del 8 - 14 de abril del 2013

Fecha decidida: **sábado 13 de abril; 17:00**

### 10.1. Datos generales

- Asistencia: presentes todos los miembros del grupo.
- Duración aproximada: 2 horas y 15 minutos.

### 10.2. Propósito de la reunión

- Decidir qué hacer ahora para terminar el proyecto
- Discutir la representación visual final del partido

### 10.3. Decisiones

#### 10.3.1. Brainstorming sobre la representación visual del partido

1. Números internos de la fórmula (por qué tengo este ánimo? El aforo?).
2. En qué repercuten las acciones que realizan.
3. El partido se representará como un croquis de un entrenador, y dibujos para cada estado para ver qué está pasando.
4. Estado, ánimo, aforo, 2 factores (ofensivo y defensivo), nivel. Estáticos durante el partido: aforo y nivel.
5. La información irá dividida en pestañas: Partido, Detalles y Crónica.

#### 10.3.2. Aspectos en los que centrarse a continuación

1. Acciones de partido.
2. Sistema de niveles
3. Comunicación.
4. Mostrar el partido.

#### 10.3.3. Conclusiones a partir de las encuestas realizadas en clase

1. No vamos a hacer foro.
2. Hacemos chat y sistema de notificaciones.
3. Entre 3-5 partidos por semana.
4. Partidos “cortos” de 15-20 minutos.

#### 10.3.4. Pendiente a realizar

1. Indicar cuánto tiempo le queda a una acción
2. En el listado de acciones, que aparezca cuantos recursos le quedan para completarse.

# JUGADOR #12

Documentación de organización: actas de entregas (1 cuatr.) Ingeniería del Software, 2013

## Introducción

En este documento están reunidas las actas de las cuatro entregas (revisiones) del proyecto durante el primer cuatrimestre.

Para cada una, se especifica la fecha, el material presentado y los puntos revisados así como el material a presentar en la siguiente revisión para los meses de octubre, noviembre y diciembre.

## Índice

1. Entrega de octubre
  - 1.1. Fecha
  - 1.2. Material presentado
  - 1.3. Puntos Revisados
  - 1.4. Material a preparar para la siguiente entrega
2. Entrega de noviembre
  - 2.1. Fecha
  - 2.2. Material presentado
  - 2.3. Puntos Revisados
  - 2.4. Material a preparar para la siguiente entrega
3. Entrega de diciembre
  - 3.1. Fecha
  - 3.2. Material presentado
  - 3.3. Puntos Revisados
  - 3.4. Material a preparar para la siguiente entrega
4. Entrega de enero
  - 4.1. Fecha
  - 4.2. Material presentado
5. Referencias

# JUGADOR #12

Documentación de organización: actas de entregas (1 cuatr.) Ingeniería del Software, 2013

## 1. Entrega de octubre

### 1.1. Fecha

30 Octubre 2012

### 1.2. Material presentado

- Presentación de las herramientas utilizadas para la gestión interna del grupo: Google-Groups para iniciar temas o hacer anuncios; y [Google-Drive como “repositorio” para la documentación generada](#) [1].
- Archivo resumen o carta de presentación de nuestro juego.

### 1.3. Puntos Revisados

- Necesidad de mantener la concordancia entre la documentación residente en Google-Groups y la carpeta de Google-Drive.
- Estructurar el archivo que registra el acta de las reuniones en
  - Fecha de la reunión
  - Propósito de la reunión
  - Puntos del día
  - Decisiones
- Organización de la documentación: estructurar los documentos con un árbol de jerarquía perfectamente definido. Separar archivos “en construcción” de archivos listos para su entrega (estos últimos irán en una carpeta llamada “documentación general”).
- Necesidad de dejar por escrito una guía de estilo para la documentación
- Añadir a la documentación el diagrama de juego.
- Añadir a la documentación un índice con la jerarquía de carpetas.
- Añadir a la documentación descripciones del contenido de los archivos.
- Revisión al documento resumen o carta de presentación: especificar el tipo de público y el género del juego; sustituir los ejemplos de acciones por un catálogo de acciones.

### 1.4. Material a preparar para la siguiente entrega

- Prototipo del juego

# JUGADOR #12

Documentación de organización: actas de entregas (1 cuatr.) Ingeniería del Software, 2013

## 2. Entrega de noviembre

### 2.1. Fecha

20 Noviembre 2012

### 2.2. Material presentado

- Presentación del [prototipo \[2\]](#), formado por un pequeño juego de cartas apoyado en un simulador de partidos escrito en Java.
- [Reestructuración del Google Drive \[3\]](#).
- Redacción de una [guía de estilo \[4\]](#) para todos los documentos.
- Redacción de las reglas del [juego de cartas \[5\]](#)
- [Definición interna de un partido \[6\]](#).
- [Definición del juego \[7\]](#) de navegador final.

### 2.3. Puntos Revisados

- Simplificación de diversos apartados del juego; redefinir un árbol de habilidades común para todos los personajes así como la eliminación de peñas del sistema de juego.
- Necesidad de crear un índice de los documentos del google Drive.

### 2.4. Material a preparar para la siguiente entrega

- Documento de especificación de requisitos.
- Calendario de planificación.
- Primera iteración sobre los casos de uso.

# JUGADOR #12

Documentación de organización: actas de entregas (1 cuatr.) Ingeniería del Software, 2013

## 3. Entrega de diciembre

### 3.1. Fecha

18 de Diciembre del 2012

### 3.2. Material presentado

- Primeros nueve casos de uso [8]
- Mock up de la página web desarrollado en HTML[9]
- Archivo ejecutable (.jar) del primer prototipo para los partidos desarrollado en Java
- Documento de métricas usadas en el proyecto (con anexo) [10]
- Documento de previsión de líneas de código (con anexo ) [11]
- Documento de especificación de requisitos [12]
- Documento de gestión de riesgos [13]
- Revisión de la definición del juego: simplificación del árbol de habilidades y las peñas [14]

### 3.3. Puntos Revisados

- Necesidad de comenzar a trabajar en el arte del proyecto
- Comienzo de la implementación del proyecto

### 3.4. Material a preparar para la siguiente entrega

- Entrega final del primer cuatrimestre: documentación generada durante todo el periodo.
- Implementación inicial.

## JUGADOR #12

Documentación de organización: actas de entregas (1 cuatr.) Ingeniería del Software, 2013

### 4. Entrega de enero

#### 4.1. Fecha

22 de enero del 2012

Nota: entrega final del primer cuatrimestre

#### 4.2. Material presentado

- Recopilación de la documentación; [enlace al índice \[15\]](#)
- Implementación inicial del proyecto; [enlace al repositorio \[16\]](#).

# JUGADOR #12

Documentación de organización: actas de entregas (1 cuatr.) Ingeniería del Software, 2013

## 5. Referencias

- [1]: Directorio de trabajo de Google Drive,  
[https://drive.google.com/?usp=chrome\\_app#folders/0B80G8C57fZdxX3JiaWhJcDRxLXc](https://drive.google.com/?usp=chrome_app#folders/0B80G8C57fZdxX3JiaWhJcDRxLXc)
- [2]: Proyecto SimuMatch en GitHub, <https://github.com/JugadorNumero12/SimuMatch>
- [3]: Documento de reestructuración de Google Drive, por Marina Bezares:  
[https://docs.google.com/drawings/d/1aML\\_auV2YM-xzWMikok5aV4njnBINtbE2caEReQ6nil/edit](https://docs.google.com/drawings/d/1aML_auV2YM-xzWMikok5aV4njnBINtbE2caEReQ6nil/edit)
- [4]: Guía de estilo de documentación, por Marina Bezares:  
<https://docs.google.com/document/d/1lisMChF8Kbewk8ngNlfEliliXMOqdNEzX8bxIzkgsj80/edit#headin>g=h.246jeceze306
- [5]: Reglas del juego de cartas, <https://docs.google.com/document/d/1RlhS4Y5YC0hCA6r-eP0ufWmulb9h9A-u4zWhiLesVHo/edit#headingh.ta9r1izehmge>
- [6] :Definición interna de un partido, <https://docs.google.com/presentation/d/1poBEec0cQCh-nCB7cnxJ3hWWI9WQUO9bgZH1vbL630Y/edit#slide=id.p>
- [7]: Documento de definición del juego, por Marina Bezares,  
<https://docs.google.com/document/d/13eyi2Sa447NXmsxGQ2RQ6cNjF2xmwyemcddt7VxcZPA/edit#heading=h.bmol66efqo27>
- [8]: Primeros casos de uso, por el grupo de trabajo de Jugador Número 12,  
<https://drive.google.com/?tab=mo&authuser=0#folders/0B80G8C57fZdxbjY0Ymt5Z2xZaEU>
- [9]: Mockup inicial de la web, por Javier López y Arturo Pareja,  
<https://github.com/JugadorNumero12/MockUp>
- [10]: Documento de métricas del proyecto, por Manuel Artero,  
<https://docs.google.com/document/d/1Cub9-jVFV-EljXf6h9kfBrN9hZj-4vGqKGXADD6icec/edit#heading=h.x2quj05lnjbz>
- [11]: Documento de previsión de líneas de código, por Javier López,  
<https://docs.google.com/document/d/1zllt-PmU0VfOqBMmNYBOliZVvQXJMViPBM4NFi5XU/edit#heading=h.ounqyldaijs>
- [12]: Documento d especificación de requisitos software, por Arturo Pareja, Javier López, Roberto Marín y Marina Bezares,  
<https://docs.google.com/document/d/1APDisxuw8AJPGsK4wZc5x1IPCQcCH5kFlVt1wFv-9jY/edit#heading=h.ydzno560smqr>
- [13]: Documento de gestión de riesgos, por Marcos Alarcón,  
<https://docs.google.com/document/d/1zQolzkCwG0EZiFbcQ4ymYlcf7K8fVtRNAY5Ex7qErs4/edit#heading=h.or316vx4288y>
- [14]: Revisión de la documentación de definición del juego,  
<https://docs.google.com/document/d/13eyi2Sa447NXmsxGQ2RQ6cNjF2xmwyemcddt7VxcZPA/edit#heading=h.bmol66efqo27>

## JUGADOR #12

**Documentación de organización: actas de entregas (1 cuatr.)** Ingeniería del Software, 2013

[15]: Enlace al índice del repositorio,

<https://docs.google.com/document/d/1O74KCMmeHSdCKXa1QvpnKYhBIC-olwKlnkWVcRgaWc8/edit#heading=h.q2ffb5bxxnog>

[16]: Enlace a la rama “master” del repositorio,

<https://github.com/JugadorNumero12/JugadorNum12/tree/master>

# JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

## Introducción

Mediante este documento se deja presente una lista de nombres de los miembros del equipo - junto con sus nombres de usuarios y emails - así como las planificaciones básicas para el proyecto y los repartos de tareas asignados.

## Índice

1. Lista de miembros
2. Métodos de planificación
3. Planificación de tareas informal del repositorio
  - 3.1. Diciembre
  - 3.2. Navidades
  - 3.3. Entrega de enero
  - 3.4. 18-28 de febrero
  - 3.5. 4-15 de marzo
  - 3.6. 20 de marzo
  - 3.7. 24 de abril
  - 3.8. 15 de mayo
4. Planificación obsoleta de Gantter
5. Reparto de tareas y seguimiento semanal

# JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

## 1. Lista de miembros

A continuación se presenta una lista de los componentes del equipo de desarrollo de Jugador Número 12. En ella podemos consultar el nombre completo de cada uno de ellos, sus nombres de usuario y direcciones de correo así como cualquier otro dato de interés.

Nombre completo	Usuario de GitHub	Dirección de email
Manuel Artero Anguita	manutero	<a href="mailto:manutero@gmail.com">manutero@gmail.com</a>
Marina Bezares Álvarez	Argaide	<a href="mailto:marinabezares@gmail.com">marinabezares@gmail.com</a>
Pedro Morgado Alarcón	PedroMorgado	<a href="mailto:p.morgado107@gmail.com">p.morgado107@gmail.com</a>
Daniel Escoz Solana	Darkhogg	<a href="mailto:darkhogg@gmail.com">darkhogg@gmail.com</a>
Samuel Méndez Galán	samuelmgalan	<a href="mailto:samuelmgalan@gmail.com">samuelmgalan@gmail.com</a>
Roberto Marín Fernández	ka2rober	<a href="mailto:ka2rober@gmail.com">ka2rober@gmail.com</a>
Javier López Gómez (Xabier)	Ltreaper	<a href="mailto:ltreap.face@gmail.com">ltreap.face@gmail.com</a> / <a href="mailto:xaby693@gmail.com">xaby693@gmail.com</a>
Marcos Alarcón Rivas	MarcosAlarconRivas	<a href="mailto:kernelpanic0x00@gmail.com">kernelpanic0x00@gmail.com</a>
Arturo Pareja García	arturopareja	<a href="mailto:arturo.pareja.garcia@gmail.com">arturo.pareja.garcia@gmail.com</a>
Alexandra Martín	leximagination	<a href="mailto:leximagination@gmail.com">leximagination@gmail.com</a>

## 2. Métodos de planificación

A lo largo del desarrollo del proyecto se han puesto a prueba diferentes métodos de planificación. A continuación se presenta una breve descripción y resultados obtenidos de los mismos:

- **Planificación a través de GitHub:** fue el primer método empleado. Se hacía uso del lenguaje MarkDown (.md) para redactar la sección de “Organización” disponible y accesible desde el repositorio del proyecto. Este tipo de reparto era muy esquemático y directo, orientado a conocer las tareas asignadas y funcionalidades a implementar con un simple vistazo.  
Estas planificaciones se actualizaban tras cada reunión. Durante éstas, se explicaba en detalle cuál sería el siguiente hito en el proyecto y de qué tarea se encargaría cada miembro del grupo. El documento de “Organización” del repositorio servía como herramienta a consultar ante las dudas.  
Este método resultaba muy adecuado para ver el reparto de tareas pero aún así resultaba demasiado esquemático.

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

- **Emails de planificación:** posteriormente, a mayores de la planificación de GitHub, se enviaba la misma a través de emails a todo el grupo de forma más detallada. Con esto se pretendía conseguir que ningún miembro del grupo tuviese dudas acerca de su labor durante el periodo entre reuniones además de seguir dejando constancia del reparto de tareas esquemático en GitHub.
- **Planificación en Gantter:** se dedicó gran cantidad de tiempo al desarrollo de una planificación global bastante precisa haciendo uso de la herramienta Gantter. El diseño de dicha planificación contenía gran información acerca de las diferentes iteraciones y fases del producto. No obstante esta metodología se descartó ya que llevaba mucho tiempo actualizarla de forma correcta cuando las fechas de los hitos no se cumplían y resultaba bastante caótica para los miembros del grupo a la hora de consultar las tareas asignadas.
- **Issues de GitHub:** por último, el método que recopila parte de todos los anteriores de forma muy efectiva: el sistema de issues del repositorio. Comenzó a emplearse relativamente tarde pero resultó ser de gran utilidad.  
Cada issue permitía definir de forma independiente los fragmentos de funcionalidad, errores e incluso documentación por crear o arreglar.  
El uso de los “milestones” permitió crear, de forma muy simple, una planificación por hitos hasta el final del proyecto.  
La unión de ambos (issues y milestones) proporcionó una herramienta perfecta para la planificación. Se veía de forma muy esquemática una organización temporal (el equivalente a Gantter) mediante “milestones” y, dado que cada miembro del grupo podía asignarse una issue concreta, quedaba patente al mismo tiempo el reparto de tareas.  
Otra gran ventaja es que cada usuario podía abrir nuevas issues para completar así como elegir aquellas con las que se sintiese más cómodo.  
Por último, la flexibilidad de asignación de issues a un milestone concreto permitió la reorganización de la planificación con gran facilidad.  
De haber conocido el sistema de issues del repositorio antes y las ventajas que aporta, es muy posible que hubiese sido empleado desde el principio.

### 3. Planificación de tareas informal del repositorio

En esta sección se presenta la planificación de tareas llevada a cabo a través de GitHub. Inicialmente era más explicativa pero a medida que se desarrollaron nuevas técnicas se puede observar cómo pasa a convertirse en un mero trámite.

#### 3.1. Diciembre

##### 1. Perspectiva general

###### **Controladores**

- Usuarios: Alex, Marina, Rober
- Registro: Alex
- Equipos: Marina, Sam

# JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

- Habilidades: Arturo, Dani
- Acciones: Dani, Pedro, Marcos
- Partidos: Marina, Arturo

## Modelos

- Usuarios: Rober
- Recursos: Alex
- Equipos: Sam
- Clasificacion: Marcos
- Habilidades: Arturo
- Acciones individuales: Pedro
- Acciones grupales: Marcos
- Acciones turno: Marcos
- Desbloqueadas: Alex
- Partidos: Sam
- Participaciones: Dani

## 2. Organización por personas

### Alex

Registro.index  
Usuarios.cuenta  
Desbloqueadas  
Recursos

### Arturo

Habilidades.adquirir  
Partidos.index  
Partidos.asistir  
Habilidades

### Dani

Habilidades.index  
Habilidades.ver  
Acciones.usar  
Participaciones

### Marcos

Clasificacion  
AccionesGrupales  
AccionesTurno  
Acciones.expulsar

### Marina

Usuarios.perfil  
Usuarios.ver

# JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

Partidos.previa  
Equipos.index

## Pedro

Acciones.index  
AccionesIndividuales  
Acciones.ver  
Acciones.participar

## Rober

Usuarios.index  
Usuarios.cambiarClave  
Usuarios.cambiarEmail  
Usuarios(+)

## Sam

Equipos.ver  
Equipos.cambiar  
Equipos  
Partidos

## 3.2. Navidades

### 1. Perspectiva general

#### Componentes

- Formula: Dani, Pedro
- Helper: Arturo
- Partido: Alex, Marcos
- Scripts: Rober
- Acciones: Arturo, Dani, Marina, Pedro, Rober, Sam

#### Data

- esquema.sql: Arturo, Xaby

#### Tests

- fixtures: Sam
- functional: Sam Alex

#### CSS

- Diseño de la jerarquia: Marina

### 2. Organización por personas

#### Alex

components/Partido.contructora  
components/Partido.cargaEstado  
components/Partido.guardaEstado  
components/Partido.inicializarEncuentro

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

functional/UsuariosTest.php

### **Arturo**

data/estructura.sql  
components/Acciones/Apostar.php  
components/Acciones/PunteroLaser.php  
components/Acciones/PromoverPartido.php

### **Dani**

components/Formula.php  
components/Acciones/BeberCerveza.php  
components/Acciones/HablarSpeaker.php

### **Marcos**

components/Partido.recogeAccionesTurno  
components/Partido.generaCronicaTurno  
components/Partido.generaBonificacion  
components/Partido.actualizaClasificacion

### **Marina**

css/comunes.css  
css/divisiones.css  
css/perfil.css  
components/Acciones/ContratarRRPP.php

### **Pedro**

components/Formula.php  
components/Acciones/FinanciarEvento.php  
components/Acciones/OrganizarHomenaje.php  
components/Partido.generarCronicaBase

### **Rober**

components/ejecutor\_turnos.php  
components/Helper.php  
components/Acciones/RetransmitirRRSS.php  
components/Acciones/IniciarOla.php

### **Sam**

fixtures/Equipos.php  
fixtures/Usuarios.php  
functional/.php  
components/Acciones/IncentivoEconomico.php

### 3.3 Entrega de enero

# JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

## 1. Perspectiva general

### Controladores

- Equipos.index: Dani
- Equipos.ver: Dani
- Equipos.cambiar: Rober
- Acciones.usar: Arturo
- Acciones.ver: Dani
- Acciones.participar: Marcos
- Acciones.expulsar: Dani
- Habilidades.index: Dani
- Habilidades.ver: Dani
- Partidos.asistir: Manu

### Vistas

- Usuarios.perfil: Marina, Alex, Pedro
- Usuarios.cuenta: Marina, Alex, Pedro
- Equipos.ver: Marina, Alex, Pedro
- Acciones.index: Marina, Alex, Pedro
- Acciones.usar: Marina, Alex, Pedro
- Acciones.ver: Marina, Alex, Pedro
- Habilidades.index: Marina, Alex, Pedro
- Habilidades.ver: Marina, Alex, Pedro

### Modelos

- Usuarios+: Sam
- Equipos+: Sam
- Acciones+: Sam
- Habilidades+: Sam

## 2. Organización por personas

### Alex

\* Vistas

### Arturo

\* Acciones.usar

### Dani

\* Equipos.index  
\* Equipos.ver  
\* Acciones.ver  
\* Acciones.expulsar  
\* Habilidades.index  
\* Habilidades.ver

# JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

## **Marcos**

Acciones.participar

## **Marina**

\* Vistas

## **Pedro**

\* Vistas

## **Rober**

Equipos.cambiar

## **Sam**

\* Usuarios+

\* Equipos+

\* Acciones+

\* Habilidades+

## 3.4. 18-28 de febrero

### 1. Perspectiva general

#### **Lógica del partido**

- Javier
- Rober
- Marina

#### **Fórmula del partido**

- Dani

#### **Front-end, estilo**

- Pedro
- Sam

#### **Módulo de actualización de recursos**

- Alex

#### **Test**

- Arturo

#### **Subir la página**

- Marcos

## 3.5. 4-15 de marzo

### 1. Perspectiva general

- Integración de los dibujos en la página.
- Terminar todas las funcionalidades previstas para la demo.
- 1 semana de margen

# JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

## 3.6. 20 de marzo

### 1. Perspectiva general

- Registro de los usuarios; selección de personaje y afición.
- Sistema de recursos completado.
- Desbloquear y usar habilidades individuales y grupales.
- Capacidad para asistir a un partido (sin interacción en él)
- Capacidad de completar acciones grupales con efecto en el estado inicial de un partido.
- Liga creada.
- 1º estado del arte del proyecto
- Documentación: diagrama de clases.
- Documentación: diagrama de secuencia.
- Documentación: casos de uso.

