



DEPARTAMENTO  
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA



Departamento de Computación,  
Facultad de Ciencias Exactas y Naturales,  
Universidad de Buenos Aires

# Trabajo Práctico 2

ISW II

Primer Cuatrimestre de 2015

Apellido y Nombre	LU	E-mail
Almansi, Emilio Guido	674/12	ealmansi@gmail.com
Gasperi Jabalera, Fernando	56/09	fgasperijabalera@gmail.com
Russo, Christian	679/10	christian.russo8@gmail.com
Tagliavini Ponce, Guido	783/11	guido.tag@gmail.com

# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Casos de uso</b>	<b>4</b>
<b>3. Planificación</b>	<b>5</b>
3.1. Iteraciones . . . . .	5
3.1.1. Fase de iniciación . . . . .	5
3.1.2. Fase de elaboración . . . . .	5
3.1.3. Fase de construcción . . . . .	5
3.1.4. Fase de transición . . . . .	6
3.2. Alcance de casos de usos de la primera iteración . . . . .	6
3.3. Tareas CU Primera iteración . . . . .	7
3.4. Dependencias entre tareas . . . . .	8
3.5. Detalle Primera iteración . . . . .	8
3.6. Plan de Proyecto . . . . .	10
<b>4. Análisis de riesgos</b>	<b>12</b>
<b>5. Atributos de calidad</b>	<b>13</b>
5.1. Disponibilidad . . . . .	13
5.2. Modificabilidad . . . . .	14
5.3. Usabilidad . . . . .	14
5.4. Performance . . . . .	15
5.5. Seguridad . . . . .	16
<b>6. Arquitectura</b>	<b>17</b>
6.1. Conectores propios . . . . .	17
6.1.1. Notación . . . . .	17
6.1.2. Holy Connector . . . . .	17
6.1.3. Asynchronous Holy Connector . . . . .	18
6.1.4. Holy Video Connector . . . . .	18
6.1.5. Holy Data Connector . . . . .	19
6.2. Diagrama . . . . .	19
6.2.1. Arquitectura orientada a datacenters . . . . .	19
6.2.2. Descripción general . . . . .	20
6.2.3. Repositorios distribuidos . . . . .	25
6.3. Atributos de Calidad vs Arquitectura . . . . .	26
<b>7. Conclusión</b>	<b>28</b>

## 1. Introducción

El presente trabajo desarrolla la organización de un proyecto basado en el TP1 pero de mucho mayor alcance, donde la aplicación de desafíos deportivos basados en simulaciones de partidos de básquet se extiende para incluir nuevos deportes y múltiples modos de participación, además de estar desarrollado con la idea de realizar una eventual expansión a escala global.

En las secciones siguientes, detallamos los casos de uso de los subsistemas más relevantes, así como un cronograma para el desarrollo de los mismos en múltiples iteraciones. Para la primera iteración en particular, realizamos una descripción más detallada de los casos de uso que la conforman, así como su alcance y su descomposición en una serie de tareas. Cada tarea lleva una estimación de tiempo de desarrollo en horas hombre, concluyendo con un diagrama de Gantt para visualizar su distribución temporal durante el desarrollo del sistema general.

Por último, realizamos un análisis de riesgo sobre el proyecto y sus instancias de desarrollo, detallando la probabilidad, el nivel de impacto y exposición de cada riesgo, así como posibles estrategias de mitigación y un plan de contingencia ante la situación adversa descripta.

## 2. Casos de uso

A continuación incluimos los principales casos de uso cubriendo la mayor parte de la funcionalidad determinada por los requerimientos del proyecto. Los mismos se encuentran ordenados de acuerdo al subsistema al cual corresponden.

### ■ Aplicaciones Web/Móvil.

- Creando nuevo usuario.
- Iniciando sesión de usuario.
- Ingresando datos de tarjeta / cuenta corriente.
- Visualizando desafíos disponibles.
- Dando de alta un nuevo desafío.
- Ingresando a un desafío.
- Armandando equipo para desafío.
- Visualizando partido de un desafío.
- Visualizando saldo.
- Visualizando premios ganados.

### ■ Servidor de Juego.

- Guardando datos de nuevo usuario.
- Autenticando datos de usuario.
- Guardando datos cifrados de tarjeta / cuenta corriente de usuario.
- Transmitiendo un partido de desafío.
- Creando desafío.
- Cobrando apuestas de desafío.
- Cobrando cuota de participación en desafío.
- Acreditando ganancias por desafío ganado.
- Acreditando premios al usuario.

### ■ Sistema de Transmisión de Partidos.

- Transmitiendo datos de una simulación en proceso.
- Transmitiendo render de una simulación en proceso.
- Transmitiendo un partido en vivo.