## 3.7. 24 de abril

### 1. Perspectiva general

- Juego completo (sin balancear ni pruebas finales).
- Interacción en los partidos.
- Árbol de habilidades desarrollado.
- Sistema de experiencia mejorado.
- Objetos integrados.
- Arte "básico" del juego terminado.
- Documentación completada.

## 3.8. 15 de mayo

### 1. Perspectiva general

- Juego completo: testeado y balanceado.
- Presentación de final de curso.

## 4. Planificación obsoleta de Gantter

En esta sección se muestra la planificación realizada mediante la herramienta Gantter a pesar de haber quedado obsoleta. Se adjunta para posibles fines referenciales.

# JUGADOR #12

**Documentación de organización: organización y reparto de tareas**      Ingeniería del Software, 2013

		Nombre	Duración	Inicio	Fin	Predecesoras	Recursos
1		⊖Fase Inicio	33d?	19/10/2012	04/12/2012		
2		Documento de visión del juego	30d?	19/10/2012	29/11/2012		
3		Documentación de especificación de requisitos	11d?	20/11/2012	04/12/2012	arturo.pareja.garcia[60%	
4		Revisión del documento de requisitos	1d?	27/11/2012	27/11/2012		
5		⊖Prototipo	22d?	29/10/2012	27/11/2012		
6		Especificación del prototipo	5d?	29/10/2012	02/11/2012	manu	
7		Implementación del prototipo	11d?	05/11/2012	19/11/2012	6	manu,marina,alex,robe
8		Documentación del prototipo	6d?	20/11/2012	27/11/2012	7	dani,sam
9		Documento de riesgos	10d?	20/11/2012	03/12/2012	marcos	
10		⊖Fase Elaboración	69d?	27/11/2012	01/03/2013		
11		⊖Casos de uso	11d?	27/11/2012	11/12/2012		
12		Primera iteración casos de uso de la arquitectura	5d	27/11/2012	03/12/2012		
13		Revisión 1 de los casos de uso	1d?	04/12/2012	04/12/2012	12	
14		Segunda iteración casos de uso de la arquitectura	3d?	05/12/2012	07/12/2012	12,13	
15		Revisión II de los casos de uso	1d?	07/12/2012	07/12/2012	13	
16		Tercera iteración casos de uso	1d?	10/12/2012	10/12/2012	14,15	
17		Revisión III (Final) de los casos de uso	1d	11/12/2012	11/12/2012	15,16	
18		Creación de diagramas de clase	7d	10/12/2012	18/12/2012		
19		Planificación de la estructura Yii	1d	10/12/2012	10/12/2012	manu,rober,xaby	
20		⊖Implementación de la arquitectura	58d?	12/12/2012	01/03/2013		
21		Vacaciones	14d	21/12/2012	09/01/2013		
22		Semana de "contingencias"	5d?	25/02/2013	01/03/2013		
23		Entrega de Diciembre	1d	18/12/2012	18/12/2012		
24		Entrega Enero	1d	22/01/2013	22/01/2013		
25		Revisión fin de año	1d	28/12/2012	28/12/2012		
26		Revisión para entrega Enero	1d	18/01/2013	18/01/2013		
27		⊖Módulo 1 (usuario - invitado)	10d	12/12/2012	25/12/2012		
28		Crear usuario (Módulo 1)	5d	12/12/2012	18/12/2012		
29		Test Módulo 1	5d	19/12/2012	25/12/2012		
30		⊖Módulo 2 (usuario - registrado)	10d	12/12/2012	25/12/2012		
31		Consultar perfil (Módulo 2)	5d	12/12/2012	18/12/2012		
32		Test Módulo 2	5d	19/12/2012	25/12/2012		
33		⊖Módulo 3 ( hincha - registrado)	25d	19/12/2012	22/01/2013		
34		Crear hincha (Módulo 3)	8d	19/12/2012	28/12/2012		
35		Consultar hincha (Módulo 3)	8d	19/12/2012	28/12/2012		
36		Acceder árbol habilidades (Módulo 3)	7d	31/12/2012	08/01/2013		
37		Asignar habilidades (Módulo 3)	10d	31/12/2012	11/01/2013		
38		Ejecutar habilidades (Módulo 3)	8d	07/01/2013	16/01/2013		
39		Test Módulo 3	5d	16/01/2013	22/01/2013		
40		⊖Módulo 4 (afición - registrado)	43d	12/12/2012	08/02/2013		
41		Crear afición (Módulo 4)	4.5d	12/12/2012	18/12/2012		
42		Consultar afición (Módulo 4)	4.38d	12/12/2012	18/12/2012		
43		Unirse a afición (Módulo 4)	7d	31/12/2012	08/01/2013		
44		salirse de la afición (Módulo 4)	7d	31/12/2012	08/01/2013		
45		Ver acciones abiertas (Módulo 4)	8d	24/01/2013	04/02/2013		
46		Participar en acciones (Módulo 4)	8d	24/01/2013	04/02/2013		
47		Test Módulo 4	4d	05/02/2013	08/02/2013		
48		⊖Módulo 5 (partido - registrado)	17d	24/01/2013	15/02/2013		
49		Consultar previo de partido (Módulo 5)	8d	24/01/2013	04/02/2013		
50		Asistir a partido (Módulo 5)	8d	24/01/2013	04/02/2013		
51		Participar en partido - MOTOR (Módulo 5)	12d	24/01/2013	08/02/2013		
52		Test Módulo 5	5d	11/02/2013	15/02/2013		
53		⊖Módulo 6 (liga - administrador)	12d	07/01/2013	22/01/2013		
54		Administrar la liga (Módulo 6)	8d	07/01/2013	16/01/2013		
55		Administrar equipos (Módulo 6)	8d	07/01/2013	16/01/2013		
56		Test Módulo 6	5d	16/01/2013	22/01/2013		
57		Test de la arquitectura	5d	18/02/2013	22/02/2013	29,32,39,47,52,5	
58		⊖Fase Construcción	55d?	04/03/2013	17/05/2013	57	
59		Primera iteración construcción	15d?	04/03/2013	22/03/2013		
60		Segunda iteración construcción	15d?	25/03/2013	12/04/2013		
61		Tercera iteración construcción	15d?	15/04/2013	03/05/2013		
62		Iteración final	5d?	06/05/2013	10/05/2013		
63		Semana contingencias	5d?	13/05/2013	17/05/2013		
64		⊖Fase Transición	6d?	20/05/2013	27/05/2013		
65		Preparar presentación	6d?	20/05/2013	27/05/2013		

JUGADOR #12

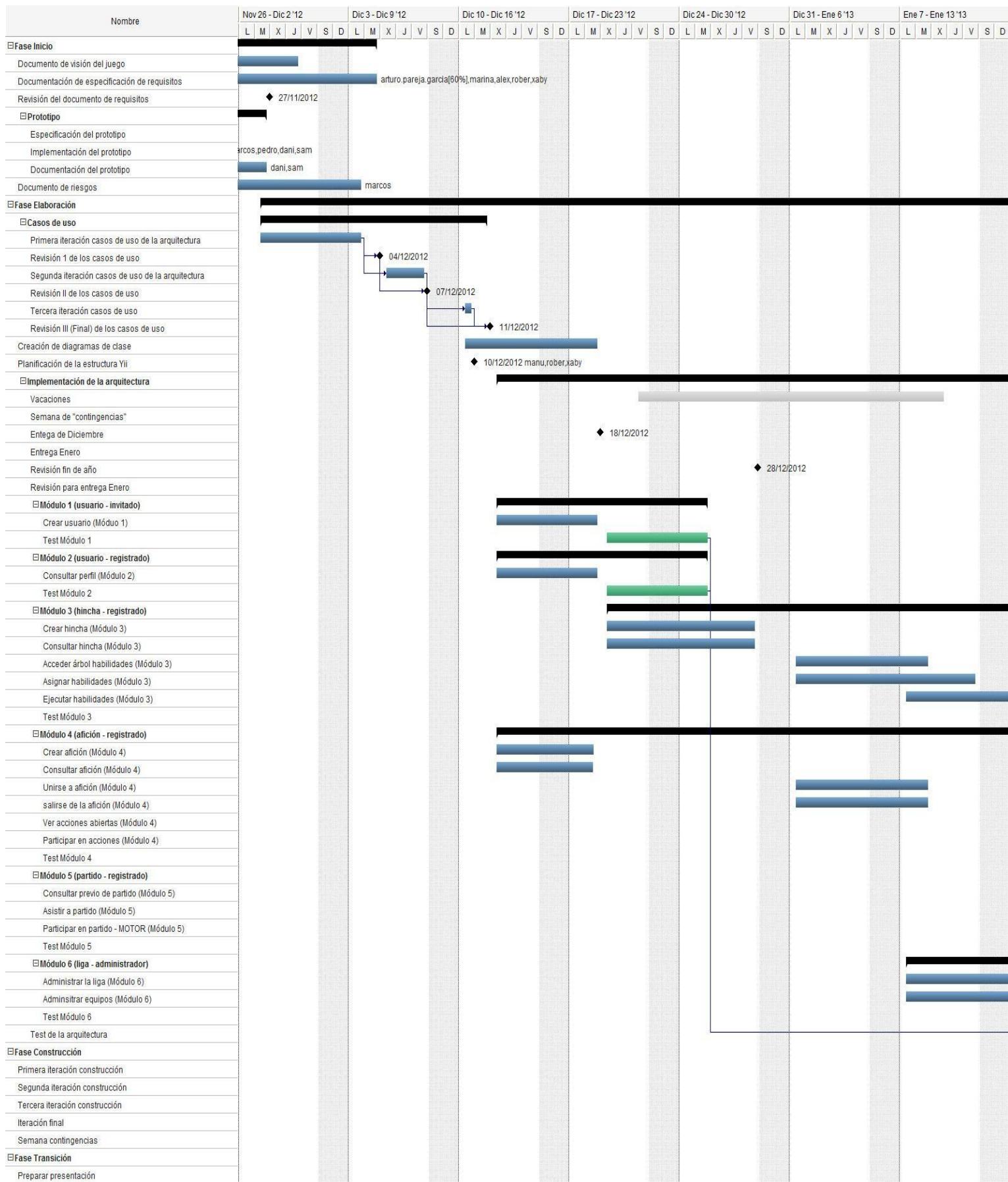
## Documentación de organización: organización y reparto de tareas

Ingeniería del Software, 2013

JUGADOR #12

## Documentación de organización: organización y reparto de tareas

Ingeniería del Software, 2013



# JUGADOR #12

## Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

Nombre	Ene 14 - Ene 20 '13		Ene 21 - Ene 27 '13		Ene 28 - Feb 3 '13		Feb 4 - Feb 10 '13		Feb 11 - Feb 17 '13		Feb 18 - Feb 24 '13		Feb 25 - Mar 3 '13															
	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D
■ Fase Inicio																												
Documento de visión del juego																												
Documentación de especificación de requisitos																												
Revisión del documento de requisitos																												
■ Prototipo																												
Especificación del prototipo																												
Implementación del prototipo																												
Documentación del prototipo																												
Documento de riesgos																												
■ Fase Elaboración																												
■ Casos de uso																												
Primerá iteración casos de uso de la arquitectura																												
Revisión 1 de los casos de uso																												
Segunda iteración casos de uso de la arquitectura																												
Revisión II de los casos de uso																												
Tercera iteración casos de uso																												
Revisión III (Final) de los casos de uso																												
Creación de diagramas de clase																												
Planificación de la estructura Yii																												
■ Implementación de la arquitectura																												
Vacaciones																												
Semana de "contingencias"																												
Entrega de Diciembre																												
Entrega Enero																												
Revisión fin de año																												
Revisión para entrega Enero																												
■ Módulo 1 (usuario - invitado)																												
Crear usuario (Módulo 1)																												
Test Módulo 1																												
■ Módulo 2 (usuario - registrado)																												
Consultar perfil (Módulo 2)																												
Test Módulo 2																												
■ Módulo 3 ( hincha - registrado)																												
Crear hincha (Módulo 3)																												
Consultar hincha (Módulo 3)																												
Acceder árbol habilidades (Módulo 3)																												
Asignar habilidades (Módulo 3)																												
Ejecutar habilidades (Módulo 3)																												
Test Módulo 3																												
■ Módulo 4 (afición - registrado)																												
Crear afición (Módulo 4)																												
Consultar afición (Módulo 4)																												
Unirse a afición (Módulo 4)																												
salirse de la afición (Módulo 4)																												
Ver acciones abiertas (Módulo 4)																												
Participar en acciones (Módulo 4)																												
Test Módulo 4																												
■ Módulo 5 (partido - registrado)																												
Consultar previo de partido (Módulo 5)																												
Asistir a partido (Módulo 5)																												
Participar en partido - MOTOR (Módulo 5)																												
Test Módulo 5																												
■ Módulo 6 (liga - administrador)																												
Administrar la liga (Módulo 6)																												
Administrar equipos (Módulo 6)																												
Test Módulo 6																												
Test de la arquitectura																												
■ Fase Construcción																												
Primera iteración construcción																												
Segunda iteración construcción																												
Tercera iteración construcción																												
Iteración final																												
Semana contingencias																												
■ Fase Transición																												
Preparar presentación																												

JUGADOR #12

## Documentación de organización: organización y reparto de tareas

Ingeniería del Software, 2013

JUGADOR #12

## Documentación de organización: organización y reparto de tareas

Ingeniería del Software, 2013

# JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

## 5. Reparto de tareas y seguimiento semanal

A pesar de la existencia de una tabla en Google Docs con esta información, se adjunta al presente documento para mejorar la visualización y localización de la misma.

### 5.1. 22 al 28 de octubre

- Commits: 0
- Número total de horas:
- Descripción: fase de brainstorming inicial. El grupo decide y redacta los aspectos más básicos y superficiales del juego.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"><li>• Brainstorming general.</li><li>• Colaboración en la redacción de los documentos iniciales. En particular un documento para definir el funcionamiento y ámbito de los recursos del juego</li></ul>	6.7
Arturo Pareja	<ul style="list-style-type: none"><li>• Brainstorming general.</li><li>• Colaboración en la redacción de los documentos iniciales</li></ul>	4
Daniel Escoz	<ul style="list-style-type: none"><li>• Brainstorming general.</li><li>• Colaboración en la redacción de los documentos iniciales</li></ul>	5
Roberto Marín	<ul style="list-style-type: none"><li>• Brainstorming general.</li><li>• Colaboración en la redacción de los documentos iniciales</li></ul>	5
Alexandra Martín	<ul style="list-style-type: none"><li>• Brainstorming general.</li><li>• Colaboración en la redacción de los documentos iniciales</li></ul>	4
Javier López	<ul style="list-style-type: none"><li>• Brainstorming general.</li><li>• Colaboración en la redacción de los documentos iniciales</li></ul>	6.5
Manuel Artero	<ul style="list-style-type: none"><li>• Brainstorming general.</li><li>• Colaboración en la redacción de los documentos iniciales</li></ul>	7
Marcos Alarcón	<ul style="list-style-type: none"><li>• Brainstorming general.</li><li>• Colaboración en la redacción de los documentos iniciales</li></ul>	4
Pedro Morgado	<ul style="list-style-type: none"><li>• Brainstorming general.</li><li>• Colaboración en la redacción de los documentos iniciales</li></ul>	5
Samuel Méndez	<ul style="list-style-type: none"><li>• Brainstorming general.</li><li>• Colaboración en la redacción de los documentos iniciales</li></ul>	3

## JUGADOR #12

Documentación de organización: organización y reparto de tareas      Ingeniería del Software, 2013

### 5.2. 29 de octubre al 4 de noviembre

- Commits: 0
- Número total de horas:
- Descripción: presentación del primer lote de documentos al profesor y refinamiento de los mismos.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>● Brainstorming general.</li> <li>● Continuación de los documentos básicos</li> <li>● Creación de una estructura del drive para trabajar</li> <li>● Creación de las guías de estilo para la documentación</li> </ul>	10
Arturo Pareja	<ul style="list-style-type: none"> <li>● Brainstorming general.</li> <li>● Continuación de los documentos básicos</li> <li>● Diseño del mockup inicial del producto</li> </ul>	5
Daniel Escoz	<ul style="list-style-type: none"> <li>● Brainstorming general.</li> <li>● Búsqueda de información y argumentación a favor del uso del framework Symfony/Sílex</li> </ul>	7
Roberto Marín	<ul style="list-style-type: none"> <li>● Brainstorming general.</li> <li>● Continuación de los documentos básicos</li> </ul>	4
Alexandra Martín	<ul style="list-style-type: none"> <li>● Brainstorming general.</li> <li>● Continuación de los documentos básicos</li> </ul>	5
Javier López	<ul style="list-style-type: none"> <li>● Brainstorming general.</li> <li>● Búsqueda de información y argumentación a favor del uso del framework Yii</li> <li>● Diseño del mockup inicial del producto</li> </ul>	7
Manuel Artero	<ul style="list-style-type: none"> <li>● Brainstorming general.</li> <li>● Documentación; en especial documento de definición del juego</li> <li>● Estudio de la fase de inicio <ul style="list-style-type: none"> <li>○ Estudio de comparativas de los diferentes frameworks</li> </ul> </li> </ul>	7
Marcos Alarcón	<ul style="list-style-type: none"> <li>● Brainstorming general.</li> <li>● Continuación de los documentos básicos</li> <li>● Especificación de habilidades y acciones</li> </ul>	6
Pedro Morgado	<ul style="list-style-type: none"> <li>● Brainstorming general.</li> <li>● Continuación de los documentos básicos</li> </ul>	4
Samuel Méndez	<ul style="list-style-type: none"> <li>● Brainstorming general.</li> <li>● Continuación de los documentos básicos</li> </ul>	4

### 5.3. 5 al 11 de noviembre

- Commits: 0
- Número total de horas:

## JUGADOR #12

Documentación de organización: organización y reparto de tareas      Ingeniería del Software, 2013

- Descripción: se toma la decisión de usar Yii como framework y GitHub como repositorio. Se deja la semana para el aprendizaje de las tecnologías y frameworks a emplear. También se comienza el desarrollo del prototipo a presentar.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>● Instrucción en las nuevas tecnologías</li> <li>● Creación del juego de cartas <ul style="list-style-type: none"> <li>○ Creación y especificación inicial de las cartas</li> <li>○ Reunión con los compañeros del equipo para probarlas (Pedro morgado y Roberto Marín)</li> <li>○ Reunión por steam para equilibrar los recursos</li> </ul> </li> </ul>	8
Arturo Pareja	<ul style="list-style-type: none"> <li>● Instrucción en las nuevas tecnologías</li> <li>● Continuación del mockup</li> </ul>	6
Daniel Escoz	<ul style="list-style-type: none"> <li>● Preparación de tutoriales sobre el repositorio</li> </ul>	6-7
Roberto Marín	<ul style="list-style-type: none"> <li>● Instrucción en las nuevas tecnologías</li> </ul>	6
Alexandra Martín	<ul style="list-style-type: none"> <li>● Instrucción en las nuevas tecnologías</li> </ul>	7
Javier López	<ul style="list-style-type: none"> <li>● Preparación de tutoriales sobre XAMPP y Yii</li> <li>● Continuación del mockup</li> </ul>	8
Manuel Artero	<ul style="list-style-type: none"> <li>● Instrucción en las nuevas tecnologías</li> <li>● Ayuda en la preparación de tutoriales de Yii y Git</li> </ul>	8
Marcos Alarcón	<ul style="list-style-type: none"> <li>● Instrucción en las nuevas tecnologías</li> </ul>	6
Pedro Morgado	<ul style="list-style-type: none"> <li>● Instrucción en las nuevas tecnologías</li> </ul>	7
Samuel Méndez	<ul style="list-style-type: none"> <li>● Instrucción en las nuevas tecnologías</li> <li>● Juego de Cartas (SimuMatch) en Java</li> </ul>	7

### 5.4. 12 al 18 de noviembre

- Commits:
- Número total de horas:
- Descripción: se continúa trabajando en el prototipo, al que se adjunta el juego de cartas. También se dan varias clases de instrucción en las nuevas tecnologías con el fin de partir de una base común.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>● Redactar el documento de especificación de requisitos</li> <li>● Redactar el documento de definición del juego</li> </ul>	6
Arturo Pareja	<ul style="list-style-type: none"> <li>● Redactar el documento de especificación de</li> </ul>	8

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

	<ul style="list-style-type: none"> <li>requisitos</li> <li>• Redactar el documento de definición del juego</li> </ul>	
Daniel Escoz	<ul style="list-style-type: none"> <li>• Preparación e impartición de la clase de Git/GitHub</li> <li>• Implementación de parte del simulador SimuMatch</li> </ul>	9
Roberto Marín	<ul style="list-style-type: none"> <li>• Realización de las cartas para el juego. Incluye poner sus efectos y el gasto de recursos de la carta</li> </ul>	6
Alexandra Martín	<ul style="list-style-type: none"> <li>• Implementación de SimuMatch (Simulador Java)</li> </ul>	14
Javier López	<ul style="list-style-type: none"> <li>• Comienzo del diseño inicial de la base de datos</li> </ul>	7
Manuel Artero	<ul style="list-style-type: none"> <li>• Tareas de gestión del grupo de desarrollo del simulador y coordinación con el grupo de documentación</li> </ul>	7
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Implementación del Simulador Java</li> </ul>	18
Pedro Morgado	<ul style="list-style-type: none"> <li>• Realización del juego de cartas.</li> </ul>	6
Samuel Méndez	<ul style="list-style-type: none"> <li>• Juego de Cartas (SimuMatch) en Java</li> <li>• Instrucción en las nuevas tecnologías</li> </ul>	7

### 5.5. 19 al 25 de noviembre

- Commits:
- Número total de horas:
- Descripción: continuación de la documentación necesaria antes de comenzar la fase de elaboración. Creación de una planificación global en Gantter.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Redacción de las reglas del juego de cartas</li> <li>• Refinamiento en las redacciones de los documentos de especificación y definición del juego</li> </ul>	7
Arturo Pareja	<ul style="list-style-type: none"> <li>• Generación de la planificación global</li> <li>• Documento de previsión de riesgos</li> </ul>	4
Daniel Escoz	<ul style="list-style-type: none"> <li>• </li> </ul>	
Roberto Marín	<ul style="list-style-type: none"> <li>• Documento de especificación de requisitos software</li> </ul>	4
Alexandra Martín	<ul style="list-style-type: none"> <li>• Estudio de casos de uso</li> </ul>	4
Javier López	<ul style="list-style-type: none"> <li>• Documento de especificación de requisitos software</li> <li>• Continuación del diseño de la base de datos</li> </ul>	8
Manuel Artero	<ul style="list-style-type: none"> <li>• Generación de la planificación global           <ul style="list-style-type: none"> <li>◦ Documento de planificación de Gantter</li> </ul> </li> <li>• Apoyo a la generación de la documentación</li> </ul>	7
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Documento de previsión de riesgos</li> </ul>	6

## JUGADOR #12

**Documentación de organización: organización y reparto de tareas** Ingeniería del Software, 2013

Pedro Morgado	<ul style="list-style-type: none"> <li>Apoyo con la documentación</li> </ul>	4
Samuel Méndez	<ul style="list-style-type: none"> <li>Documentación sobre el prototipo en Java</li> <li>Apoyo a la generación de la documentación</li> </ul>	4

### 5.6. 26 de noviembre al 2 de diciembre

- Commits: 7
- Número total de horas:
- Descripción: se perfilan los documentos y se eliminan las peñas del producto a petición del profesor. Se comienzan a detallar los primeros casos de uso.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>Apoyo a la generación de la documentación</li> </ul>	2
Arturo Pareja	<ul style="list-style-type: none"> <li>Redacción de casos de uso</li> </ul>	8
Daniel Escoz	<ul style="list-style-type: none"> <li>Redacción de casos de uso</li> </ul>	5
Roberto Marín	<ul style="list-style-type: none"> <li>Revisión de documentos eliminando todo lo que hablará de peñas y ajustando estos documentos</li> </ul>	4
Alexandra Martín	<ul style="list-style-type: none"> <li>Apoyo a la generación de la documentación</li> </ul>	3
Javier López	<ul style="list-style-type: none"> <li>Finalización del diseño de la base de datos inicial</li> <li>Construcción del esqueleto de la aplicación con un sistema de login básico</li> </ul>	7
Manuel Artero	<ul style="list-style-type: none"> <li>Revisión de la documentación generada hasta el momento</li> <li>Diseño de los estados del partido</li> <li>Implementación del esqueleto de la aplicación.</li> </ul>	11
Marcos Alarcón	<ul style="list-style-type: none"> <li>Modelos, debug, y revision de documentacion</li> </ul>	9
Pedro Morgado	<ul style="list-style-type: none"> <li>Redacción de casos de uso</li> </ul>	5
Samuel Méndez	<ul style="list-style-type: none"> <li>Casos de uso</li> <li>Documentación sobre resumen del juego</li> </ul>	4

### 5.7. 3 al 9 de diciembre

- Commits: 18
- Número total de horas:
- Descripción: esta semana se dejó libre para que los miembros del grupo pudiesen ponerse al día por completo en todas las prácticas de otras asignaturas. Se comenzó también el diseño de la estructura básica de la aplicación, cuyo fin fue agilizar en gran medida la primera fase de elaboración.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>Apoyo a la generación de la documentación</li> </ul>	2

## JUGADOR #12

**Documentación de organización: organización y reparto de tareas** Ingeniería del Software, 2013

Arturo Pareja	<ul style="list-style-type: none"> <li>• Redacción de casos de uso</li> <li>• Revisión del documento de especificación de requisitos</li> </ul>	6
Daniel Escoz	<ul style="list-style-type: none"> <li>• Aprendizaje inicial de las tecnologías a usar (Yii, etc.)</li> </ul>	3
Roberto Marín	<ul style="list-style-type: none"> <li>• Revisión de la base de datos inicial</li> <li>• Diseño de la estructura básica de la primera iteración de la aplicación.</li> </ul>	8
Alexandra Martín	<ul style="list-style-type: none"> <li>• Estudio de las nuevas tecnologías.</li> </ul>	9
Javier López	<ul style="list-style-type: none"> <li>• Diseño de la estructura básica de la primera iteración de la aplicación.</li> <li>• Revisión de la base de datos inicial</li> </ul>	9
Manuel Artero	<ul style="list-style-type: none"> <li>• Trabajo mínimo en la asignatura como apoyo en el documento de gestión de riesgos</li> </ul>	2
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Revisión de la base de datos inicial</li> <li>• Debug del código.</li> </ul>	5
Pedro Morgado	<ul style="list-style-type: none"> <li>• Aprendizaje de las nuevas tecnologías</li> </ul>	5
Samuel Méndez	<ul style="list-style-type: none"> <li>• Aprendiendo Yii a través de un tutorial</li> <li>• Documentación casos de uso</li> </ul>	6

### 5.8. 10 al 16 de diciembre

- Commits: 18
- Número total de horas:
- Descripción: al comienzo de la semana se finalizó por completo la estructura básica y reparto de tareas. Aquí comenzó la primera iteración de la fase de elaboración.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Apoyo a la generación de la documentación</li> </ul>	1
Arturo Pareja	<ul style="list-style-type: none"> <li>• Revisión de los casos de uso realizados anteriormente</li> </ul>	8
Daniel Escoz	<ul style="list-style-type: none"> <li>• Revisión de los casos de uso realizados anteriormente</li> </ul>	4
Roberto Marín	<ul style="list-style-type: none"> <li>• Realización de las relaciones en el modelo Usuarios</li> <li>• Realización de acciones en el controlador de Usuarios</li> </ul>	10
Alexandra Martín	<ul style="list-style-type: none"> <li>• Revisión de documentación.</li> </ul>	3
Javier López	<ul style="list-style-type: none"> <li>• Finalización estructura básica</li> <li>• Testeo de la estructura inicial (login)</li> <li>• Apoyo a los miembros y revisión del código a implementar durante la semana</li> </ul>	10

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

Manuel Artero	<ul style="list-style-type: none"> <li>Finalización estructura básica</li> <li>Reparto de tareas de la primera iteración</li> <li>Gestión del grupo: comienzo de la programación</li> </ul>	9
Marcos Alarcón	<ul style="list-style-type: none"> <li>Estudio de tutoriales de Yii</li> </ul>	5
Pedro Morgado	<ul style="list-style-type: none"> <li>Aprendizaje de las nuevas tecnologías</li> </ul>	5
Samuel Méndez	<ul style="list-style-type: none"> <li>Aprendiendo Yii a través de un tutorial</li> </ul>	3

### 5.9. 17 al 23 de diciembre

- Commits: 57
- Número total de horas:
- Descripción: durante esta semana se revisó y corrigió la estructura de la base de datos. Se continuó la implementación de los primeros módulos y se comenzó el diseño de los siguientes.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>Vacaciones de navidad</li> </ul>	0
Arturo Pareja	<ul style="list-style-type: none"> <li>Vacaciones de navidad</li> </ul>	0
Daniel Escoz	<ul style="list-style-type: none"> <li>Controlador de Acciones</li> </ul>	4
Roberto Marín	<ul style="list-style-type: none"> <li>Realización de acciones para cambiar la contraseña y el email</li> </ul>	7
Alexandra Martín	<ul style="list-style-type: none"> <li>Estudio tutoriales y modelos.</li> </ul>	4
Javier López	<ul style="list-style-type: none"> <li>Apoyo a los miembros y revisión del código a implementar durante la semana</li> <li>Diseño de las acciones e implementación de algunas de ejemplo</li> <li>Diseño del sistema de acciones y partidos</li> </ul>	9
Manuel Artero	<ul style="list-style-type: none"> <li>Vacaciones de navidad</li> </ul>	0
Marcos Alarcón	<ul style="list-style-type: none"> <li>Código de habilidades</li> <li>Definidas relaciones en los modelos</li> </ul>	10
Pedro Morgado	<ul style="list-style-type: none"> <li>Controlador de Acciones</li> <li>Modelo Acciones individuales</li> </ul>	5
Samuel Méndez	<ul style="list-style-type: none"> <li>Vistas y modelos</li> </ul>	8

### 5.10. 24 al 30 de diciembre

- Commits: 70
- Número total de horas:
- Descripción: continuación de la implementación anterior durante las vacaciones.

## JUGADOR #12

Documentación de organización: organización y reparto de tareas      Ingeniería del Software, 2013

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Función del controlador de usuarios y su vista. Perfil(Solo una pero la primera vez lleva mucho tiempo adaptarse)</li> </ul>	7
Arturo Pareja	<ul style="list-style-type: none"> <li>• Estudio de tutoriales de Yii</li> </ul>	5
Daniel Escoz	<ul style="list-style-type: none"> <li>• Finalización de controlador Acciones y modelo Participaciones</li> </ul>	6
Roberto Marín	<ul style="list-style-type: none"> <li>• Corrección de errores en los modelos</li> </ul>	2
Alexandra Martín	<ul style="list-style-type: none"> <li>• Modelos y relaciones, vistas.</li> <li>• Debug.</li> </ul>	9
Javier López	<ul style="list-style-type: none"> <li>• Apoyo a los miembros y revisión del código a implementar durante la semana</li> <li>• Diseño de las acciones e implementación de algunas de ejemplo</li> <li>• Revisión continua y adecuación de la base de datos</li> </ul>	7
Manuel Artero	<ul style="list-style-type: none"> <li>• Planificación de la nueva iteración del proyecto</li> </ul>	6
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Implementación de acciones grupales</li> </ul>	12
Pedro Morgado	<ul style="list-style-type: none"> <li>• Controlador Acciones</li> <li>• Modelo Acciones Individuales</li> </ul>	7
Samuel Méndez	<ul style="list-style-type: none"> <li>• Nada</li> </ul>	0

### 5.11. 31 de diciembre al 6 de enero

- Commits: 117
- Número total de horas:
- Descripción: continuación de la implementación asignada para las vacaciones.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Controlador de usuarios y su vista (ver jugador)</li> <li>• Controlador de equipos y su vista (ver equipo)</li> <li>• Controlador de partidos y su vista (ver previa)</li> </ul>	8
Arturo Pareja	<ul style="list-style-type: none"> <li>• Controlador asistir y su vista</li> <li>• Controlador de partidos y vista index</li> </ul>	6
Daniel Escoz	<ul style="list-style-type: none"> <li>• Implementación y diseño de LA FÓRMULA</li> </ul>	8-10
Roberto Marín	<ul style="list-style-type: none"> <li>• Realización del Helper</li> <li>• Realización de las acciones que me asignaron</li> </ul>	8
Alexandra Martín	<ul style="list-style-type: none"> <li>• Debug y limpieza de código.</li> </ul>	5
Javier López	<ul style="list-style-type: none"> <li>• Apoyo a los miembros y revisión del código a implementar durante la semana</li> </ul>	6

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

	<ul style="list-style-type: none"> <li>• Diseño de las acciones e implementación de algunas de ejemplo</li> <li>• Revisión continua y adecuación de la base de datos</li> </ul>	
Manuel Artero	<ul style="list-style-type: none"> <li>• Integración del código generado: debug.</li> <li>• Apoyo a los miembros y revisión del código a implementar durante la semana</li> <li>• Diseño de las acciones</li> <li>• Planificación de la nueva iteración del proyecto</li> </ul>	12
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Debug y limpieza de código</li> <li>• Actualizados comentarios por todos el código</li> </ul>	5
Pedro Morgado	<ul style="list-style-type: none"> <li>• Implementación y diseño de LA FÓRMULA</li> </ul>	7
Samuel Méndez	<ul style="list-style-type: none"> <li>• Acciones y Fixtures (PHPUnit)</li> </ul>	6

### 5.12. 7 al 13 de enero

- Commits: 84
- Número total de horas:
- Descripción: revisión y unión de todo lo implementado durante las navidades. Nuevo reparto de tareas centrado en la entrega del día 22 de enero.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Función contratar RRPP (funcionalidad)</li> <li>• CSS: Nueva pantalla de inicio (Esto me llevó una cantidad de horas brutal. Coges CSS por primera vez es difícil )</li> <li>• CSS: Rediseño del layout Main</li> </ul>	17
Arturo Pareja	<ul style="list-style-type: none"> <li>• Revisión del modelo Habilidades</li> <li>• Action adquirir del controlador de Habilidades y vista</li> </ul>	9
Daniel Escoz	<ul style="list-style-type: none"> <li>• Revisión y ampliación de LA FÓRMULA</li> </ul>	7
Roberto Marín	<ul style="list-style-type: none"> <li>• Realización del script que ejecuta los turnos de un partido</li> <li>• Corrección de errores en las funciones de la clase Helper</li> </ul>	6
Alexandra Martín	<ul style="list-style-type: none"> <li>• Funciones de partido.</li> <li>• Mejoras en vistas.</li> <li>• Estudio testing con Yii.</li> </ul>	10
Javier López	<ul style="list-style-type: none"> <li>• Revisión completa, testeo e identificación de fallos en todo el código implementado a lo largo de las navidades</li> </ul>	9
Manuel Artero	<ul style="list-style-type: none"> <li>• Diseño de la nueva itearción. Reparto de tareas, y documentación añadida.</li> </ul>	12
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Implementación de las bonificaciones</li> <li>• Redo del código de participaciones</li> </ul>	11

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

	<ul style="list-style-type: none"> <li>• Añadido formulario de confirmación</li> </ul>	
Pedro Morgado	<ul style="list-style-type: none"> <li>• Inicio del estilo de la página</li> </ul>	8
Samuel Méndez	<ul style="list-style-type: none"> <li>• Revisión de Modelos</li> </ul>	4

### 5.13. 14 al 20 de enero

- Commits: 47
- Número total de horas:
- Descripción: continuación de la primera iteración de la fase de elaboración. Todo centrado en la entrega del día 22.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• CSS: Vista perfil usuario y su afición</li> <li>• CSS general de la página</li> </ul>	6
Arturo Pareja	<ul style="list-style-type: none"> <li>• Revisión del controlador de Acciones</li> <li>• Implementación de usar Acciones</li> </ul>	10
Daniel Escoz	<ul style="list-style-type: none"> <li>• Apoyo general a compañeros, tanto en documentación como en implementación</li> </ul>	4
Roberto Marín	<ul style="list-style-type: none"> <li>• Realización de código para cambiar de equipo.</li> </ul>	5
Alexandra Martín	<ul style="list-style-type: none"> <li>• Registro.</li> <li>• Limpieza de código y debug.</li> </ul>	7
Javier López	<ul style="list-style-type: none"> <li>• Revisión continua del código implementado.</li> <li>• Apoyo en los diferentes módulos a implementar</li> <li>• Garantizar correcto funcionamiento de los mismos (debugging)</li> </ul>	7
Manuel Artero	<ul style="list-style-type: none"> <li>• Limpieza total de la documentación</li> <li>• Preparación de la documentación</li> </ul>	6
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Implementación de las clasificaciones de liga</li> <li>• Limpieza de código</li> </ul>	10
Pedro Morgado	<ul style="list-style-type: none"> <li>• Estilo general de las vistas con CSS</li> </ul>	7
Samuel Méndez	<ul style="list-style-type: none"> <li>• Revisión de Modelos</li> </ul>	4

### 5.14. 21 al 27 de enero

- Commits: 0
- Número total de horas:
- Descripción: presentación del avance del proyecto. Primera iteración de módulos finalizada.