### ■ Sistema de Pagos.

- Validando datos de tarjeta / cuenta corriente.
- Debitando monto a una tarjeta / cuenta corriente.
- Acreditando monto a una tarjeta / cuenta corriente.

### 3. Planificación

#### 3.1. Iteraciones

Dividimos los CU en distintas iteraciones, teniendo en cuenta diversos factores. El criterio principal fue seguir un estilo de desarrollo horizontal, en el sentido de desarrollar varios subsistemas en simultáneo, en contraposición a un desarrollo por bloques, en el cual un subsistema no empieza a desarrollarse hasta que gran parte de otro esté completado. Creemos que este estilo nos permitirá poder modificar más fácilmente los componentes, a medida que tengamos un mejor conocimiento sobre las interacciones entre ellos.

Concretamente, en la primera iteración busca desarrollar mínimamente la Aplicación Web/Móvil, el Servidor de Juego y el Sistema de Transmisión de Partidos, y algunas interacciones entre la Aplicación Web/Móvil y el Sistema de Transmisión de Partidos.

En segundo lugar, tuvimos en cuenta el análisis de riesgos realizado, priorizando casos de uso que evalúen o ataquen en alguna medida a estos riesgos, no necesariamente proveyendo una solución. Concretamente, los riesgos encontrados son la confiabilidad de los motores gráficos, la confiabilidad de los clientes (en términos de calidad de hardware y conexión), y la seguridad de los datos almacenados.

En la primera iteración comenzamos a probar el motor gráfico 2D tratando, apuntando a comenzar a evaluar el funcionamiento de dicho motor, y analizamos y desarrollamos estrategias para garantizar la seguridad de los datos de usuarios y desafíos.

##### 3.1.1. Fase de iniciación

Esta fase queda representada por el enunciado del TP2.

##### 3.1.2. Fase de elaboración

###### Primera iteración

1. Guardando datos de nuevo usuario.
2. Visualizando partido de un desafío.
3. Transmitiendo datos de una simulación en proceso.
4. Creando desafío.

###### Segunda iteración

1. Iniciando sesión de usuario.
2. Autenticando datos de usuario.
3. Ingresando datos de tarjeta / cuenta corriente.
4. Guardando datos cifrados de tarjeta / cuenta corriente de usuario.

##### 3.1.3. Fase de construcción

###### Tercera iteración

1. Dando de alta un nuevo desafío.
2. Ingresando a un desafío como participante.
3. Transmitiendo un partido de desafío.

4. Guardando datos de nuevo usuario.
5. Armandando equipo para desafío.

#### **Cuarta iteración**

1. Visualizando saldo de usuario.
2. Cobrando cuota de participación en desafío.
3. Acreditando premios al usuario.
4. Visualizando desafíos disponibles.

### **3.1.4. Fase de transición**

#### **Quinta iteración**

1. Cobrando apuestas de desafío.
2. Visualizando premios ganados.
3. Transmitiendo render de una simulación en proceso.
4. Validando datos de tarjeta / cuenta corriente.

#### **Sexta iteración**

1. Transmitiendo un partido en vivo.
2. Debitando monto a una tarjeta / cuenta corriente.
3. Acreditando monto a una tarjeta / cuenta corriente.
4. Acreditando ganancias por desafío ganado.

## **3.2. Alcance de casos de usos de la primera iteración**

### **Guardando datos de nuevo usuario.**

El Servidor de Juego recibió los datos de un usuario a suscribir, y debe almacenarlos en algún medio físico, garantizando su disponibilidad, y haciéndolo de forma segura.

Los datos de un usuario podrían ser almacenados en un único servidor o en un conjunto de servidores. Dependiendo de la escala que pretenda alcanzar la primer versión del sistema, optaremos por un almacenamiento centralizado o distribuido. Si la cantidad de usuarios es del orden de los millones, una base de datos relacional y centralizada puede ser suficiente. Si no, es posible que sea necesaria una base de datos no relacional y distribuida.

La replicación de datos es un factor a considerar, relacionado con la persistencia de estos datos. Observar que la necesidad de replicación depende del volumen de los datos, sino del volumen de pedidos al servidor. En efecto, por más que todos los usuarios sean almacenables en forma localizada en una sola máquina, debemos asegurar que estén disponibles, dada la cantidad de accesos a esa información. Para esto podemos replicar una misma base de datos, sobre muchas máquinas.

Con respecto a los datos concretos almacenados, en principio serán sólo los personales, aunque en el futuro también podría utilizarse y guardarse información sobre preferencias y gustos, para la visualización personalizada de publicidad.

Como la seguridad es un atributo de calidad prioritario, debemos almacenar todos estos datos en forma segura, aunque debe invertirse cierto tiempo en estudiar otras alternativas.