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>Retoques para la entrega</li> </ul>	2
Arturo Pareja	<ul style="list-style-type: none"> <li>Inicio del sistema de issues del proyecto</li> <li>Testeo de la aplicación a fondo para sacar todas las issues posibles.</li> <li>Primeras 14 issues solucionadas.</li> </ul>	12
Daniel Escoz	<ul style="list-style-type: none"> <li>Entrega de otras asignaturas - sólo clase</li> </ul>	3
Roberto Marín	<ul style="list-style-type: none"> <li>Realizado código para actualizar la clasificación y repartir las bonificaciones</li> </ul>	6
Alexandra Martín	<ul style="list-style-type: none"> <li>Estilo (CSS)</li> <li>Investigar cron.</li> <li>Script para aumentar recursos.</li> </ul>	8
Javier López	<ul style="list-style-type: none"> <li>Revisión y debugging de la implementación realizada</li> <li>Diseño de los nuevos módulos</li> </ul>	8
Manuel Artero	<ul style="list-style-type: none"> <li>Revisión y debugging de la implementación realizada</li> <li>Diseño de los nuevos módulos</li> </ul>	5
Marcos Alarcón	<ul style="list-style-type: none"> <li>Añadido aviso de ban en el log de la aplicación</li> <li>Revisión de la robustez del código</li> </ul>	7
Pedro Morgado	<ul style="list-style-type: none"> <li>Estilo general de las vistas con CSS</li> </ul>	4
Samuel Méndez	<ul style="list-style-type: none"> <li>Documentación</li> </ul>	2

### 5.15. 28 de enero al 3 de febrero

- Commits: 4
- Número total de horas:
- Descripción: semana de parón general por exámenes.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>Exámenes de febrero</li> </ul>	0
Arturo Pareja	<ul style="list-style-type: none"> <li>Exámenes de febrero</li> </ul>	0
Daniel Escoz	<ul style="list-style-type: none"> <li>Exámenes</li> </ul>	0
Roberto Marín		
Alexandra Martín	<ul style="list-style-type: none"> <li>Exámenes</li> </ul>	0
Javier López	<ul style="list-style-type: none"> <li>Mínimo debugging sobre el sistema</li> </ul>	3
Manuel Artero	<ul style="list-style-type: none"> <li>Exámenes de febrero</li> </ul>	0

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

Marcos Alarcón	<ul style="list-style-type: none"> <li>• Botón de participaciones múltiples</li> <li>• Arreglo en la transacciones de las acciones</li> <li>• Añadidos js para simular los flashes</li> </ul>	6
Pedro Morgado	<ul style="list-style-type: none"> <li>• Exámenes de febrero</li> </ul>	0
Samuel Méndez	<ul style="list-style-type: none"> <li>• Exámenes de febrero</li> </ul>	0

### 5.16. 4 al 10 de febrero

- Commits: 2
- Número total de horas:
- Descripción: semana de exámenes, parón bastante generalizado.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Exámenes de febrero</li> </ul>	0
Arturo Pareja	<ul style="list-style-type: none"> <li>• Exámenes de febrero</li> </ul>	0
Daniel Escoz	<ul style="list-style-type: none"> <li>• Exámenes</li> </ul>	0
Roberto Marín		
Alexandra Martín	<ul style="list-style-type: none"> <li>• Exámenes</li> </ul>	
Javier López	<ul style="list-style-type: none"> <li>• Revisión y corrección de las relaciones y atributos de los modelos</li> <li>• Incorporación de las “foreign key” en la base de datos</li> <li>• Intento de incorporación de PHPUnit como método de testing</li> </ul>	7
Manuel Artero	<ul style="list-style-type: none"> <li>• Exámenes de febrero</li> </ul>	0
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Testeo de la página</li> <li>• Estudio proyectos similares para la documentación</li> </ul>	10
Pedro Morgado	<ul style="list-style-type: none"> <li>• Exámenes de febrero</li> </ul>	0
Samuel Méndez	<ul style="list-style-type: none"> <li>• Revisión documentación</li> </ul>	2

### 5.17. 11 al 17 de febrero

- Commits: 51
- Número total de horas:
- Descripción: última semana de parón por exámenes. Revisión del módulo de partido para garantizar su funcionamiento.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Vuelta a clase.</li> <li>• Preparación del partido</li> </ul>	2

## JUGADOR #12

**Documentación de organización: organización y reparto de tareas** Ingeniería del Software, 2013

Arturo Pareja	<ul style="list-style-type: none"> <li>• Reparación de las últimas issues hasta la fecha</li> <li>• Renovación de la lista de issues con nuevos bugs</li> </ul>	5
Daniel Escoz	<ul style="list-style-type: none"> <li>• Final de exámenes</li> <li>• Investigación de CSS/LESS</li> </ul>	2
Roberto Marín		
Alexandra Martín		
Javier López	<ul style="list-style-type: none"> <li>• Revisión y corrección de la clase de Partido para su correcto funcionamiento a la vuelta de exámenes</li> </ul>	7
Manuel Artero	<ul style="list-style-type: none"> <li>• Planteamiento del módulo arte del proyecto</li> <li>• Búsqueda de artista externo para desarrollar el arte</li> </ul>	11
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Problemas con la versión subida</li> <li>• Búsqueda de otro host</li> </ul>	7
Pedro Morgado	<ul style="list-style-type: none"> <li>• Estilo general de las vistas con CSS</li> </ul>	7
Samuel Méndez	<ul style="list-style-type: none"> <li>• CSS y LESS por primera vez</li> </ul>	6

### 5.18. 18 al 24 de febrero

- Commits: 228
- Número total de horas:
- Descripción: se comienza la implementación de nuevos módulos al mismo tiempo que se perfilan los anteriores. Inicio del estilo visual del producto.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Función generar crónica (Partido)</li> <li>• Función genera estado (Partido)</li> <li>• CSS: Aspectos visuales de la vista asistir partido</li> </ul>	8
Arturo Pareja	<ul style="list-style-type: none"> <li>• Investigación de CSS/LESS</li> <li>• Solucionadas un par de issues</li> </ul>	4
Daniel Escoz	<ul style="list-style-type: none"> <li>• Comienzo del diseño de La Barra.</li> </ul>	5
Roberto Marín	<ul style="list-style-type: none"> <li>• Corrección de errores en las funciones de actualización de la clasificación y de bonificaciones.</li> </ul>	4
Alexandra Martín		
Javier López	<ul style="list-style-type: none"> <li>• Revisión continua del módulo de partido para garantizar su correcto funcionamiento</li> <li>• Generación de datos iniciales para el testeo de partidos</li> <li>• Comienzo de la implementación de las acciones existentes</li> </ul>	14
Manuel Artero	<ul style="list-style-type: none"> <li>• Tareas generales de gestión del grupo</li> <li>• Estar con un ojo a todos los grupos de desarrollo</li> </ul>	7

## JUGADOR #12

**Documentación de organización: organización y reparto de tareas** Ingeniería del Software, 2013

Marcos Alarcón	<ul style="list-style-type: none"> <li>• Merging y resolución de conflictos de las ramas olvidadas</li> <li>• Preparación del proyecto para subirlo al servidor</li> </ul>	12
Pedro Morgado	<ul style="list-style-type: none"> <li>• Estilo general de la página con CSS</li> </ul>	5
Samuel Méndez	<ul style="list-style-type: none"> <li>• Documentación diagramas UML de prototipo Java y del proyecto en Yii (se descartaron)</li> <li>• Estilo visual de la página</li> </ul>	12

### 5.19. 25 de febrero al 3 de marzo

- Commits: 135
- Número total de horas:
- Descripción: se continúa la implementación de nuevos módulos y se centra todo en la demo del 20 de marzo.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Función generar crónica (Partido)</li> <li>• Función genera estado (Partido)</li> <li>• CSS: Aspectos visuales de la vista asistir partido</li> </ul>	7
Arturo Pareja	<ul style="list-style-type: none"> <li>• Empezando estilo del Login</li> <li>• Terminando con las últimas issues hasta la fecha</li> </ul>	6
Daniel Escoz	<ul style="list-style-type: none"> <li>• Diseño e implementación de la vista global de la página</li> </ul>	10-12
Roberto Marín	<ul style="list-style-type: none"> <li>• Realización de la acción para usar las habilidades.</li> <li>• Realización del código correspondiente a las acciones individuales y grupales</li> </ul>	14
Alexandra Martín		
Javier López	<ul style="list-style-type: none"> <li>• Finalización de las acciones presentes para la demo de marzo</li> <li>• Incorporación de Ajax al partido</li> <li>• Revisión del módulo de partidos e incorporación de nuevas funcionalidades</li> </ul>	16
Manuel Artero	<ul style="list-style-type: none"> <li>• Gestión del grupo en semana de gran actividad           <ul style="list-style-type: none"> <li>◦ Revisión diaria y sistemática de las issues del proyecto (todos los miembros asignados sin problemas para trabajar)</li> </ul> </li> <li>• Implementación de la página de index principal</li> <li>• Ayuda en asuntos menores al resto de miembros</li> </ul>	11
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Arreglo sobre los datos originales</li> <li>• Infructuoso intento por hacer funcionar la versión online</li> <li>• Documentadas las versiones para el control del server</li> <li>• Actualización de la documentación antigua</li> </ul>	13

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

Pedro Morgado	<ul style="list-style-type: none"> <li>Estilo del Main Layout</li> </ul>	10
Samuel Méndez	<ul style="list-style-type: none"> <li>Estilo visual de la página</li> <li>Registro de la página</li> </ul>	10

### 5.20. 4 al 10 de marzo

- Commits: 42
- Número total de horas:
- Descripción: continuación de la segunda iteración. Todo el equipo centrado en la presentación de la demo de marzo.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>Mensajería interna</li> </ul>	8
Arturo Pareja	<ul style="list-style-type: none"> <li>Estilo de los mensajes flash</li> <li>Estilo del Login</li> </ul>	9
Daniel Escoz	<ul style="list-style-type: none"> <li>Continuación de diseño e implementación de la vista principal</li> <li>Integración de fondos</li> </ul>	10-12
Roberto Marín	<ul style="list-style-type: none"> <li>Realización de la vista correspondiente al calendario.</li> <li>Corrección de errores en el controlador Acciones</li> </ul>	10
Alexandra Martín	<ul style="list-style-type: none"> <li>Módulo de mensajería.</li> </ul>	11
Javier López	<ul style="list-style-type: none"> <li>Creación del sistema de actualización de recursos</li> <li>Creación del sistema de finalización de acciones grupales/individuales</li> <li>Reparación de diversos bugs en general</li> <li>Subida del sistema a Amazon Web Services</li> </ul>	15
Manuel Artero	<ul style="list-style-type: none"> <li>Búsqueda de imágenes libres para la página</li> <li>Trabajo de diseño con el artista externo</li> <li>Gestión del grupo: revisión diaria y sistemática de todas las issues de la organización</li> </ul>	13
Marcos Alarcón	<ul style="list-style-type: none"> <li>Casos de uso</li> <li>Documentación variada</li> </ul>	4
Pedro Morgado	<ul style="list-style-type: none"> <li>Estilo del Main Layout</li> </ul>	8
Samuel Méndez	<ul style="list-style-type: none"> <li>Estilo de la página</li> <li>Cambios menores en BD y vistas</li> </ul>	6

### 5.21. 11 al 17 de marzo

- Commits: 82
- Número total de horas:
- Descripción: esta semana se hizo un simulacro para la demo del día 20. Se detectaron todos los fallos y se reasignaron para su corrección.

# JUGADOR #12

Documentación de organización: organización y reparto de tareas      Ingeniería del Software, 2013

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Acabar mensajería interna</li> <li>• Comenzar UML</li> </ul>	6
Arturo Pareja	<ul style="list-style-type: none"> <li>• Retoque en el estilo de Login</li> <li>• Reestructuración de la vista Registro de usuarios</li> <li>• Comienzo del estilo de Registro de usuarios</li> <li>• Comienzo del estilo de Registro de equipos</li> </ul>	6
Daniel Escoz	<ul style="list-style-type: none"> <li>• Diseño e implementación de páginas individuales</li> </ul>	6-7
Roberto Marín	<ul style="list-style-type: none"> <li>• Cambios en la fórmula del partido.</li> <li>• Ajuste de los efectos de las acciones</li> </ul>	12
Alexandra Martín	<ul style="list-style-type: none"> <li>• Debug mensajería interna.</li> <li>• Mostrar errores de la página con notificaciones Flash.</li> <li>• Preparación de la demo</li> </ul>	11
Javier López	<ul style="list-style-type: none"> <li>• Corrección de los problemas de Ajax en el partido</li> <li>• Corrección de los problemas de factores del partido</li> <li>• Corrección de problemas al participar en grupales</li> </ul>	12
Manuel Artero	<ul style="list-style-type: none"> <li>• Organización del grupo de cara a la demo del día 20</li> <li>• Creación de las nuevas issues y asignación a los miembros del grupo</li> </ul>	10
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Revisión de código, correcciones menores</li> </ul>	2
Pedro Morgado	<ul style="list-style-type: none"> <li>• Diseño e implementación de las páginas individuales</li> </ul>	7
Samuel Méndez	<ul style="list-style-type: none"> <li>• Cambios menores en BD y vistas</li> </ul>	2

## 5.22. 18 al 24 de marzo

- Commits: 8
- Número total de horas:
- Descripción: presentación de la primera versión del producto. Generada nueva documentación del segundo cuatrimestre.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• UML</li> </ul>	10
Arturo Pareja	<ul style="list-style-type: none"> <li>• Investigación de JQuery</li> <li>• Retoque en estilo del Login</li> <li>• Mejoras en Registro de usuario y equipo</li> <li>• Comienzo del estilo del Registro de personajes</li> <li>• Añadidas funciones JQuery para Registro de equipo y personajes</li> <li>• Optimización y redimensionamiento de las imágenes de</li> </ul>	8

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

	escudos y fondos	
Daniel Escoz	<ul style="list-style-type: none"> <li>Finalización del diseño de la página</li> </ul>	12
Roberto Marín	<ul style="list-style-type: none"> <li>Corrección de pequeños bugs en las acciones</li> </ul>	6
Alexandra Martín		
Javier López	<ul style="list-style-type: none"> <li>Corrección de pequeños bugs restantes antes de la presentación</li> <li>Generación de los datos finales de la presentación</li> <li>Diagramas de flujo de ejecución</li> </ul>	16
Manuel Artero	<ul style="list-style-type: none"> <li>Diseño y documentación de la nueva iteración del proyecto</li> </ul>	7
Marcos Alarcón	<ul style="list-style-type: none"> <li>Revisión de la BD</li> <li>Revisión de la seguridad</li> </ul>	5
Pedro Morgado	<ul style="list-style-type: none"> <li>Finalización del diseño de la página</li> </ul>	10
Samuel Méndez	<ul style="list-style-type: none"> <li>Resolución de bugs</li> </ul>	1

### 5.23. 25 al 31 de marzo

- Commits: 5
- Número total de horas:
- Descripción: vacaciones de Semana Santa.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>Vacaciones de semana Santa</li> </ul>	0
Arturo Pareja	<ul style="list-style-type: none"> <li>Vacaciones de semana Santa</li> </ul>	0
Daniel Escoz	<ul style="list-style-type: none"> <li>Retoques puntuales del diseño</li> </ul>	1
Roberto Marín		
Alexandra Martín		
Javier López	<ul style="list-style-type: none"> <li>Separación de las funciones complejas a los modelos correspondientes</li> </ul>	4
Manuel Artero	<ul style="list-style-type: none"> <li></li> </ul>	
Marcos Alarcón	-	0
Pedro Morgado	<ul style="list-style-type: none"> <li>Vacaciones de Semana Santa</li> </ul>	0
Samuel Méndez	<ul style="list-style-type: none"> <li>Búsqueda e implementación en Java del algoritmo para generar una liga de fútbol</li> </ul>	5

### 5.24. 1 al 7 de abril

## JUGADOR #12

Documentación de organización: organización y reparto de tareas      Ingeniería del Software, 2013

- Commits: 74
- Número total de horas:
- Descripción: comienzo de una tercera iteración de implementaciones. Revisión completa de la documentación.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	• Mensajería interna	4
Arturo Pareja	• Testeo del estilo y de las nuevas funciones JQuery	2
Daniel Escoz	• Comienzo de la vista del partido	7-8
Roberto Marín		
Alexandra Martín	• Esqueleto de las notificaciones	2
Javier López	• Corrección de la documentación del primer cuatrimestre en función de la revisión del profesor • Generación de nueva documentación del segundo cuatrimestre	22
Manuel Artero		
Marcos Alarcón	• Implementado el script generaLiga()	12
Pedro Morgado		
Samuel Méndez	• Debugging básico	2

### 5.25. 8 al 14 de abril

- Commits: 106
- Número total de horas:
- Descripción: continuación de la tercera iteración de implementaciones y revisión de la documentación.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	• Mensajería interna	6
Arturo Pareja	• Nueva redimensión del tamaño de los escudos • Pruebas con las funciones JQuery del Registro • Diseño e implementación de la vista de Afición	6
Daniel Escoz	• Diseño e implementación de la vista del partido	12
Roberto Marín	• Código para ejecutar las acciones de partido. • Código en el modelo de AccionesTurno	8
Alexandra Martín	• Módulo de notificaciones • Cron para las notificaciones	10

## JUGADOR #12

**Documentación de organización: organización y reparto de tareas** Ingeniería del Software, 2013

Javier López	<ul style="list-style-type: none"> <li>• Corrección de la documentación del primer cuatrimestre en función de la revisión del profesor</li> <li>• Generación de nueva documentación del segundo cuatrimestre</li> </ul>	17
Manuel Artero	<ul style="list-style-type: none"> <li>• Apoyo en la implementación de diversas funcionalidades</li> </ul>	4
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Generación de partidos</li> <li>• Reparto de ánimo automático</li> <li>• Calendario de liga según las horas</li> </ul>	8
Pedro Morgado	<ul style="list-style-type: none"> <li>• Diseño e implementación de la vista de habilidades</li> </ul>	7
Samuel Méndez	<ul style="list-style-type: none"> <li>• Comienzo con el chat. Funcionamiento básico y algo de estilo</li> </ul>	7

### 5.26. 15 al 21 de abril

- Commits: 143
- Número total de horas:
- Descripción: se continúa con la implementación y revisión de las funcionalidades de la demo de abril. Se revisa por completo el estilo de la web.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Sistema de experiencia. Que se necesite un nivel para desbloquear habilidades y eso</li> <li>• CSS de mensajería y notificaciones.</li> <li>• CSS vista de participar en acción grupal</li> </ul>	10
Arturo Pareja	<ul style="list-style-type: none"> <li>• Diseño e implementación de la vista de Afición</li> <li>• Comenzando estilo de la vista de Afición</li> <li>• Cambios en las vistas de Registro</li> </ul>	10
Daniel Escoz	<ul style="list-style-type: none"> <li>• Diseño e implementación de la vista del partido</li> <li>• Integración de AJAX</li> </ul>	12
Roberto Marín	<ul style="list-style-type: none"> <li>• Corrección de errores en el controlador de Acciones</li> <li>• Mejora de la fórmula del partido</li> <li>• Corregir bugs en las acciones.</li> </ul>	9
Alexandra Martín	<ul style="list-style-type: none"> <li>• Validación de formulario</li> <li>• Debug</li> </ul>	4
Javier López	<ul style="list-style-type: none"> <li>• Corrección de la documentación del primer cuatrimestre en función de la revisión del profesor</li> <li>• Generación de nueva documentación</li> <li>• Cambios en el estilo de varias secciones de la web</li> <li>• Debugging para la demo del día 28 de abril</li> </ul>	20
Manuel Artero	<ul style="list-style-type: none"> <li>• Imágenes de equipos editadas en Gimp</li> <li>• Imágenes de acciones del juego</li> <li>• Ayuda en el estilo</li> </ul>	16

## JUGADOR #12

**Documentación de organización: organización y reparto de tareas** Ingeniería del Software, 2013

Marcos Alarcón	<ul style="list-style-type: none"> <li>• Recuperación de documentos perdidos</li> <li>• Actualización de la documentación por los cambios en el proyecto</li> </ul>	5
Pedro Morgado	<ul style="list-style-type: none"> <li>• Diseño e implementación de la vista de habilidades</li> </ul>	9
Samuel Méndez	<ul style="list-style-type: none"> <li>• Integración del chat con la página. Corrección errores.</li> <li>• Debugging de la página</li> </ul>	6

### 5.27. 22 al 28 de abril

- Commits: 0
- Número total de horas:
- Descripción: presentación de la beta. Semana centrada completamente en minimizar los bugs del sistema.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Hacer que las acciones de partido den experiencia</li> <li>• Notificaciones</li> </ul>	6
Arturo Pareja	<ul style="list-style-type: none"> <li>• Creación de combinaciones de colores para los dibujos del partido con Photoshop</li> <li>• Redimensión para los iconos de habilidades</li> <li>• Reducida la resolución de los fondos</li> <li>• Estilo de la vista de Afición</li> <li>• Retoque en la vista de partidos</li> <li>• Retoque en el estilo del chat</li> </ul>	12
Daniel Escoz	<ul style="list-style-type: none"> <li>• Finalización de la vista del partido para la demo</li> </ul>	10-12
Roberto Marín	<ul style="list-style-type: none"> <li>• Corrección de bugs en la fórmula</li> </ul>	5
Alexandra Martín	<ul style="list-style-type: none"> <li>• Recuperación de influencias de forma gradual</li> </ul>	7
Javier López	<ul style="list-style-type: none"> <li>• Corrección de errores de cara a la demo del día 28 de abril</li> <li>• Ampliación del servidor para dicha demo</li> </ul>	26
Manuel Artero	<ul style="list-style-type: none"> <li>• Búsqueda y anotación en issues de los bugs en la aplicación de cara a la presentación de abril</li> <li>• Tratamiento de dibujos de partido en el editor de imágenes Gimp</li> </ul>	15
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Añadidos datos la genera el partido</li> <li>• Añadidos datos a la vista de asistir al partido</li> <li>• Revisión de las vistas de acciones grupales</li> </ul>	12
Pedro Morgado	<ul style="list-style-type: none"> <li>• Revisión general de las vistas que no tenían estilo</li> <li>• Ayuda con la corrección de fallos</li> </ul>	12
Samuel Méndez	<ul style="list-style-type: none"> <li>• Footer</li> <li>• Debugging de la página</li> <li>• Chat independiente para cada partido</li> </ul>	5

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

### 5.28. 29 de abril al 5 de mayo

- Commits: 12
- Número total de horas:
- Descripción: avance mínimo del proyecto. Semana de descanso para retomar la última tanda de implementaciones y documentación.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	● Puente de mayo	0
Arturo Pareja	● Puente de mayo	0
Daniel Escoz	● Puente de Mayo	0
Roberto Marín		
Alexandra Martín	● Añadidos sin importancia (botones)	1
Javier López	● Mínima revisión de documentos	0.5
Manuel Artero		
Marcos Alarcón	-	0
Pedro Morgado	● Puente de mayo	0
Samuel Méndez		0

### 5.29. 6 al 12 de mayo

- Commits: 11
- Número total de horas:
- Descripción: se deciden los últimos módulos de la cuarta iteración. Se comienza a agregar la ayuda, mejorar el partido y cambiar algunos aspectos menores. También se continúa la documentación del segundo cuatrimestre.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	● Comenzada la ayuda	7
Arturo Pareja	● Post-puente (otras asignaturas)	0
Daniel Escoz	● Post-puente (otras asignaturas)	0
Roberto Marín	● Incorporando las nuevas acciones	2
Alexandra Martín		
Javier López	● Revisión de los documentos para su finalización en la última iteración del proyecto	6
Manuel Artero		9

## JUGADOR #12

**Documentación de organización: organización y reparto de tareas** Ingeniería del Software, 2013

Marcos Alarcón	<ul style="list-style-type: none"> <li>• Implementando el cooldown en las acciones de partido</li> <li>• Revisión de los datos de prueba</li> <li>• Pequeños cambios en la base de datos</li> </ul>	10
Pedro Morgado	<ul style="list-style-type: none"> <li>• Pequeños cambios en el estilo de la página</li> </ul>	3
Samuel Méndez	<ul style="list-style-type: none"> <li>• Cambios menores</li> </ul>	2

### 5.30. 13 al 19 de mayo

- Commits: 125
- Número total de horas:
- Descripción: continuación de los nuevos cambios para la demo del mes de mayo y la documentación.

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>• Ayuda</li> </ul>	6
Arturo Pareja	<ul style="list-style-type: none"> <li>• Documentación del proyecto con PHPDocumentor</li> </ul>	9
Daniel Escoz	<ul style="list-style-type: none"> <li>• Implementación final de la vista del partido</li> </ul>	12-14
Roberto Marín	<ul style="list-style-type: none"> <li>• Realización de todo el código de las nuevas acciones</li> </ul>	8
Alexandra Martín	<ul style="list-style-type: none"> <li>• Pruebas</li> </ul>	2
Javier López	<ul style="list-style-type: none"> <li>• Continuación de la documentación del segundo cuatrimestre</li> <li>• Inicio de redacción de la parte técnica final</li> </ul>	10
Manuel Artero		
Marcos Alarcón	<ul style="list-style-type: none"> <li>• Revisión de los cooldown en las acciones individuales</li> </ul>	2
Pedro Morgado	<ul style="list-style-type: none"> <li>• Corrección de pequeños fallos en la página</li> <li>• Pequeños cambios en el estilo de algunas vistas</li> </ul>	5
Samuel Méndez	<ul style="list-style-type: none"> <li>• Cambios menores</li> </ul>	2

### 5.31. 20 al 26 de mayo

- Commits: ? ;(
- Número total de horas:
- Descripción: en este periodo se han preparado todo para la demo del mes de mayo. Incorporada ayuda en el juego, mejoras en la información del partido y otros cambios menores. Presentación de la versión final en los laboratorios.

## JUGADOR #12

Documentación de organización: organización y reparto de tareas      Ingeniería del Software, 2013

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	• Tutorial	9
Arturo Pareja	• Subida toda la documentación de PHPDocumentor	3
Daniel Escoz	• Barras del partido, AJAX, etc. • Testeo final del partido	10-12
Roberto Marín	• Debugging de las nuevas funcionalidades	3
Alexandra Martín	• Diagramas de casos de uso y revisión de los casos de uso.	4
Javier López	• Debugging de las nuevas funcionalidades del sistema • Montaje de un nuevo servidor para soportar mayor concurrencia de usuarios • Finalización de detalles menores en la implementación de la demo	15
Manuel Artero		12
Marcos Alarcón	• Revisión general del código	2
Pedro Morgado	• Corrección de fallos en la página • Cambio de estilo en algunas vistas • Dibujos que se muestran durante el partido	10
Samuel Méndez	• Dibujos que se muestran durante el partido	7

### 5.32. 27 de mayo al 2 de junio

- Commits: 0
- Número total de horas:
- Descripción: finalización de documentación y preparación de la presentación final en el salón de actos

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	• Preparación para la presentación	2
Arturo Pareja	• Ensayo del teatro dentro de la presentación final	4
Daniel Escoz	• Preparación de la presentación final	3
Roberto Marín	• Ensayo del teatro dentro de la presentación final	2.5
Alexandra Martín	• Ensayo para la presentación	5
Javier López	• Redacción de la parte técnica de la presentación • Creación de las diapositivas de la primera parte de tecnologías • Finalización de los casos de uso y diagramas de flujo	9

## JUGADOR #12

Documentación de organización: organización y reparto de tareas Ingeniería del Software, 2013

Manuel Artero	<ul style="list-style-type: none"> <li>Video de presentación del proyecto</li> </ul>	14
Marcos Alarcón	<ul style="list-style-type: none"> <li>Creación de las diapositivas de la segunda parte de tecnologías</li> </ul>	4
Pedro Morgado	<ul style="list-style-type: none"> <li>Preparación de la presentación final</li> </ul>	4
Samuel Méndez	<ul style="list-style-type: none"> <li>Primer esquema del teatro para la presentación y transparencias</li> </ul>	4

### 5.33. 3 al 9 de junio

- Commits: 0
- Número total de horas:
- Descripción: últimos ensayos y presentación final en el salón de actos

Miembro	Tareas a realizar	Horas dedicadas
Marina Bezares	<ul style="list-style-type: none"> <li>Últimos retoques en las presentaciones</li> <li>Presentación del proyecto en el salón de actos</li> </ul>	2
Arturo Pareja	<ul style="list-style-type: none"> <li>Últimos retoques en las presentaciones</li> <li>Presentación del proyecto en el salón de actos</li> </ul>	4
Daniel Escoz	<ul style="list-style-type: none"> <li>Últimos retoques en las presentaciones</li> <li>Presentación del proyecto en el salón de actos</li> </ul>	4
Roberto Marín	<ul style="list-style-type: none"> <li>Últimos retoques en las presentaciones</li> <li>Presentación del proyecto en el salón de actos</li> </ul>	1
Alexandra Martín	<ul style="list-style-type: none"> <li>Últimos retoques en las presentaciones</li> <li>Presentación del proyecto en el salón de actos</li> </ul>	3
Javier López	<ul style="list-style-type: none"> <li>Últimos retoques en las presentaciones</li> <li>Presentación del proyecto en el salón de actos</li> </ul>	2
Manuel Artero	<ul style="list-style-type: none"> <li>Últimos retoques en las presentaciones</li> <li>Presentación del proyecto en el salón de actos</li> </ul>	
Marcos Alarcón	<ul style="list-style-type: none"> <li>Últimos retoques en las presentaciones</li> <li>Presentación del proyecto en el salón de actos</li> </ul>	4
Pedro Morgado	<ul style="list-style-type: none"> <li>Últimos retoques en las presentaciones</li> <li>Presentación del proyecto en el salón de actos</li> </ul>	4
Samuel Méndez	<ul style="list-style-type: none"> <li>Últimos retoques en las presentaciones</li> <li>Presentación del proyecto en el salón de actos</li> </ul>	4

# JUGADOR #12

Documentación adicional: encuestas

Ingeniería del Software, 2013

## Encuesta del 20 de marzo

Te damos la siguiente escala del 1 al 5; te pedimos que nos digas tu opinión sobre las siguientes ideas (rellena con un número cada cuadrado).

- 1: No sólo es mala idea, sino que lo empeoraría.
- 2: No gastéis esfuerzo en eso
- 3: Bien... (ni me gusta ni me disgusta)
- 4: Me gusta la idea, ¡adelante!
- 5: IM-PLE-MEN-TAD-LO ¡Ya!

### 1. Comunicación entre jugadores



A. Se nos ha ocurrido añadir un **foro para las acciones de equipo**: cada vez que alguien inicie una acción en la que participa todo el equipo, se abriría un hilo de conversación para discutir con tus compañeros qué recursos aportar. El hilo lo eliminaríamos al finalizar la acción.



B. Hemos pensado en añadir una ventana de **chat durante el partido**. Para picar a los rivales que estén viendo el partido y hablar con tus compañeros de equipo.



C. Estamos metiendo un **sistema de notificaciones** automático, de forma que el sistema te avise cada vez que tu equipo completa una acción, o alguien participa en una acción de la que formes parte.

Dinos el por qué de tus respuestas.

### 2. Frecuencia de partidos



A. Estamos dudando si programar 2 - 3 partidos *a la semana*. Esto haría un **juego más estratégico** en el que forzaríamos a ser más cuidadosos a la hora de elegir cuántos recursos gastamos, en qué acciones etc.



B. Por el contrario, podríamos establecer 1 - 2 partidos *al día*. Las acciones que completas con tus compañeros serían más baratas y con menor repercusión en el partido buscando un **juego más fluido**.

Dinos el por qué de tus respuestas.

## JUGADOR #12

Documentación adicional: encuestas

Ingeniería del Software, 2013

### 3. Ritmo del partido

A. Hemos mostrado un partido de prueba en el que no podías interactuar con él. La idea es que puedas hacer acciones durante el partido ¿Qué opinas de un partido corto de 15 min. aproximadamente?

B. ¿Te gusta más la idea de partidos largos y estratégicos (alrededor de una hora)?

Dinos el por qué de tus respuestas.

### 4. ¿Qué ha sido lo que más te ha gustado?

### 5. ¿Qué ha sido lo que menos te ha gustado?

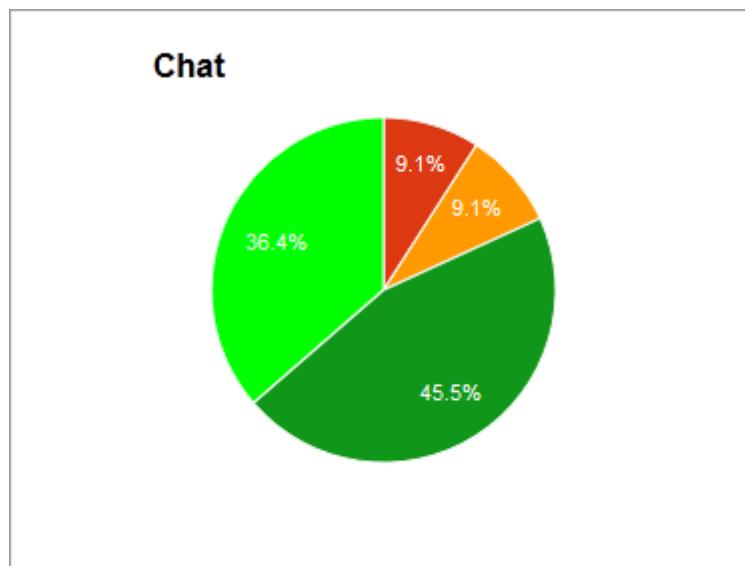
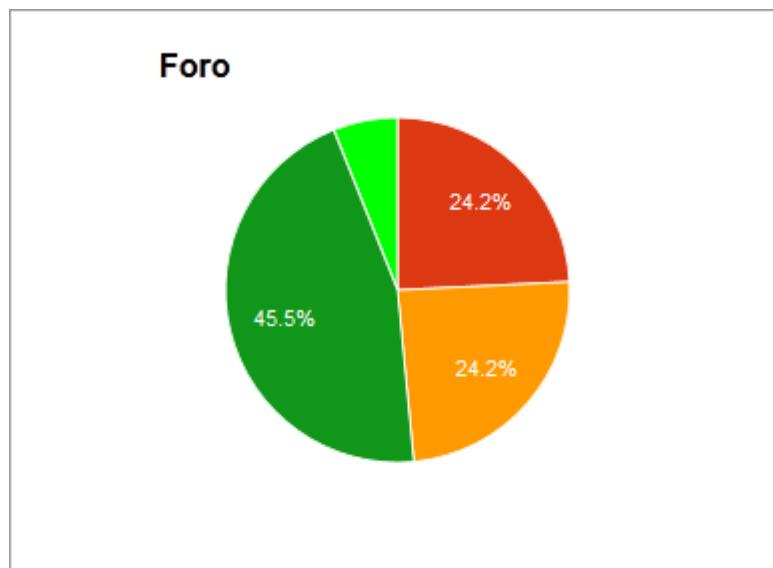
# JUGADOR #12

Documentación adicional: encuestas

Ingeniería del Software, 2013

## Resultados de la encuesta del 20 de marzo

	Foro	Chat	Notificaciones	2-3 p / semana	1 - 2 p / día	Partido 15 min	Partido hora
Muy mala idea	0	0	0	1	5	4	11
Mala idea	8	3	1	6	8	5	17
Regular	8	3	10	12	8	5	4
Buena idea	15	15	14	8	9	15	1
Muy buena idea	2	12	7	5	2	5	1

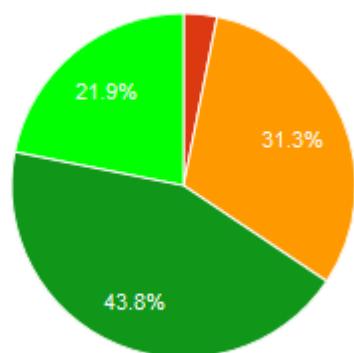


# JUGADOR #12

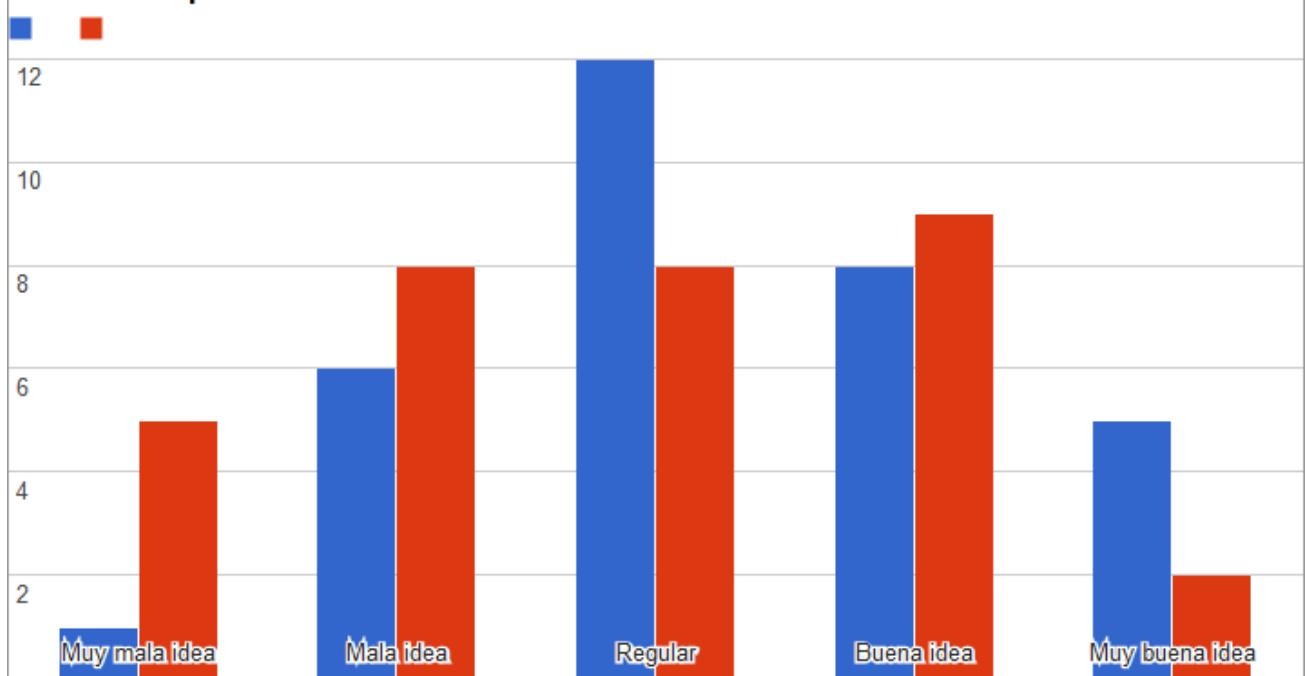
Documentación adicional: encuestas

Ingeniería del Software, 2013

## Notificaciones



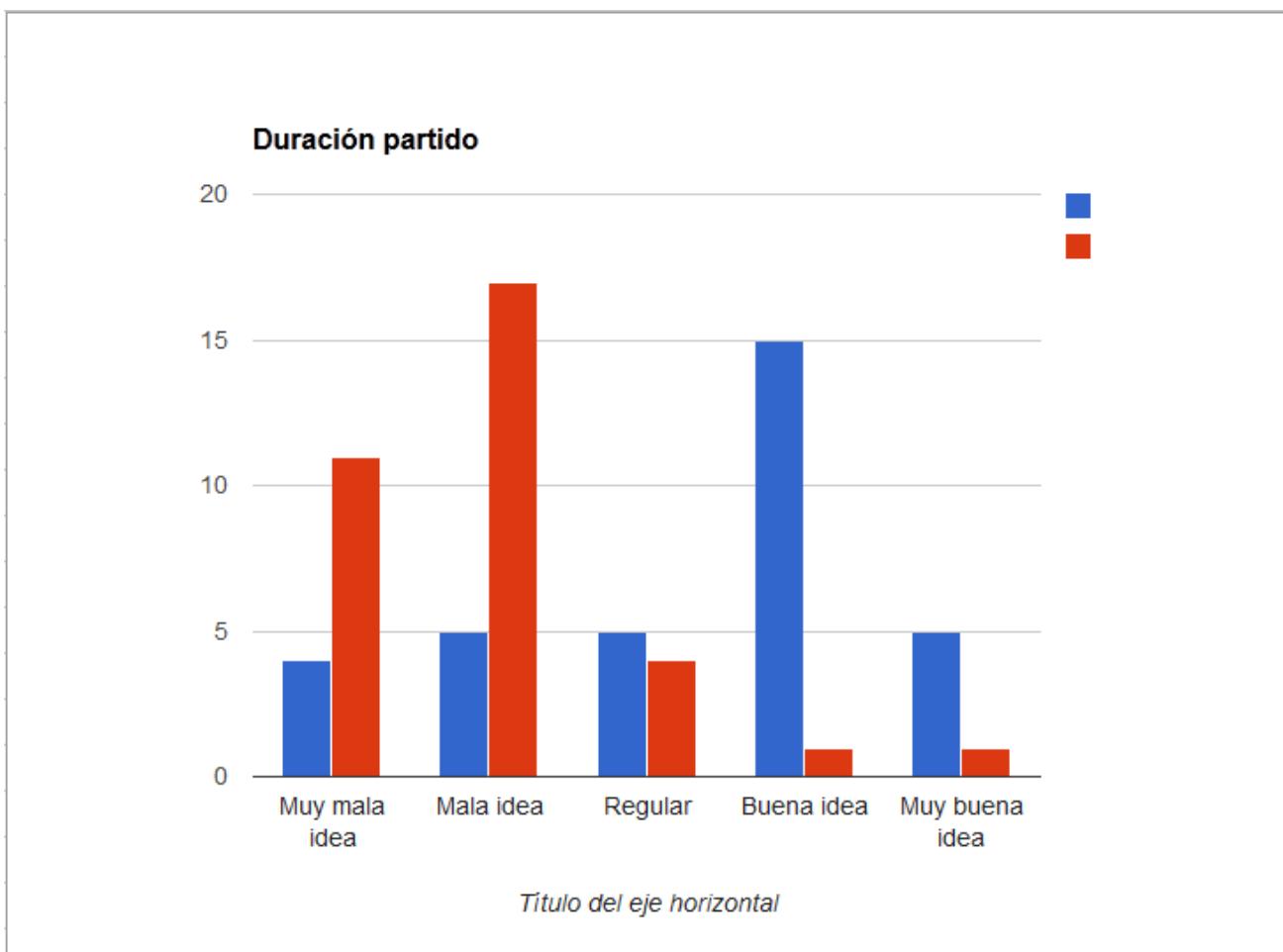
## Frecuencia partidos



## JUGADOR #12

Documentación adicional: encuestas

Ingeniería del Software, 2013



## Encuesta del 25 de abril

1. Nuestro mayor problema es conseguir un juego lo suficientemente intuitivo como para que un nuevo jugador no se sienta perdido en nuestra página... sabemos que tenemos que trabajar en hacer nuestra web más fácil de usar.

Cualquier idea, opinión o sugerencia para mejorar ese aspecto será bienvenida.

## JUGADOR #12

Documentación adicional: encuestas

Ingeniería del Software, 2013

### 2. ¿Qué idea te gusta más?... y, ¿por qué?

- Un jugador puede **asistir como espectador** (pero no participar) a partidos en los que no juega su equipo; puede asistir si quiere a cualquier partido del juego.
- Un jugador puede asistir como espectador (pero no participar) a partidos en los que no juega su equipo; y además **escribir en el chat del partido**.
- Un jugador no puede asistir (ni siquiera como espectador) a un partido en el que no juegue su equipo.

### 3. ¿Qué opinas acerca del **partido**? ¿es lo que esperabas? ¿tenías claro qué equipo estaba controlando el partido en cada momento?.