### **Visualizando partido de un desafío.**

La Aplicación Web/Móvil debe mostrar la ejecución de una simulación (si el desafío es de modo simulación) o la transmisión de un partido real (si es de modo fantasía). En la primera iteración, sólo nos interesa visualizar una simulación 2D.

Debe analizarse las tecnologías a utilizar para desarrollar los clientes, dado que para crecer en cantidad de usuarios, necesitamos soportar determinadas plataformas masivamente usadas, como por ejemplo Android.

### Transmitiendo datos de una simulación en proceso.

El Servidor de Juego debe enviar a todos los clientes visualizando cierto desafío, el flujo de datos que produce la simulación asociada. Este envío de datos se da únicamente en el caso que un usuario opta por la renderización 2D, y además su conexión de red no soporta la bajada de un flujo de video, que requiere un gran ancho de banda. En estos casos, los datos enviados corresponden a una descripción del transcurso de la simulación, que es renderizada localmente por el usuario.

En la primera iteración definiremos interfaces sencillas para la comunicación entre un cliente y un servidor. El cliente simplemente pedirá los datos de una simulación, y el servidor transmitirá un flujo de datos correspondiente a una simulación artificial.

### Creando desafío.

El Servidor de Juego recibió un nuevo desafío, y lo debe almacenar en algún medio físico, garantizando su disponibilidad, y haciéndolo de forma segura.

En esta primera iteración nos interesa decidir cuestiones de arquitectura, disponibilidad y seguridad, similares a las dichas en el alcance del CU *Guardando datos de nuevo usuario*.

## 3.3. Tareas CU Primera iteración

A continuación se detallan las tareas diagramadas para los casos de uso incluidos en la primera iteración con su respectiva estimación de horas hombre.

CU1	Guardando datos de nuevo usuario.	114h
T1.1	Definir motor de base de datos a utilizar para datos de usuario, contemplando la escalabilidad.	8h
T1.2	Definir método de encriptación para contraseñas y datos sensibles.	10h
T1.4	Definir conjunto de datos a persistir por cada usuario.	16h
T1.5	Diseñar base de datos de usuarios.	10h
T1.6	Implementación de CU1	40h
T1.7	Testing de CU1	30h
CU2	Visualizando partido de un desafío.	161h
T2.1	Investigar tecnologías web a utilizar para el cliente web.	16h
T2.2	Investigar tecnologías a utilizar para los clientes móviles.	16h
T2.3	Desarrollar stubs de las aplicaciones web y móviles.	8h
T2.4	Integrar motor de gráficos 2D a la aplicación web.	18h
T2.5	Integrar motor de gráficos 2D a la aplicación móvil.	18h
T2.6	Implementación de CU2.	45h
T2.7	Testing de CU2	40h
CU3	Transmitiendo datos de una simulación en proceso.	103h
T3.1	Investigar plataforma para el servidor de transmisión.	10h
T3.2	Definir la interfaz de comunicación con el servidor.	10h
T3.3	Desarrollar stub del servidor.	8h
T3.4	Desarrollar cliente simple de prueba.	10h
T3.5	Implementación de CU3.	35h
T3.6	Testing de CU3	30h

CU4	Creando desafío.	99h
T4.1	Definir motor de base de datos a utilizar para datos de desafío, contemplando la escalabilidad.	8h
T4.2	Definir conjunto de datos a persistir por cada desafío.	10h
T4.3	Investigar alternativas para la persistencia segura de datos del desafío.	10h
T4.4	Diseñar base de datos de desafíos.	16h
T4.5	Implementación de CU4	30h
T4.6	Testing de CU4	25h

### 3.4. Dependencias entre tareas

Las tareas involucradas en cada caso de uso no son necesariamente independientes, en el sentido de que alguna de ellas puede requerir que otra sea finalizada para comenzar con su desarrollo. En los siguientes diagramas, incluimos una representación gráfica de las dependencias entre las diferentes tareas para cada caso de uso.

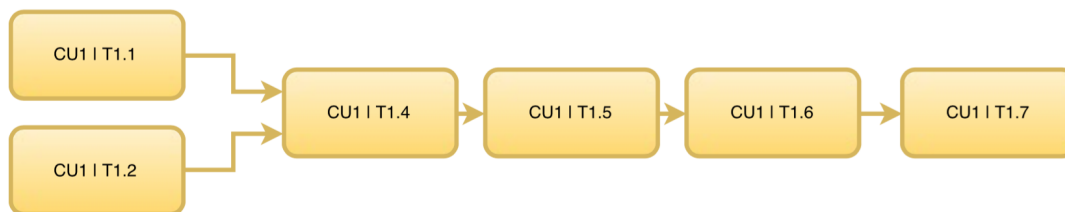


Figura 1: Diagrama de dependencias entre las tareas del CU nro 1.