### 4. ¿Qué es lo que **más** te ha gustado?

## JUGADOR #12

Documentación adicional: encuestas

Ingeniería del Software, 2013

**5.** ¿Qué es lo que **menos** te ha gustado?

¡ Gracias !

## JUGADOR #12

Documentación adicional: revisión del 13 de mayo, informal      Ingeniería del Software, 2013

### Usuario

- Recursos fuera de orden en muchos sitios.
- Casi todo está poco accesible o accesible desde lugares extraños <= DETALLAR <= TODO. ABSOLUTAMENTE TODO. El partido es difícil de encontrar, las acciones no están en "acciones", los partidos no están en "partidos"...
- Voy a participar en una acción, me pone cuántos recursos voy a meter ¿pero cuántos faltaban? ¿cuántos han participado? ¿cuántos recursos hay ya?
- Las descripciones de las habilidades son una mierda, solo repiten el nombre sin decir para qué sirven.(Marcos)
- Resaltar mi próximo partido, tengo que tener accesibilidad absoluta a él.
- EN la pantalla de afición, en la lista de acciones abiertas, quiero hacer una nueva; tengo que irme hasta otra ventana que luego me lleve a esta...
- Estado -7 qué es esto. Voy ganando o voy perdiendo
- La barra es casi verde, es que voy bien? Los colores de los equipos están al revés
- ¿No puedo borrar la cuenta? Si existe,no veo la opción.
- Estoy esperando al partido, en la pantalla de previa, llega la hora de partido me manda al calendario ¡Estaba esperando en mi sitio!
- Desde la pantalla de afición quiero crear acciones nuevas
- Desbloqueadas, habilidades, lista de la afición => facilitar crear acciones
- El login está sin acabar
- Se pueden completar muy pocas acciones grupales con los recursos iniciales. (Marina)
- El ultimo que completa una acción grupal con influencia en realidad no gasta influencia porque se la devuelven automáticamente (Marina)
- \*Las acciones de partido deberían quitarse porque no sirven para nada y cuestan recursos de desbloquear. En la demo la gente no sabrá que no sirven, las desbloquearan y confundiremos a nuestro público (Marina)
- Eso de que en una acción grupal solo puedan participar 3 es un poco escaso. La mayoría de las veces no se pueden completar. Sobre todo si en la demos del 24 solo van a comenzar con los recursos iniciales (Marina)
- La acción de generarse expectativas es un cheto. Esánimoo gratis por la cara porque la puedes usar siempre que te de la gana. (Marina)

### Otros

- La crónica sube automáticamente en el scroll cada 5 segundos
- En el partido, salen los cuadros de los equipos (los logos) muy raros
- La influencia se recupera muy rápido
- Los demás recursos también.
- La crónica pasa directamente del Turno 5 al 7 sin comentar que hay un descanso en medio.
- Aumentar número de participantes en las grupales (Samu)
- Lanzar aviso cuando tu acción iniciada se acaba por falta de recursos  
(a los 3 minutos, creo) (Samu)
- Un reloj en pantalla (Samu)

## JUGADOR #12

Documentación adicional: revisión del 13 de mayo, informal      Ingeniería del Software, 2013

- La imagen del césped no cubre la pantalla durante el registro (al elegir afición y pj).
- Al elegir afición te muestra los escudos pero no se puede elegir pinchando en ellos.
- Tengo que repetir el login nada más loggearme (es como el mensaje de confirmación)
- El nivel de equipo en el partido es 1, pero en la afición dice que es 10 (ambos equipos).
- Durante el partido por el pie chart, (el circulito rojo y verde que indicaba el estado) parecía que ganaban los verdes. Pero no es cierto estaban ganando los rojos. Los colores estaban intercambiados con respecto a local y visitante y daba esa sensación. (Marina)
- En la vista de partido pone estadísticas visitante visitante (Marina)
- La vista de participar en grupales es muy muy caótica (Marina)
- \*Al meterese en la vista del formulario para participar en una accion grupal y le das a volver hay que recargar la página. No está mal pero es muy pesado (Marina)

### FUTURO => ISSUES

Enhancement => Lanzar aviso despues de participar (¿ha tenido éxito?, ¿se ha llenado los usuarios?)

Enhancement => informacion extra en la pantalla de paritcipar recursos

Enhancement => influencias ..

Enhancement => Cambiar el layout de los errores => mensajito en lugar de error 404

Bug => Cuando ya has participado en una grupal y se ha alcanzado el número máximo de jugadores no deja volver a participar, aunque seas participante (Samu)

Bug => *if de rober* Al adquirir una acción individual y darle a Usar, la página peta al final (Samu)

de jugadores no deja volver a participar, aunque seas participante

¿Qué va a pasar con 100 usuarios?

Enhancement => Al crear una grupal me añade como particioante ¿por qué me pregunta?

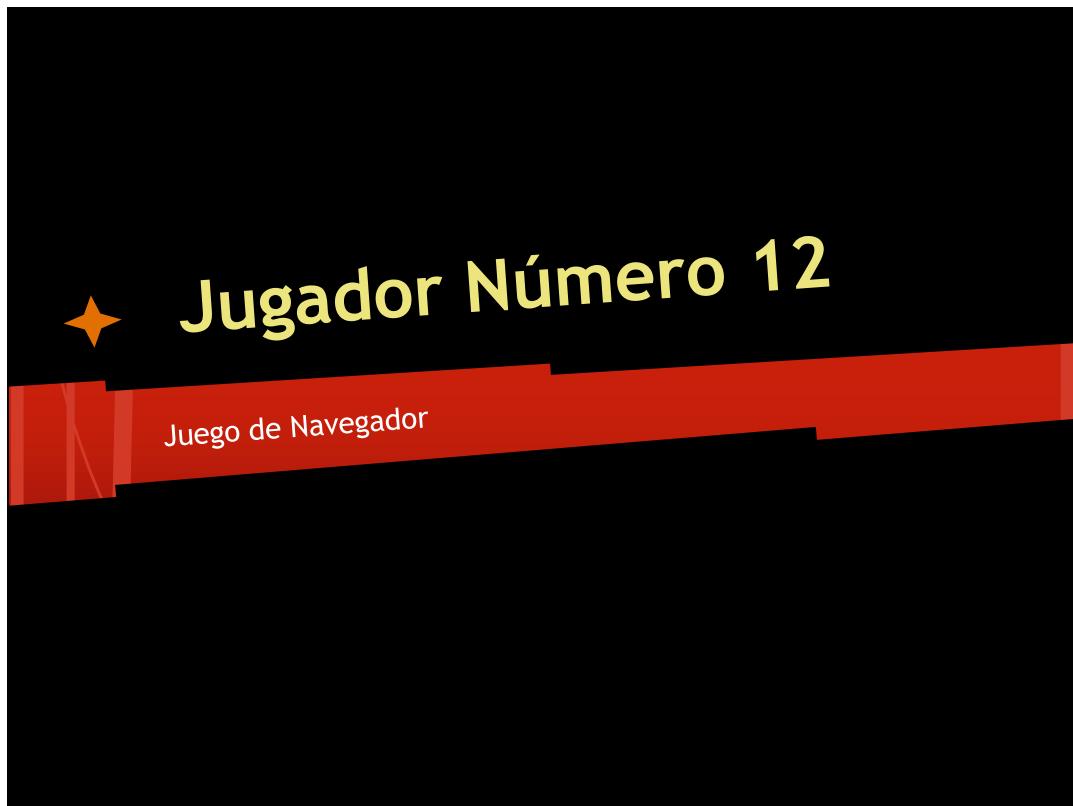
Bug => Comprobar actualizar la clasificación (Samu)

Revision/Debugging => ¿Al completar una accion grupal te devuelve los recursos de golpe?

Me ha parecido que así era. Todos, no solo la influencia (Marina)

Revision / Debugging => ¿Al acabar el partido nos ha dado ánimo?

### Presentación inicial del proyecto



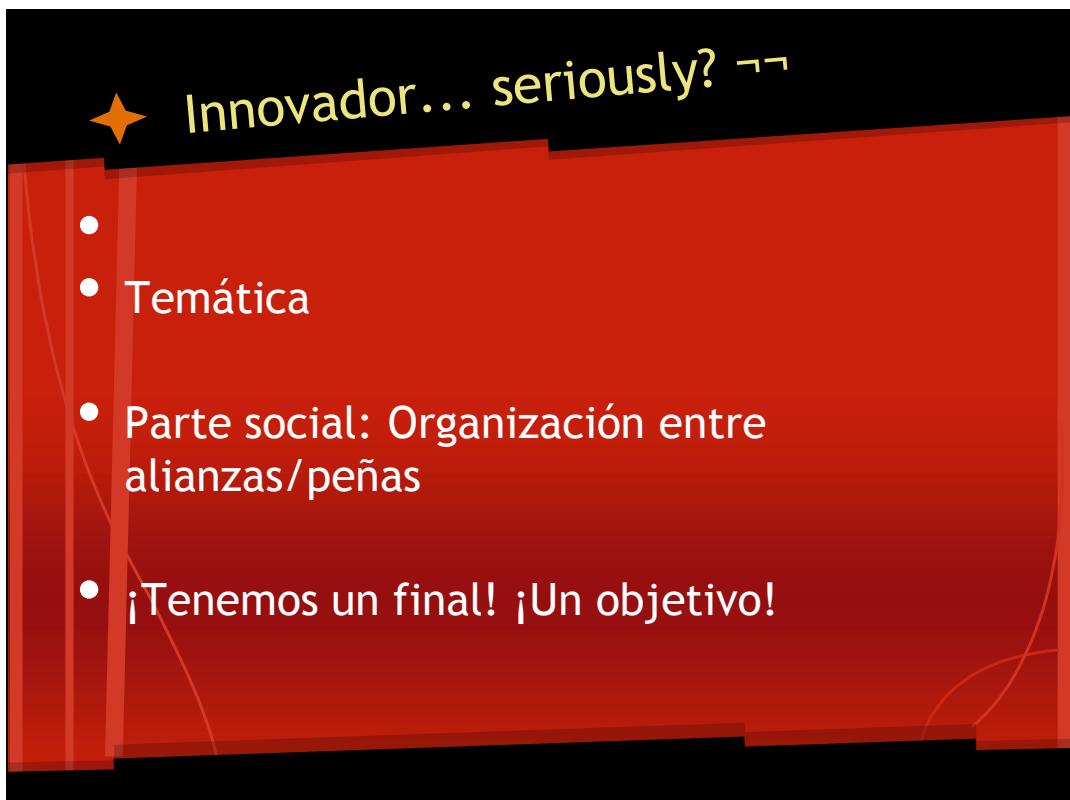
De qué va esto?

- Juego de navegador innovador al estilo OGame
- Temática => Lucha entre aficiones de equipos de fútbol
- ¿Y si no me gusta el fútbol? => Nos centramos en la movilización de hinchas, en la originalidad para animar ¡No en el deporte!

# JUGADOR #12

Documentación adicional: presentación inicial proyecto

Ingeniería del Software, 2013



### ◆ Bueeeeeno... cómo se juega

- Te registras en la Web y eliges un personaje.
- Te haces aficionado de un equipo y creas, te unes o pasas de las peñas de hinchas
- Administración de recursos para preparar los partidos y crear la mejor afición de la historia. ¡Organización de las peñas!

### ◆ El Ultra



## JUGADOR #12

Documentación adicional: presentación inicial proyecto

Ingeniería del Software, 2013

### ★ El empresario



### ★ La animadora de masas



### ► Ideas, ideas ¡IMPLEMENTACIÓN!

- Aplicación Web

- PHP
- AJAX
- DB

### ► ESCALABLE

- 
- Juego
- Aplicación para móvil (Android)
- Partidos interactivos

### ◆ Aliciente principal

VA A ESTAR... SÍ SÍ, VA A ESTAR  
MONTADO Y FUNCIONANDO EN UN  
SERVIDOR REAL.

♫ Usuarios premium ♫

### ◆ Contacto; ¿preguntas?

- 
- Grupo 8 personas
- [jugadornumerodoce@googlegroups.com](mailto:jugadornumerodoce@googlegroups.com)

## JUGADOR #12

Documentación adicional: revisión del proyecto en diciembre Ingeniería del Software, 2013

### Revisión del proyecto de diciembre

## SEGUIMIENTO DEL PROYECTO JUGADOR NÚMERO 12

### Métricas y estimaciones

#### ¿Qué métricas utilizamos?

Hemos elegido los **Puntos de función**; motivos:

1. Mide directamente la funcionalidad.
2. Son independientes de la tecnología.
3. Capacidad de medir un avance en el desarrollo en cualquiera de las fases de vida del software.
4. Métrica adaptable al proyecto.

## JUGADOR #12

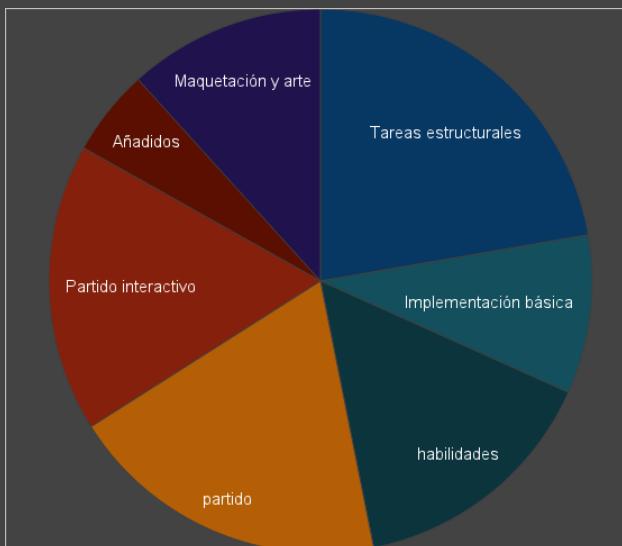
Documentación adicional: revisión del proyecto en diciembre Ingeniería del Software, 2013

### Puntos de función definidos

El proyecto consta de 350 puntos de función que representan funcionalidad del sistema agrupada en varios módulos.

Módulo	Peso	Dificultad	PF
Login y registro de usuarios	3	2	6
Uso de habilidades de jugador	7	2	14
Partido: jugarse sin interacción	10	4	40
Partido: partido interactivo	12	5	60
Arte	9	3	27

### Puntos de función definidos



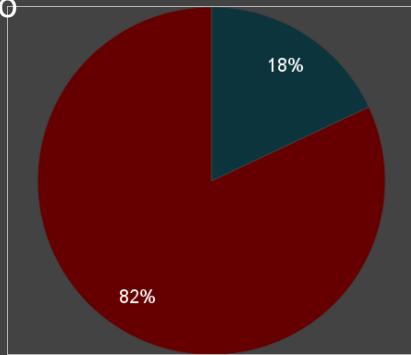
### ¿Cómo medimos la progresión?

Escala propia para medir la progresión en una determinada tarea; 4 factores:

1. **Implementación** de la funcionalidad: 40%
2. **Seguridad**: la implementación es segura (transacciones, inyección sql...): 20%
3. **Testing**: 20%
4. **Integración** con el resto de funcionalidad: 20%

### ¿Por dónde vamos?

- Fase de inicio: definición del juego.
- Fase de elaboración: implementación de los primeros casos de uso

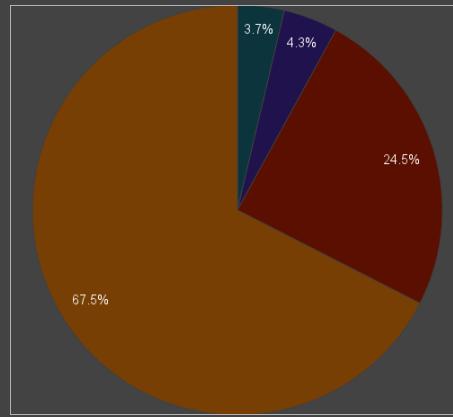


## JUGADOR #12

Documentación adicional: revisión del proyecto en diciembre Ingeniería del Software, 2013

### Estimación LDC

Lenguaje	LDC
CSS	700
JavaScript	600
MySQL	700
HTML	4000
PHP	11000
Total	17000



## Introducción

El presente documento muestra los resultados de aplicar el Proceso Unificado de Desarrollo al proyecto actual. Sólo se dará una visión de las fases del mismo, sin entrar al detalle de actividades estructurales.

## Índice

1. Consideraciones iniciales
2. Ciclo de desarrollo
  - 2.1 Fase de inicio
  - 2.2 Fase de elaboración
  - 2.3 Fase de construcción
  - 2.4 Fase de transición
3. Futuros ciclos de desarrollo
4. Casos de uso
5. Diagramas UML
6. Controles de calidad
7. Consideraciones finales

## 1. Consideraciones iniciales

En el proyecto actual se hace uso de un Proceso de Desarrollo Unificado (RUP). Esto obliga a seguir determinadas pautas como la dirección por casos de uso, división de la construcción y versionamiento del producto en una serie de ciclos o el uso de diagramas UML.

## 2. Ciclo de desarrollo

El modelo RUP califica la vida de un sistema como una serie de ciclos. Dadas las limitaciones de tiempo del curso actual, para la asignatura de Ingeniería del Software tan sólo podrá ser llevado a cabo uno de estos ciclos. En él se tratará de llegar a una versión básica y perfectamente funcional del producto.

Este ciclo, siguiendo la teoría del modelo RUP, se divide en varias fases explicadas a continuación. Éstas se encuentran brevemente explicadas en el apéndice del documento de especificación de requisitos software, por lo que en el presente documento nos limitaremos a dar una visión de la aplicación de las mismas al proyecto actual.

### 2.1 Fase de inicio

Esta fase ha ocupado las primeras semanas tras la creación del grupo de trabajo. En ella se han llevado a cabo reuniones más frecuentes para hacer brainstorming de ideas y delimitar lo máximo posible la estructura básica del producto (Jugador Numero 12).

Se ha dedicado bastante tiempo a esta fase ya que se consideraba importante para que todos los miembros del grupo tuvieran suficientemente claro el enfoque y funcionalidades del producto para poder trabajar cómodamente.

### 2.2 Fase de elaboración

Al igual que en el caso anterior, esta fase ha ocupado otras pocas semanas del proyecto. Se han llevado a cabo reuniones regulares con diversos fines:

- Selección de las tecnologías y frameworks a emplear. Hubo bastantes debates con respecto al framework de PHP a emplear. Se presentaron dos alternativas (Yii y Symfony/Silex). Finalmente se decidió hacer uso de Yii.
- Selección del repositorio y alojamiento de la documentación. En este aspecto se decidió de forma unánime el uso de GitHub en lo referente al código de la aplicación y Google Drive (con Google Docs) para la documentación.
- Establecer unas bases arquitectónicas sólidas para la creación del producto.
- Instrucción a los miembros del grupo en las tecnologías a usar mediante clases privadas en reuniones, emails de ayuda y tutoriales en la documentación.
- Creación de nuevos roles. Al margen del papel de líder de grupo, que, como es obvio, ya había sido establecido, se crearon nuevos roles:
  - El denominado “bibliotecario”, que se encargaría de la supervisión de la estructura y coherencia de la documentación. Este rol iría cambiando cada semana para evitar que un único miembro del grupo cargase con esta tediosa tarea durante demasiado tiempo.
  - El supervisor de código, cuya función principal sería garantizar la corrección del código creado para la aplicación y asesorar en caso de dudas de

## JUGADOR #12

Documentación adicional: RUP en el proyecto actual

Ingeniería del Software, 2013

implementación. Esto es, que se respete el patrón MVC, que se eliminen las máximas brechas de seguridad posibles, que se garantice la estabilidad del sistema, etc.

- El supervisor de repositorio, que se ocuparía de garantizar la coherencia en los commits del repositorio. Por ejemplo, que no se suban fragmentos incoherentes, que se respete la organización de ramas, etc.

A medida que el proyecto avanzaba y todo se empezaba a hacer de forma correcta (los commits organizados, ramas respetadas, código decente dentro de unos límites, etc) este sistema de roles se ha ido relegando a un segundo plano.

- Asignación de tareas. A cada miembro del grupo le fueron asignadas una serie de tareas iniciales para comenzar la construcción del producto.

### 2.3 Fase de construcción

Siguiendo una estructura de iteraciones sobre el producto, cerca del mes de diciembre se empezó con la implementación del sistema. Para ello se partió de una base muy simple que incluía el sistema de sesiones y el esqueleto básico de la aplicación. Sobre ella se irían agregando diversos módulos directamente relacionados con las funcionalidades/casos de uso y que, una vez unidos, conformarían el sistema terminado.

Para cada iteración se ha establecido una breve fase de análisis de requisitos y diseño, posteriormente se ha procedido a la implementación y finalmente a las pruebas de la misma.

- En una primera iteración se fueron creando elementos básicos como el sistema de registro, la visualización de datos básicos y la modificación de los mismos (todo dejando de lado el apartado de presentación). Algunos miembros del grupo fueron asignados a estas tareas de implementación, otros a la elaboración de la documentación y una minoría continuó planificando los futuros módulos a implementar. Durante esta iteración se ralentizó un poco el desarrollo del sistema pero se consiguió documentación que serviría de consulta para posteriores dudas.
- Una vez finalizada esta primera fase, en una segunda iteración, se procedió a agregar el módulo del partido, las acciones individuales/grupales/pasivas, algunas funcionalidades como el cambio de equipo y el nuevo diseño del estilo de la página. Esta iteración duró algo más que la primera dada la complejidad de ciertas implementaciones y constituyó los elementos presentes en la demo del mes de marzo.
- Posteriormente, en una tercera iteración, que comenzó en el mes de abril, se incorporaron nuevas funcionalidades. Entre las principales podemos destacar la interacción durante el partido (acciones de partido), la recarga automática de éste (incorporación de Ajax a varios niveles), el sistema de mensajería privada y el de notificaciones. Asimismo se agregaron otros elementos como el chat durante el encuentro o las estadísticas del mismo. Por último, se generó un nuevo estilo para toda la web, mucho más atractivo que el anterior.

Todos estos contenidos se organizaron para estar presentes en la demo del día 28

# JUGADOR #12

Documentación adicional: RUP en el proyecto actual

Ingeniería del Software, 2013

de abril.

- En último lugar encontramos la cuarta iteración de la fase de construcción. Ésta se dedicó exclusivamente a reparación completa de errores, retoques finales del estilo y finalización de la documentación. Se incorporaron pequeños módulos como el tutorial o el cooldown de acciones de partido.

El resultado de todos estos cambios se presentó en la demo del día 22 de mayo.

## 2.4 Fase de transición

Durante esta última fase se llevaron a cabo todos los preparativos para la entrega final: desde reparar pequeños bugs encontrados en el producto hasta finalizar/actualizar los últimos documentos necesarios.

Asimismo, se generó el material necesario para el acto oficial del día 4 de junio y se presentó el proyecto en el mismo.

## 3. Futuros ciclos de desarrollo

Una vez finalizado este primer ciclo y el curso de Ingeniería del Software, el sistema será, casi con toda seguridad, llevado a un segundo ciclo de desarrollo al margen de la asignatura. En él se retomarán de nuevo con las 4 fases anteriores y se refinará y mejorará el producto inicial para garantizar una implementación suficientemente decente como para salir al mercado sin problemas.

## 4. Casos de uso

Se han creado varios documentos con los casos de uso asociados a la funcionalidad del sistema. En una primera versión presentaban los fragmentos de funcionalidad de forma muy básica pero posteriormente, en base a ejemplos reales de casos de uso, fueron refinados y detallados con más precisión.

Se optó por tener muchos casos de uso simples con el fin de mejorar su comprensión. Esto ha permitido poder unir varios de ellos para crear diferentes módulos de funcionalidad a implementar en el sistema. Otra opción hubiese sido crear pocos casos de uso pero con mucho más contenido y funcionalidad, pero era algo que resultaba bastante engorroso de consultar.

Podemos notar, además, que los casos de uso están muy relacionados con los diagramas de flujo de ejecución. Esto permite que, en caso de duda, se pueda consultar el diagrama (ya que es más directo) y, si hace falta información más detallada, se pase al caso de uso correspondiente.

Este tipo de documentación ha servido para dirigir el diseño de las iteraciones del producto, ya que se tomaban casos de uso restantes, se agrupaban en funcionalidades, y se acoplaban a las nuevas iteraciones de la fase de construcción.

## 5. Diagramas UML

La generación de estos diagramas comenzó bastante tarde. No se llevaron a cabo durante las etapas oportunas del desarrollo. Más bien, se podría decir que se generaron como

## JUGADOR #12

Documentación adicional: RUP en el proyecto actual

Ingeniería del Software, 2013

documentación adicional con el principal fin de aprender a crearlos. Por supuesto, también estaban a disposición de todo aquel que quisiera consultarlos, pero, francamente, no se han usado para mucho más que con fines de aprendizaje.

No obstante, de esto sí hemos aprendido que, realizados a tiempo, pueden ser de gran utilidad (al igual que toda la documentación más “visual” como diagramas de flujo). Con un simple vistazo permiten diferenciar claramente estructuras del sistema.

### 6. Controles de calidad

Dado que la fase de construcción comenzó de forma algo retrasada, no se llevaron a cabo reuniones específicas de control de calidad del código y documentación. En lugar de esto, dichas supervisiones fueron llevadas a cabo atendiendo a los siguientes puntos:

- Horas de laboratorio: se aprovechaban para juntar y revisar todo el código funcional hecho durante la semana. Se optó por este método ya que era la forma ideal de reunir a todo el grupo sin tener que fijar horario extraacadémicos.
- Reuniones de implementación: usadas principalmente para resolver posibles dudas de implementación, continuar con la construcción de funcionalidades y revisar el código creado.
- Revisión por parte de miembros del grupo: al margen de las horas de laboratorio y reuniones de implementación, dos componentes del equipo (Javier López y Manuel Artero) se han encargado de ir revisando continuamente las implementaciones realizadas. Para este apartado ha resultado especialmente útil el rol de “supervisor de código” descrito en un punto anterior.
- Documentación: no se han realizado reuniones estrictas para este tema. El principal control de calidad ha sido el aportado por el rol de “bibliotecario” (descrito también en un punto anterior). A pesar de existir un periodo más intenso de implementación en el cuál la documentación ha sido relegada a un segundo plano, se ha hecho un esfuerzo por retomar toda la documentación y actualizarla en la medida de lo posible.

### 7. Consideraciones adicionales

Esta sección incluye una breve opinión y conclusiones obtenidas de la aplicación del Proceso Unificado de Desarrollo al proyecto actual. Se dividirá en varios puntos para mejorar su comprensión:

- En primer lugar, mencionar que, al principio, tanto la fase de inicio como la de elaboración (en menor medida) parecían una pérdida de tiempo. Lo mismo ocurría con la documentación. El tiempo pasaba y los proyectos que seguían metodologías ágiles avanzaban rápidamente mientras que el nuestro permaneció “estancado” hasta principios de diciembre. Ahora bien, una vez el proyecto avanzaba se iba notando la importancia de estas fases iniciales. Dado que en los primeros meses se habían dejado muy claras las bases del producto, la fase de construcción pudo centrarse tan sólo en la implementación, sin necesidad de reuniones de última hora o preguntas acerca de cómo funciona el sistema (salvo algunas puntuales que siempre surgen). Asimismo, la documentación (en especial los diagramas UML)

## JUGADOR #12

Documentación adicional: RUP en el proyecto actual

Ingeniería del Software, 2013

servían como buena referencia a la hora de diseñar e implementar los diferentes módulos.

- Al principio, el modelo RUP resultaba algo caótico, hasta el punto de no saber si se estaba aplicando correctamente. Una vez que el proyecto avanzaba y se repasaban las bases de dicho proceso, quedaba cada vez más claro cómo funcionaba este modelo y se veían errores cometidos al aplicarlo por primera vez.
- Es muy posible que, de poder volver atrás en el tiempo, hubiésemos seleccionado una metodología ágil como proceso de desarrollo. Ésta nos hubiese permitido llegar a la presentación final con más funcionalidades sin tener que recortar tantas. Otra posible solución, en caso de futuros ciclos del proyecto, sería reducir el tiempo dedicado a las fases de inicio/elaboración o bien hacerlas más productivas. No obstante, este ciclo RUP ha servido, sin duda, para poder gestionar de manera correcta nuevos ciclos o procesos de desarrollo.
- Han quedado bastante claras las diferencias entre algunos procesos de desarrollo. Por ejemplo, en el caso de trabajar para una “startup” que necesite un producto cuanto antes, es muy posible que lo recomendable sea aplicar una metodología ágil para terminar el sistema antes de la fecha límite y después, una vez lanzado, se puede refinar toda la documentación existente con más calma. En el caso de proyectos de mayor envergadura o grupos de trabajo grandes (o en los que los programadores son desconocidos) es mejor aplicar el modelo RUP ya que permite generar unas bases muy sólidas para el proyecto y basarse en ellas para ir realizando pequeñas iteraciones de funcionalidades.
- Otro detalle importante aprendido es a no dejar de lado la documentación. A lo largo del proyecto las ideas iniciales van cambiando y con ellas las decisiones de diseño. Es muy complicado sentar las bases de un proyecto que permanezca immutable a lo largo de su desarrollo. Por estos cambios, es conveniente revisar periódicamente la documentación existente y actualizarla. Uno de nuestros errores ha sido dejar dicha documentación de lado una vez iniciada la fase de construcción. El problema llegó a la hora de enfrentarse a una documentación sin actualizar, algo que podía haberse evitado con pequeñas revisiones periódicas de la misma.

# JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

## Introducción

El presente documento contiene toda aquella información empleada durante el desarrollo del proyecto de una forma muy informal. Se adjunta con fines puramente referenciales. Se advierte que los presentes documentos pueden contener erratas así como expresiones coloquiales.

## Índice

1. Brainstorming: primera visión de las acciones y sus efectos
2. Búsqueda inicial del framework a emplear
3. Brainstorming: ideas para habilidades
4. Brainstorming: tabla del árbol de habilidades
5. Listado de posibles ampliaciones
6. Documento antiguo de organización

# JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

## 1. Brainstorming: primera visión de las acciones y sus efectos

### Acciones permanentes

- Factor de partido: aforo base
- Factor de partido: nivel del equipo
- Estadio: aforo máximo

### Acciones preparatorias (próximo partido)

- Factor de partido: aforo
- Factor de partido: nivel del equipo
- Factor de partido: 1/n goles de ventaja
- Bonus/penalización: efecto de las acciones de partido interactivo
- Bonus/penalización: ganancia de ánimo tras el partido
- Bonus/penalización: riesgo de detectar acciones ilícitas
- Bonus/penalización: obtener/perder gol de ventaja
- Bonus/penalización: efecto de los eventos generados por el jugador
- Bonus/penalización: efecto general de los eventos para el próximo partido

### Acciones inmediatas

- Recurso: dinero
- Recurso: influencias
- Recurso: ánimo
- Jugadores: anular penalización

### Habilidades pasivas

- Bonus: salario
- Bonus: ganancia de influencia
- Bonus: ánimo máximo
- Bonus: salir indemne de acciones ilícitas

## 2. Búsqueda inicial del framework a emplear

Después de haber revisado y descartado un montón de frameworks (cakePHP, DooPHP, Kohana, etc.) voy a dar una visión de los dos que, en mi opinión, son más susceptibles de ser elegidos para el proyecto. Conste de antemano que no hablo desde la experiencia ya que no he hecho nada grande en ninguno de los dos, solo que llevo todo el día leyendo opiniones de usuarios que han trabajado con ambos. Hay que tener en cuenta, antes de empezar a analizarlos, que lo que voy a hacer no es un “este framework es mejor que este otro”, sino un “este framework se adapta mejor que este otro al proyecto”.

### Características generales

Dedico este apartado a decir que hay cosas que ambos tienen en común, como el uso de caché para BBDD por ejemplo, comparten una arquitectura MVC (modelo-vista-controlador), soportan JQuery y Ajax, etc.

## JUGADOR #12

**Documentación adicional: documentación informal adicional** Ingeniería del Software, 2013

Ya no voy a hablar ni de PHP4 porque parto de la idea de que vamos a usar seguro PHP5. Solo decir que ambos frameworks trabajan con esta versión de PHP por lo que quedan empatados en este sentido.

### Bases de datos

Aquí creo que viene la primera carga de espacio extra. Yii trabaja con unas pocas bases de datos (usando Active Record, entre ellas MySQL) mientras que Symfony trabaja con muchas más. Como la idea principal no es tener un sistema que puede cambiar fácilmente de BBDD, ¿para qué todas esas clases extra si con MySQL nos llega?

### Comunidad/Documentación

En este punto está claro que Symfony tiene más usuarios aportando datos a la comunidad y documentación (lo cual no significa que Yii apenas tenga) pero creo que con lo que se pueda obtener de Google sobre Yii nos llegará para el proyecto.

### Tutorial

Os reto a leer las 15 primeras páginas de la guía de introducción rápida a Symfony (porque tiene muchas cosas como los bundles que, ¿realmente son necesarias?). Es mortal. Por otro lado, para Yii he encontrado algunos tutoriales mucho más sencillitos.

### Abstracción de la BBDD y ORM

Yii tiene una sintaxis más compleja que Symfony, lo cual no significa que sea menos potente. Si es cierto que Symfony tiene muchas más características, pero, volviendo a lo de siempre, ¿son todas necesarias? Gran parte de la carga del proyecto será (corgidme si es incorrecto) la concurrencia en BBDD, hacer selects y updates, etc. y eso lo podemos hacer sin problema.

### Auth, sesiones y validación

En este aspecto ambos están prácticamente igualados de cara a lo que queremos conseguir. Symfony tiene algunas ventajas más en cuanto a auth. pero ambos funcionan sin problema alguno.

### Debugging

Ambos tienen clases que permiten hacer un debug/profiling. No las he probado pero creo que lo importante es que las tienen, ya que otros frameworks no hacen uso de ellas.

### Curva de aprendizaje

Aquí no hay duda alguna, en todos lados he visto los mismos datos indiscutibles. El tiempo necesario para aprender a manejar correctamente Symfony es muy superior al de Yii o cualquier otro framework de categoría inferior.

### Necesidad del framework

Como llevo diciendo a lo largo de todo esto, ¿realmente necesitamos todas las características de Symfony? En casi todos lados coinciden en que Symfony está genial para proyectos de nivel grande-empresarial y Yii para proyectos de nivel básico-grande. Symfony ofrece muchas más posibilidades y herramientas que Yii, pero no vamos a hacer una web que trabaja a nivel mundial con 500 BBDD, vamos a hacer algo más sencillo en lo cual, de

## JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

hecho, para liberar carga de trabajo se usa el sistema de limitar los usuarios de un servidor y crear otros paralelos.

### Respaldos a ambos frameworks en la vida real

Voy a resaltar sobre todo el uso de Symfony por empresas grandes como Yahoo. En Yii he encontrado un montón de páginas de empresas más pequeñas pero que funcionaban a la perfección. Encontré también un ejemplo de una persona que daba su opinión sobre Yii (creo que en Alemania) porque lo habían usado para hacer la web de una compañía telefónica y actualmente estaba funcionando con 200k usuarios sin problemas.

### Peticiones por segundo (RPS)

He mirado un montón de benchmarks (sé que no son fiables, pero cuando todos coinciden en lo mismo..) sobre diferentes frameworks de PHP. En todos ellos Yii se encontraba muy por encima de Symfony. De hecho, Symfony estaba casi por debajo de todos los que entraban en los benchmarks (CodeIgniter, Yii, DooPHP, etc). Esto no significa que sea malo, sino que para las aplicaciones probadas en benchmarks (que son muy sencillitas) funcionaba mucho mejor Yii porque no cargaba tantas clases ni hacía tantas llamadas a funciones. Por supuesto, en una aplicación de nivel empresarial con muchísimos datos y parafernalias va a funcionar Symfony mejor que cualquier otro (o Zend).

Como ejemplo de benchmarks he elegido todos menos los de Yii y los de Symfony (por lo evidente. En el caso de Symfony por ejemplo, se decía que Symfony2 sería un 20% más rápido que Solar beta 3, pero podéis encontrar artículos por Google en los que se han probado los sistemas del benchmark y resulta que solo ha sido un 5% más rápido. PERO, todo esto teniendo las clases de Symfony2 precargadas y el sistema de Solar si iniciar, porque si empezaban a igualdad de condiciones resultaba que Solar era un 28% más rápido que Symfony2 (los sistemas están subidos en GIT para pruebas públicas). Algo así pasaba con Yii, que decían que sería un 75% más rápido pero lo habían probado con una de las betas de Yii y Symfony2 precargado.

### Recursos

Otro punto en común por internet es el consumo de recursos, sobre todo la RAM. Symfony consume mucho más (precisamente por precargar todas estas clases y que el consumo de ActiveRecord es menor que Doctrine) en memoria que Yii. Esto hace que, como dije antes, Symfony funciona muy bien en proyectos empresariales con servidores muy potentes, ya que al tener todo precargado y listo funciona más rápido que el resto. De cara al futuro y que el proyecto salga bien y se monte en un servidor, no creo que la economía esté como para pagar uno de esos, entonces doy otro punto a favor de Yii por el consumo de recursos.

### Programación

Como dicen por internet, Symfony (la versión 2 en menor medida que la 1) hace mucha "magia" o abstracción de la mayoría de funciones para facilitar y acelerar el desarrollo de aplicaciones y llega un punto en el que estás programando en "symfony" y no en PHP ni nada por el estilo. Yii es algo más parecido a CodeIgniter, más cercano a PHP (si, también más engorroso al verlo escrito). Otro tema que se discutía en foros es que al intentar cambiar o entender funciones o el propio core de Symfony, para la mayoría era mucho más complicado que crear o modificar las de CodeIgniter u otro framework más simple.

## JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

Algo que había que preguntarse antes de todo esto es si ambos frameworks sirven para el proyecto. Dado que Ogame estaba inicialmente hecho en PHP y HTML (y poco más) creo que sí, que hay de sobra con lo que los dos ofrecen.

También voy a resumir un poco por qué descartamos finalmente otros frameworks:

- **CodeIgniter:** tenía opiniones muy buenas en general , pero posiblemente se quedase en algo muy justo para lo que pretendíamos, o más difícil de ampliar.
- **DooPHP:** presume de ser el más rápido de todos los frameworks, y posiblemente hubiese sido una buena elección, pero era más reciente que los demás y no daba la sensación de estar implantado al 100%.
- **CakePHP:** ni me leí la documentación, pero no me llamó demasiado.
- **Kohana, Zend, etc.** : son frameworks del estilo de Symfony, Yii y CodeIgniter que hacen prácticamente lo mismo pero tenían en parte peores opiniones.

Bueno, creo que he matizado todos los puntos que necesitaba. Iba a enlazar los benchmark o algo de eso pero buscad por google que lo último que quiero ahora es volver a ver dónde estaban. A modo de resumen y a falta de probar Yii (que de momento solo lo ha hecho Rober) mi voto se retira de Symfony (sobre todo después de leer las primeras páginas del manual jaja). Creo que tiene mucho más de lo que vamos a necesitar y veo un gasto de tiempo/recursos innecesario en aprender determinadas cosas que se pueden hacer de formas más sencillas aunque menos elegantes. Ahora Dani te toca defender Symfony jaja

**Nota:** mañana subiré el manual de XAMPP que hoy con esto de los frameworks no lo hice.

<http://www.yiiframework.com/>

### 3. Brainstorming: ideas para habilidades

#### Empresario

- Sobornar jugadores del otro equipo para que jueguen mal
- Sobornar al árbitro o a los linieros,
- Conseguir a los mejores jugadores (fichajes)
- Ampliar estadio
- Pagar multas (fianzas de otros jugadores)
- Fichar nuevo entrenador
- Financiar espectáculo para el siguiente partido
- Invertir en bolsa (mejoras económicas)
- Recalificar terrenos (mejoras económicas)
- Mejorar Habilidades Directivas (mejora la influencia)
- Pactar con la mafia => versión destructiva del empresario
- Incentivo económico: subir la prima de los jugadores de tu equipo
- Apostar: subir o bajar apuestas sobre tu equipo. => ¿Tirar las apuestas de tu equipo?
- Tratos en el palco: pactar empates, victoria por un solo gol, etc.
- Conseguir un nuevo videomarcador en el estadio =>
- Mejorar las gradas del estadio => Versión reducida de ampliar el estadio (aforo++)
- Mejorar los palcos VIP => (aumenta tus influencias)

## JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

- Contratar azafatas para los palcos => (versión ampliada de mejorar palcos vips)
- Mejorar altavoces de animación en el estadio
- Conseguir mejora de comodidad en los asientos(Mejora la influencia)
- Tener mayor relación con el sector ultra(Mejora de pasión/fervor,cosas que pocos empresario tienen en la realidad)
- Conseguir mayor seguridad a aficionados del equipo contra ataques de la afición rival
- Comprar jaula/recinto donde los cánticos rivales pierdan fuerza

### Movedora

- Organizar fiestas antes de los partidos para agotar a los jugadores contrarios
- Organizar fiestas para animar al equipo (sin alcohol)
- Convocar a la prensa
- Introducir artículos favorables para el equipo en el periódico
- Organizar eventos filantrópicos para mejorar la imagen del equipo
- Inventar el baile del equipo
- Crear página oficial del equipo
- Crear emisora de radio del equipo
- Homenaje a jugador leyenda en la grada
- Cantante antes del partido (himno equipo)
- Comprar la mascota del equipo
- Organizar jornadas contra el racismo, dia del equipo o cosas asi, donde haya torneos(futbol,baloncesto,petanca...)
- Conseguir que empresa(casa de apuestas,mahou,empresa en internet...) patrocine al equipo ya sea en la publicidad del estadio o en camisetas(no se si puede ir aqui o al empresario por el hecho de conseguir dinero con el patrocinio...pero en muchos equipos tienen patrocinios que consiguen cosas para aficionados como cenas,bebidas en el estadio más baratas..)
- Realizar partido benéfico con famosos
- Crear una fundación para que aprendan a jugar al fútbol(quizá esto sea del empresario pero puede hacer que la usuario gane mayor influencia)

### Ultra

- Pinchar ruedas autobús equipo contrario
- Incendiar estadio...
- Secuestrar gato del árbitro =< OH GOD
- Pintarse del color del equipo
- Correr en ropa interior
- Tatuar el nombre del equipo en la frente
- Envolver a la mascota del equipo contrario en carteles del equipo propio (o el estadio pero eso sería una locura, o el autobús)
- Rapar a los otros jugadores por la noche
- Echar somníferos en el agua del otro equipo
- Robar las zapatillas a los otros jugadores
- Pintar el estadio del contrario de rosa
- Manipular el banquillo del otro equipo

## JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

- Realizar pancarta burlándose de la afición contraria
- Realizar mural de apoyo al equipo
- Lanzamiento de bengala
- Animación con botes de humo con los colores del equipo
- Realizar nuevo cántico al equipo que anime más al equipo
- Realizar pegatinas para plagar los alrededores del estadio rival
- Fabricar banderas grandes y pequeñas para la animación
- Realizar disturbios en las inmediaciones del estadio
- Perseguir aficionados del otro equipo robandoles bufandas,bombos,megáfonos...
- Salir de espontáneo al campo en pelotas.
- Destrozar el local de una peña rival.
- Organizar peleas entre peñas.
- Emborracharse para aumentar el fervor.