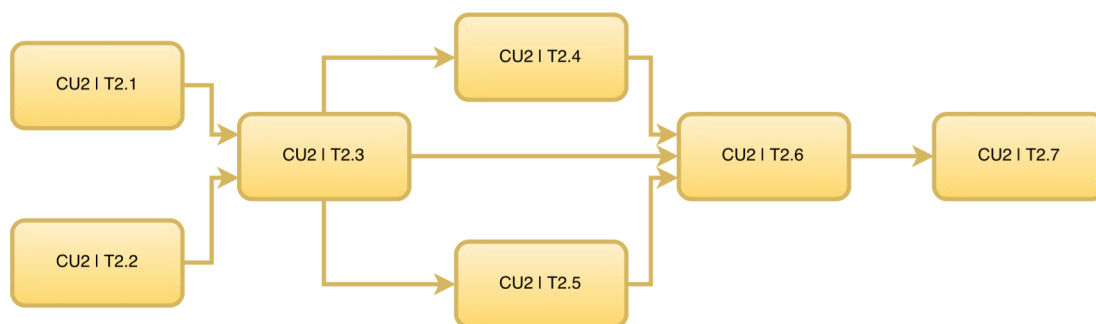


Figura 2: Diagrama de dependencias entre las tareas del CU nro 2.

### 3.5. Detalle Primera iteración

- Identificación: E1



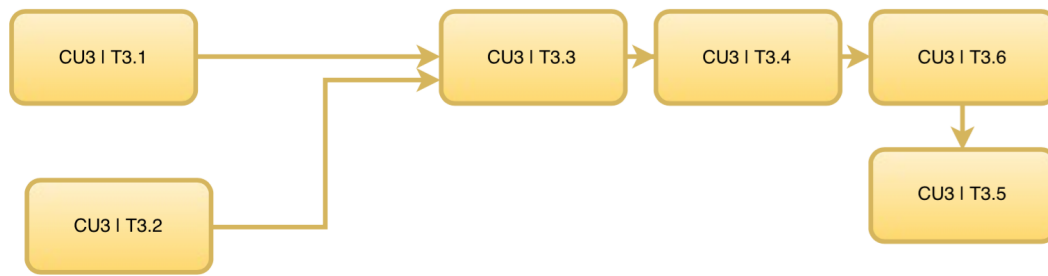


Figura 3: Diagrama de dependencias entre las tareas del CU nro 3.

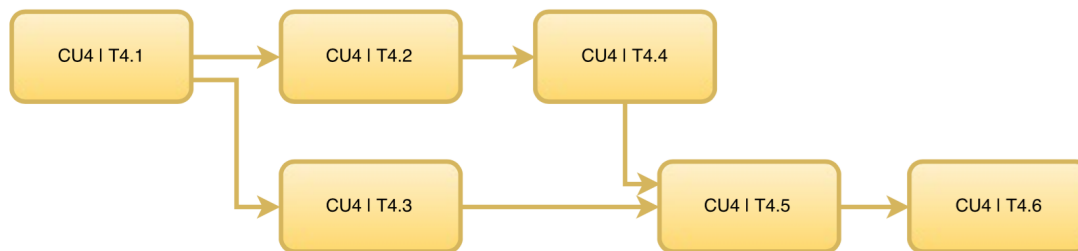


Figura 4: Diagrama de dependencias entre las tareas del CU nro 4.

■ **Tipo de iteración:** Elaboración

■ **Cantidad total de horas:** 528

■ **Tareas:**

1. Definir organización de la base de código y sistema de versionado. (4h)
2. Definir motor de base de datos a utilizar para datos de usuario. (8h)
3. Desarrollar un servidor de integración continua y testing. (20h)
4. Definir método de encriptación para contraseñas y datos sensibles. (10h)
5. Definir conjunto de datos a persistir por cada usuario. (16h)
6. Elegir lenguajes y plataformas para el desarrollo de cada subsistema. (10h)
7. Establecer criterios de calidad y formato de código. (2h)
8. Diseñar base de datos de usuarios. (10h)
9. Desarrollar un servidor de deploy. (15h)
10. Implementación de CU1 (40h)
11. Testing de CU1 (30h)
12. Investigar tecnologías web a utilizar para el cliente web. (16h)
13. Investigar tecnologías a utilizar para los clientes móviles. (16h)
14. Desarrollar stubs de las aplicaciones web y móviles. (8h)
15. Integrar motor de gráficos 2D a la aplicación web. (18h)
16. Integrar motor de gráficos 2D a la aplicación móvil. (18h)