### 4. Brainstorming: tabla del árbol de habilidades

Ultra

NIVEL	ACCIÓN	DESCRIPCIÓN	MORALIDAD LEGALIDAD	ÁMBITO
Bajo	Gritar Ánimos	Es casi inevitable si quieres a tu equipo	Honrada	Individuo
Bajo	Gritar Insultos	¿Insultos? No, son críticas constructivas	Honrada	Peña
Bajo	Beber	¿Te estas quedando sin energías? Para eso esta el alcohol	Honrada	Individuo
Bajo	Ola	Solo hay que sincronizarse	Honrada	Peña
Bajo	Unirse a los Disturbios	No importa el por que, ¡Pelea!	Ilícita	Individuo
Bajo	Preparar Pancarta de Ánimo	Que sepan que les apoyas incondicionalmente	Honrada	Individuo

## JUGADOR #12

Documentación adicional: documentación informal adicional      Ingeniería del Software, 2013

Medio	Vender en la Reventa	Buena forma de aumentar tus ingresos y ayudar a quien no la compró a tiempo	Ilícita	Individuo
Medio	Pintarse del color del equipo	La mejor forma de usar media docena de litros de pintura	Honrada	Individuo
Medio	Preparar Pancarta de Insultos	\$%#&&! Y su &\$%@ madre!!!	Honrada	Individuo
Medio	Amenazar al Árbitro	No hace falta pegarle, solo que crea que vas a hacerlo	Ilícita	Individuo
Medio	Consumir Drogas	Un plus de energía cuando lo necesites	Ilícita	Individuo
Medio	Preparar Cóctel Mólotov	No es que vayas a usarlo, es un experimento de química.	Ilícita	Individuo
Medio	Robar objetos	¿Por qué comprar equipamiento de partido cuando el otro equipo te lo trae hasta la puerta?	Ilícita	Peña
Medio	Llevar banderas	Nada mejor que un estadio entero ondeando banderas de tu equipo	Honrada	Peña
Alto	Tatuarse el nombre del equipo	Desaconsejado para chaqueteros	Honrada	Individuo

## JUGADOR #12

Documentación adicional: documentación informal adicional      Ingeniería del Software, 2013

Alto	Cocinar MDMA	Dinero para ti y fervor para tus compañeros	Ilícita	Peña
Alto	Tirar bengala	¿Que además sirven para iluminar?	Ilícita	Individuo
Alto	Espontáneo Desnudo	Es una buena forma de llamar la atención	Ilícita	Individuo
Alto	Introducir Armas	Mejor llevarlas y no necesitarlas que necesitarlas y no llevarlas	Ilícita	Individuo
Alto	Crear nuevo cántico	Un canto que inspire a la afición y al equipo	Honrada	Peña
Muy Alto	Romper Pierna a la Estrella	No, él se tropezó con mi bate de béisbol	Ilícita	Individuo
Muy Alto	Provocar incendio	Si el partido no va como esperabas mejor que no termine	Ilícita	Individuo
Muy Alto	Secuestrar al hijo del árbitro	Puedes pedir un rescate, y además el próximo partido irá como la seda	Ilícita	Peña
Muy Alto	Rapar a los Jugadores mientras duermen	A ver si les siguen contratando para anuncios ahora	Ilícita	Individuo
Muy Alto	Quemar autobús adversario	No hay mejor forma de decir que no les quieres en tu estadio	Ilícita	Peña

## JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

Ultimate	Asaltar la cárcel	Entra en la cárcel con un grupo armado y libera a tus compañeros de afición	Ilícita	Afición
Ultimate	Fundar grupo fanático	Conviértete en un líder de los hinchas más radicales	Honrada	Individuo
Ultimate	Secuestrar jugador estrella	No solo le quitas de en medio, servirá de advertencia para los demás	Ilícita	Peña

### Empresario

NIVEL	ACCIÓN	DESCRIPCIÓN	MORALIDAD LEGALIDAD	ÁMBITO
Bajo	Pagar multas (fianzas de otros jugadores)	Resuelve los pleitos legales de otros jugadores	Honrada - legal	Peña
Bajo	Invertir en bolsa	Consigue líquido para poder financiar tus acciones. ¿Vía rápida? la bolsa	Honrada - legal	Peña
Bajo	Meter mano en la cotización de las casas de apuestas	Invirtiendo dinero en el mercado de apuestas puedes conseguir recuperar con creces el dinero que apuestes para el próximo partido	Ilícita - ilegal	Peña
Bajo	Comprar un	Haz pequeñas	Honrada - legal	Peña

## JUGADOR #12

Documentación adicional: documentación informal adicional      Ingeniería del Software, 2013

	nuevo video marcador para el estadio	inversiones para mejorar el estadio		
Medio	Incentivo económico a los jugadores	Existe una relación directamente proporcional entre sueldo - rendimiento en el campo ¡aprovéchala!	Honrada - legal	Peña
Medio	Financiar espectáculo para el siguiente partido	Un buen espectáculo pirotécnico gustará a los aficionados. ¿Quién dijo que un empresario no podía ser carismático?	Honrada - legal	Peña
Medio	Mejorar tus habilidades directivas	Es hora de escalar puestos en el sistema capitalista para ganar más dinero y poder hacer inversiones mayores	Honrada - legal	Individual
Medio	Mejorar las gradas del estadio	Es posible mejorar el estadio sin inversiones de grandes sumas de dinero	Honrada	Peña
Medio	Inversión en seguridad: Contratar mejores guardias de seguridad	Mantén un ojo en los ultras rivales, toda precaución es poca.	Honrada	Peña

## JUGADOR #12

Documentación adicional: documentación informal adicional      Ingeniería del Software, 2013

Medio	Comprar altavoces para el estadio	Haz pequeñas inversiones para mejorar el estadio	Honrada	Peña
Alto	Contratar a un bufete de abogados	Mejor prevenir que curar: cuídate las espaldas con los mejores abogados de la ciudad	Honrada	Peña
Alto	Mejorar los palcos VIP con sillones del Starbucks y azafatas en ropa interior	Aprende a jugar al juego de palcos; un poco de vino, buena compañía y un buen partido de fondo es el escenario ideal para firmar acuerdos importantes	Honrada	Peña
Alto	Conseguir un contrato de patrocinio para el equipo	Consiguiendo contratos para el equipo conseguirás fondos para financiar tus futuras acciones e influencias dentro del club	Honrada - legal	Peña
Alto	Sobornar a un jugador del equipo contrario para que se marque en propia puerta	Lleva a la práctica el sutil arte del soborno y asegúrate una pequeña ventaja	Ilícita - ilegal	Peña
Muy alto	Pactar con la mafia	Si quieres aplastar a los hinchas rivales, mejor hacerlo con estilo y	Ilícita - ilegal	Peña

## JUGADOR #12

Documentación adicional: documentación informal adicional      Ingeniería del Software, 2013

		profesionalidad		
Muy alto	Recalificar terrenos	Primer paso para conseguir ese estadio con el doble de aforo, además de sacar tajada de la situación	Ilícita - ilegal	Afición
Muy alto	Sobornar al juez de línea	Un fuera de juego en el momento oportuno puede salvar el partido.	Ilícita - ilegal	Peña
Muy alto	Fichar a una estrella del fútbol	Ese jugador aportará profundidad al juego del equ.... ¡Tonterías; venderá camisetas!	Honrada - legal	Afición
ULTIMATE	Pactar la victoria del próximo partido: comprar el partido	Todo el mundo tiene un precio. Métete el siguiente partido en el bolsillo	Ilícita - ilegal	Peña
ULTIMATE	Comprar estadio	Duplica el aforo máximo del estadio. Requiere que previamente se hayan recalificado los terrenos	Honrada	Afición

### Movedora

NIVEL	ACCIÓN	DESCRIPCIÓN	MORALIDAD LEGALIDAD	ÁMBITO
Bajo	Publicitar Equipo en las	Dale un poco de bombo al próximo	Honrada - legal	Individual

## JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

	redes sociales	encuentro. Cuanta más gente lo siga, mejor		
Bajo	Escribir artículos de opinión para periódicos digitales	¿Qué mejor forma de ganar un dinerillo que escribiendo sobre tu equipo?	Honrada - legal	Individual
Bajo	Mejorar opinión pública del club: campaña en blogs y foros	Es importante que nuestro equipo cuente con una buena publicidad	Honrada - legal	Peña
Bajo	Administrar un foro para aficionados del equipo	Hazte un nombre en la comunidad digital.	Honrada - legal	Individual
Medio	Organizar cena entre los jugadores del equipo	Las cenas de hermandad nunca sobran en un equipo. Cuanto más unidos estén los jugadores, mejor	Honrada - legal	Peña
Medio	Escribir artículos favorables del equipo en los periódicos	Un artículo en un periódico en papel llegará a muchísima gente	Honrada - legal	Individual
Medio	Encontrar una mascota para el equipo	Encontrar una mascota carismática es sinónimo de atraer más aficionados al campo	Honrada - legal	Peña
Medio	Inventar baile para la mascota del	Un movimiento original para que nuestra mascota	Honrada - legal	Peña

## JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

	equipo	anime en el partido. ¡Publicítalo en Youtube!		
Alto	Crear la página oficial del equipo	Un gran club necesita una gran página web a su altura	Honrada - legal	Afición
Alto	Atacar la página oficial del equipo contrario	Hackea la página oficial del equipo contrario	Ilegal - ilícita	Peña
Alto	Hacer llamamiento para llenar el estadio por un programa de radio	Convence a los hinchas de la importancia de llenar el estadio en el próximo encuentro gracias a la radio	Honrada - legal	Peña
Alto	Iniciar campaña de difamación por la red contra un jugador rival	Los sentimientos hacia un equipo están directamente ligados con los sentimientos hacia sus jugadores: ataca a sus jugadores	Ilegal - ilícita	Peña
Alto	Redactora de las crónicas del equipo en periódico digital	Deja atrás los días en los que escribías sobre tu equipo como una freelance.	Honrada - legal	Individual
Muy alto	Organizar un homenaje a un jugador leyenda en el estadio	Es importante una percepción positiva de cómo este club trata a sus viejas glorias	Honrada - legal	Peña
Muy alto	Contratar	Miles de	Honrada - legal	Peña

## JUGADOR #12

Documentación adicional: documentación informal adicional      Ingeniería del Software, 2013

	músicos famosos para poner música al himno del equipo	gargantas cantarán al unísono en el estadio si das con una melodía pegadiza		
Muy alto	Llevar un programa de debates en la radio con ataques constantes a los equipos rivales	Las campañas de difamación están muy bien... pero mejor un ataque constante para mermar el ánimo rival	Illegal - ilícita	Peña
Muy alto	Organizar evento social para mejorar la imagen pública del club	Mejora la imagen pública de tus jugadores de una forma sencilla pero eficaz: que lleven regalitos a niños enfermos al hospital. nos dará espacio publicitario en los telediarios asegurado	Honrada	Peña
ULTIMATE	Presentadora de las noticias	Llega al medio de difusión máximo: la tele. Redacta el "corazón corazón de los hombres", deportes cuatro	Honrada	Peña
ULTIMATE	Acabar públicamente con un jugador	Saca a la luz trapos sucios, inventa, difama; lo que sea con tal de acabar públicamente con la imagen de la estrella rival	Illegal - ilícita	Peña

## JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

ULTIMATE	Organizar campaña solidaria en África patrocinada por el equipo	Funda una escuela para los más necesitados y vistelos con los colores del equipo para que posen para la foto. No existe mejor publicidad.	Honrada	Peña
ULTIMATE	Politicizar el siguiente encuentro	Lleva la tensión del siguiente partido a otro nivel: haz de él una cuestión de estado	Illegal - ilícita	Peña

### 5. Listado de posibles ampliaciones

1. Peñas dobles
2. Apoyar a jugadores en concreto (foco en jugadores concretos)
3. Varias competiciones
4. Añadir jugadores topo (o troll si queréis) que permitan destruir una afición desde dentro. Esto representa una expansión del juego, ya que habría que dotar a una afición de herramientas para detectar a jugadores “topo”, herramientas para obligarlo a irse a otra afición y herramientas para hacer el boicot en sí.  
¿Esta “extensión” no entraría en colisión con la habilidad “destructiva” del ultra?
5. Implementar “grados” de evento, esto es: una vez pagados los requerimientos de una acción, se pueda invertir más para mejorar el resultado.
6. Complicar el uso de fervor haciendo más efectiva la habilidad en función del porcentaje (total o parcial) invertido en la misma.
7. Contador de días que un jugador no se conecta y que después de X días envíe un correo al jugador preguntándole si quiere darse de baja o no.

### LOGROS

1. Lotina => Desciende a varios equipos de forma simultánea
2. Chaquetero => Cambia 3 veces de afición en una misma semana
3. Asciende a un equipo

### 6. Documento antiguo de organización

#### 1. La metodología de trabajo

Los pilares básicos de nuestra organización son dos. El responsable o el jefe del grupo y las reuniones periódicas semanales en las que todos los miembros del grupo participan en la supervisión del estado del proyecto.

## JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

### 1.1. El responsable

El proyecto ha seguido desde el primer momento una **estructura centralizada**.

Manuel Artero Anguita desde el primer día fue designado responsable del grupo. Siendo más exactos el organizador y coordinador del grupo, puesto que todos somos responsables, en conjunto, de nuestro trabajo.

Las tareas del responsable son:

1. Tener una visión global del proyecto.
2. Organización del proyecto
3. Determinar las metas y objetivos a corto y largo plazo
4. Toma de las decisiones determinantes.
5. Presentación de las entregas mensuales al profesor.,
6. Preparación del acta de las reuniones semanales de acuerdo a las exigencias del proyecto.
7. El reparto de las tareas del grupo

### 1.2. Las reuniones periódicas semanales

Aunque el proyecto nos apasiona y nos gustaría pasar mucho más tiempo del que tenemos desarrollándolo, todos tenemos otras obligaciones que cumplir y es necesario trabajar por separado y luego juntar los resultados.

Las reuniones semanales duran casi todo el día. Comienzan los martes en el laboratorio de IS y acaban casi cuando se cierra la facultad.

Durante este tiempo, ponemos en común lo que hemos trabajado durante la semana, lo revisamos y si hace falta lo ensamblamos. También debatimos el estado del proyecto. Se ponen en común las dudas; propuestas o ideas del proyecto para debatirse en grupo.

**Fijamos cuáles son los objetivos** para la semana siguiente y se reparten las tareas intentando que sea lo más equilibrados posibles entre los miembros del grupo. Haremos especial hincapié en el reparto de tareas más adelante

### El reparto de tareas

Lo más importante en la organización es el reparto de tareas de manera equilibrada entre todos los miembros del grupo. También, que los miembros del grupo sepan qué tareas tienen asignadas el resto de compañeros, ya que, muchas veces las tareas se relacionan unas con otras.

El responsable es quién decide el reparto de tareas durante la semana. Nos las comunica durante la reunión y cuelga en el proyecto de GitHub un documento con el resumen de tareas para la semana, quién hace cada una y una pequeña descripción.

El profesor, puede consultar este [reparto de tareas desde nuestra proyecto en github](#).

No a todos los miembros del grupo se les asigna la misma tarea, excepto las tareas de responsable, todas las demás tareas van rotando entre los miembros del grupo.

Aprendiendo así la necesidad de comentar el código en el proyecto, ya que es necesario para el que venga después entienda lo que hemos hecho.

# JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

## Explicación del documento de organización

El documento está estructurado por meses. Dentro de cada mes se encuentran dos apartados: la perspectiva general, en la que se puede apreciar la totalidad de la funcionalidad que se pretende conseguir en cada mes; y la organización detallada por personas, en la que queda perfectamente detallado qué tiene que hacer cada miembro del grupo.

Un ejemplo de esta organización sería:

### Diciembre

#### **Perspectiva general**

Controladores

- Usuarios: Alex, Marina, Rober
- Registro: Alex
- Equipos: Marina, Sam
- Habilidades: Arturo, Dani
- Acciones: Dani, Pedro, Marcos
- Partidos: Marina, Arturo

Modelos

- Usuarios: Rober
- Recursos: Alex
- Equipos: Sam
- Clasificacion: Marcos
- Habilidades: Arturo
- Acciones individuales: Pedro
- Acciones grupales: Marcos
- Acciones turno: Marcos
- Desbloqueadas: Alex
- Partidos: Sam
- Participaciones: Dani

#### **Organización por personas**

1. **Alex:** Registro.index; Usuarios.cuenta; Desbloqueadas(+); Recursos(+)
2. **Arturo:** Habilidades.adquirir; Partidos.index; Partidos.asistir; Habilidades(+)
3. **Dani:** Habilidades.index; Habilidades.ver; Acciones.usar; Participaciones(+)
4. **Marcos:** Clasificacion(+); AccionesGrupales(+); AccionesTurno(+);  
Acciones.expulsar
5. **Marina:** Usuarios.perfil; Usuarios.ver; Partidos.previa; Equipos.index
6. **Pedro:** Acciones.index; AccionesIndividuales(+); Acciones.ver; Acciones.participar

## JUGADOR #12

Documentación adicional: documentación informal adicional Ingeniería del Software, 2013

7. **Rober:** Usuarios.index; Usuarios.cambiarClave; Usuarios.cambiarEmail; Usuarios(+)

8. **Sam:** Equipos.ver; Equipos.cambiar; Equipos(+); Partidos(+)

En este mes, por ejemplo, vemos que se han repartido las tareas de la implementación básica del proyecto.

Se ha indicado en la perspectiva general los grupos temporales de personas que implementan funcionalidad similar; por ejemplo Marina y Sam les corresponde implementar funcionalidad en el controlador de equipos.

Además, se considera que siempre que un miembro del grupo tenga que hacer una función en el controlador (indicada con una notación de controlador.funcionalidad) le corresponderá implementar también su vista.

Por último, los modelos están indicados con un +.

### Otras organizaciones que no surtieron efecto

Al principio del curso se hizo un diagrama de gantter para planificar el proyecto. Sin embargo se descartó su uso para la coordinación del grupo. He aquí las razones por las que se desestimó como herramienta de organización:

1. Todos los editores de gantter que encontramos eran, sino difíciles, tediosos de manejar.
2. Requería un tiempo excesivo: *plasmar la organización en gantter requería más tiempo que la planificación en sí.*
3. Al principio se hizo un diagrama global que abarcaba todo el año con ésta herramienta. Si bien fue útil para hacernos una idea del tiempo que teníamos y nos hizo pensar a largo plazo con nuestro proyecto, Cuando terminamos el esquema no era intuitivo, era muy grande y no permitía tener una visión global y modificarlo era tan tedioso como hacerlo desde el comienzo.

En gantter si poníamos las tareas específicas de la persona perdíamos la visión global. Pero si poníamos las tareas generales la herramienta no servía para el reparto de tareas.