17. Implementación de una visualización de partido de prueba. (45h)
18. Testing de CU2 (40h)
19. Investigar plataforma para el servidor de transmisión. (10h)
20. Definir la interfaz de comunicación con el servidor. (10h)
21. Desarrollar stub del servidor. (8h)
22. Implementar transmisión de datos de una simulación de prueba. (35h)
23. Desarrollar cliente simple de prueba. (10h)
24. Testing de CU3 (30h)
25. Definir motor de base de datos a utilizar para datos de desafíos. (8h)
26. Definir conjunto de datos a persistir por cada desafío. (10h)
27. Investigar alternativas para la persistencia segura de datos del desafío. (10h)
28. Diseñar base de datos de desafíos. (16h)
29. Implementación de CU4 (30h)
30. Testing de CU4 (25h)

### 3.6. Plan de Proyecto

En el siguiente diagrama detallamos el plan de desarrollo del proyecto, con una planificación estimada del desarrollo de las tareas a realizar en la primera iteración. Para la distribución en tiempo de las mismas, se asignó el tiempo de cuatro desarrolladores con una dedicación de cuatro horas diarias al proyecto.

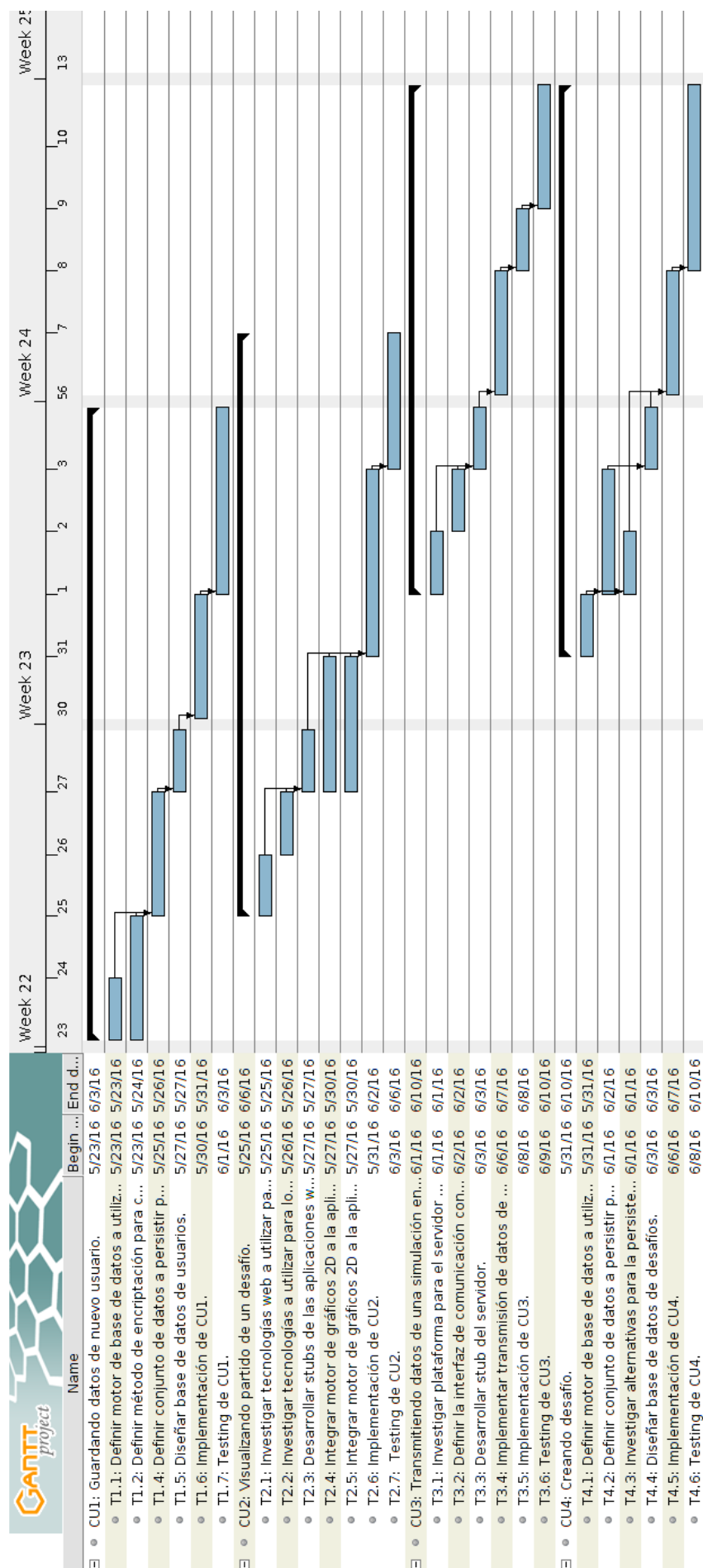


Figura 5: Diagrama de Gantt de la 1era iteración con la asignación de tiempo

## 4. Análisis de riesgos

### Riesgo 1:

- **Descripción:** Los motores gráficos 2D y 3D son desarrollados por terceros. Si alguna de las dos compañías deja de funcionar, perdemos soporte para el correspondiente motor gráfico e incrementa fuertemente el costo y tiempo de desarrollo para realizar mejoras o agregar funcionalidad en el mismo.
- **Probabilidad:** Baja
- **Impacto:** Alto
- **Exposición:** Media
- **Mitigación:** Establecer una capa de abstracción encima de los motores gráficos para que los mismos sean reemplazables sin que sea necesario modificar los demás sistemas.
- **Plan de contingencia:** Licitación el motor gráfico que ha perdido soporte entre nuevas empresas.

### Riesgo 2:

- **Descripción:** Actualmente, los usuarios tienen múltiples modos para visualizar un partido: simulación 2D o 3D a partir del feed de la simulación, streaming de partidos reales y streaming de un render de la simulación realizado del lado del servidor. Si un usuario tiene un dispositivo sin las prestaciones necesarias para realizar una simulación de forma nativa, cada vez que tenga mala conectividad tampoco será capaz de visualizar streamings y no podrá obtener información sobre los partidos de ninguna manera.
- **Probabilidad:** Media
- **Impacto:** Medio
- **Exposición:** Media
- **Mitigación:** Desarrollar un feed de información acerca del partido en ejecución que sea apto para clientes con bajos recursos de hardware y mala conectividad.
- **Plan de contingencia:** Proponer al usuario afectado la visualización del feed del partido basado en texto.

### Riesgo 3:

- **Descripción:** El sistema en su conjunto procesa varios conjuntos de datos altamente sensibles que refieren a la identidad de sus usuarios, a sus medios de pago, y la realización de las simulaciones que determinan los desafíos para bajo apuestas o con premios, así como los resultados de los mismos. Si alguno de estos sistemas es comprometido, la integridad personal o económica de los usuarios o del sistema mismo pueden sufrir consecuencias irreversibles.
- **Probabilidad:** Media
- **Impacto:** Alto
- **Exposición:** Media
- **Mitigación:** Investigar fuertemente las mejores prácticas en seguridad para el desarrollo de los sistemas que procesan información sensible.
- **Plan de contingencia:** Suspensión inmediata de actividades en la plataforma con posibles resultados irreversibles (pagos, entrega de premios) e investigación sobre el siniestro cooperando con las autoridades.

## 5. Atributos de calidad

### 5.1. Disponibilidad

<b>Descripción</b>	Los partidos deben servirse con excelente calidad y sin cortes a pesar de que servidores de la empresa proveedora se pueden caer sin previo aviso.
<b>Fuente</b>	Servidor del proveedor
<b>Estímulo</b>	Se cae, por lo tanto deja de enviar datos.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se identifica la caída del server por timeout y se inicia una nueva comunicación con otro server.
<b>Medición</b>	El tamaño del buffer de streaming es suficiente para que sólo se perciba la caída 0,001 seg/hora dada que el uptime de los servidores del proveedor es de 95 %.

<b>Descripción</b>	El sistema debe evitar que los servidores superen su límite de clientes y se caigan.
<b>Fuente</b>	Usuario
<b>Estímulo</b>	Se hace un pedido.
<b>Entorno</b>	Sistema con servidor saturado
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se selecciona un servidor que no está saturado y se direcciona al cliente a él.
<b>Medición</b>	El 0.01 % de veces un servidor se satura y se direcciona un cliente a él antes de que la notificación de que está saturado llegue al router.

<b>Descripción</b>	Desea que haya un esfuerzo por respetar a los países en los que su legislación no permite que directamente se ingrese al sitio.
<b>Fuente</b>	Externa
<b>Estímulo</b>	Un usuario proveniente de un país que en su legislación no permite ingresar al sitio intenta ingresar.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se muestra una pantalla con una leyenda indicando que está prohibido el ingreso.
<b>Medición</b>	En el 0,001 % de los casos un usuario perteneciente a un país que prohíbe el acceso al sitio logra ingresar.

<b>Descripción</b>	Se requiere que sea fácil desactivar una cuenta por un tiempo para ayudar a los adictos en recuperación.
<b>Fuente</b>	Usuario adicto
<b>Estímulo</b>	Un usuario adicto intenta conectarse.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se bloquea el acceso al sistema.
<b>Medición</b>	El 99,99 % de los casos que un usuario adicto intenta conectarse desde una cuenta desactivada el acceso al sistema es bloqueado.

## 5.2. Modificabilidad

<b>Descripción</b>	El simulador debe poder extenderse para soportar reglamentaciones nuevas ya que se espera poder incluir países adicionales.
<b>Fuente</b>	Desarrollador
<b>Estímulo</b>	Se quiere agregar reglamentaciones nuevas para incluir países adicionales.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema en ejecución
<b>Respuesta</b>	Se realiza la extensión exitosamente.
<b>Medición</b>	El costo del cambio toma un 20 % del que tomó implementarlo por primera vez.

<b>Descripción</b>	El sistema debe correr en la mayor cantidad de plataformas posible. El sistema debe poder extenderse fácilmente para incluir plataformas nuevas.
<b>Fuente</b>	Desarrollador
<b>Estímulo</b>	Se quiere agregar nuevas plataformas donde correr el sistema.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se agrega con éxito una nueva plataforma donde correr el sistema.
<b>Medición</b>	El tiempo de overhead de agregar una nueva plataforma al sistema por sobre el mínimo necesario que requiere la nueva plataforma es del 5 %.

<b>Descripción</b>	Se quiere poder controlar las publicidades en las simulaciones y el sitio en general en base a el tipo de audiencia.
<b>Fuente</b>	Administrador
<b>Estímulo</b>	Desea poder tener el control sobre las publicidades del sitio.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema en ejecución
<b>Respuesta</b>	Se cambian las publicidades que se muestran en las simulaciones.
<b>Medición</b>	Se modifican solamente 2 repositorios del sistema.

<b>Descripción</b>	La simulación debe poder mejorarse para que sea más realista de manera incremental sin que los cambios sean costosos.
<b>Fuente</b>	Desarrollador
<b>Estímulo</b>	Se quiere mejorar la simulación para que sea mas realista.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se mejora el sistema logrando que sea mas realista.
<b>Medición</b>	Se modifica exactamente un módulo.

## 5.3. Usabilidad

<b>Descripción</b>	Los datos de pago deben guardarse para que el usuario sólo tenga que actualizarlos esporádicamente.
<b>Fuente</b>	Usuario
<b>Estímulo</b>	Un usuario desea solamente actualizar los datos de pago.
<b>Entorno</b>	Normal.
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	El usuario actualiza los datos de pago.
<b>Medición</b>	El usuario realiza 4 clicks para actualizar los datos de pago.

<b>Descripción</b>	La interfaz gráfica de los usuarios debe ser similar a la de un videojuego (sobre todo al momento de ver a los jugadores, las jugadas de los técnicos, colocar el nombre y logo del equipo del participante, con animaciones y efectos especiales con aceleración gráfica).
<b>Fuente</b>	Externa
<b>Estímulo</b>	El usuario utiliza la interfaz de simulación.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se muestra la simulación como si fuera un videojuego
<b>Medición</b>	El usuario puede ver la simulación como si fuera un videojuego haciendo menos de 3 clicks.

#### 5.4. Performance

<b>Descripción</b>	La calidad del streaming debe adaptarse al bandwidth disponible para que la reproducción del video sea fluida.
<b>Fuente</b>	Interna
<b>Estímulo</b>	Se desea reproducir un streaming de video.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Reproductor de streaming
<b>Respuesta</b>	Se reproduce el straming en el motor 2D o 3D segun se soporte.
<b>Medición</b>	En caso de que el bandwidth sea menor que 128Mb/s se usa el motor 2D, si no el 3D.

<b>Descripción</b>	Se debe utilizar el engine 3D siempre que sea posible. Si el cliente no lo soporta adecuadamente debe utilizarse el 2D.
<b>Fuente</b>	Extenera
<b>Estímulo</b>	Se desea transmitir streaming de video.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Cliente
<b>Respuesta</b>	Dependiendo de las funcionalidades del cliente se muestra el engine 3D o 2D segun corresponda.
<b>Medición</b>	En el 78 % de los dispositivos clientes se utilizan los motores 3D.

<b>Descripción</b>	Los desafíos globales que requiere el streaming a múltiples regiones simultáneamente debe funcionar satisfactoriamente. La experiencia del usuario no debe verse afectada.
<b>Fuente</b>	Externa
<b>Estímulo</b>	Un usuario visualiza un streaming con múltiples regiones.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se procesa el streaming sin demoras ni buffering.
<b>Medición</b>	En el 0,001 % de los casos el usuario experimenta demoras.

## 5.5. Seguridad

<b>Descripción</b>	Los pagos y las credenciales de pago de los usuarios deben manejarse con seguridad, debe impedirse que hackers redireccionen los pagos o interfieran con las transacciones.
<b>Fuente</b>	Individuo externo
<b>Estímulo</b>	Un individuo intenta redireccionar un pago.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se denega el acceso a este usuario y se prohíbe que ingrese.
<b>Medición</b>	En el 0.00001 % de los casos el individuo externo logra redireccionar el pago.

<b>Descripción</b>	Los datos de usuarios deben estar protegidos contra robos. Tiene que asegurarse la confidencialidad e integridad de los mismos.
<b>Fuente</b>	Individuo externo
<b>Estímulo</b>	Un individuo externo intenta acceder a los datos de otros usuarios.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se denega el acceso al individuo externo.
<b>Medición</b>	En el 0.00001 % de los casos el individuo externo logra acceder a los datos de otro usuario.


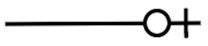
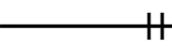


<b>Descripción</b>	Todas las transacciones de dinero deben estar logueadas de manera segura para poder presentarlas como evidencia a las autoridades de cada región.
<b>Fuente</b>	Interna
<b>Estímulo</b>	Se realiza una transacción de dinero.
<b>Entorno</b>	Normal
<b>Artefacto</b>	Sistema
<b>Respuesta</b>	Se guarda en un log los datos de la transacción de dinero.
<b>Medición</b>	El 99.9 % de las transacciones de dinero son guardadas correctamente en el log.



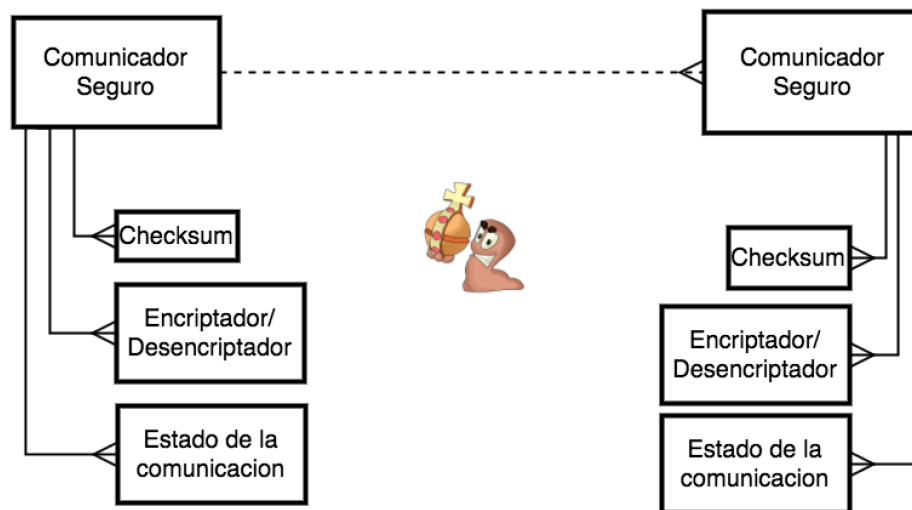
## 6. Arquitectura

### 6.1. Conectores propios

#### 6.1.1. Notación

- Pipe 
- Holy Connector 
- Asynchronous Holy Connector 
- Holy Video Connector 
- Holy Data Connector 

#### 6.1.2. Holy Connector

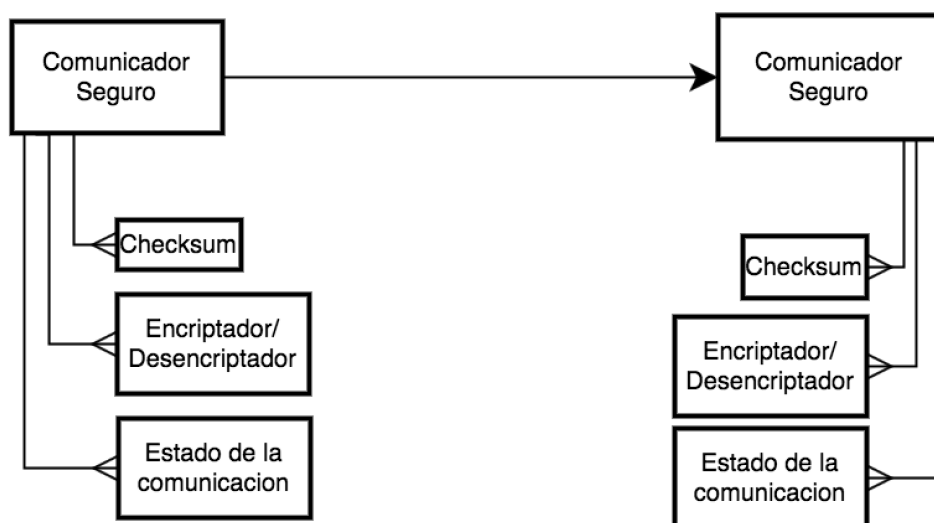


Este conector provee:

- **Confidencialidad** Mediante un componente de encriptación de un extremo, y uno de desencriptación del otro, de los datos enviados.
- **Integridad** Mediante un componente de checksum.
- **Confiabilidad de la conexión.** Mediante un componente que mantiene el estado de la conexión. Por ejemplo, este componente podría implementar el protocolo TCP.

Observemos que entre los extremos utilizamos un conector tipo client-server que, asumimos, que se extiende sobre un medio inseguro. Por lo tanto, podemos pensar al Holy Connector como un cliente-server seguro, sobre un medio inseguro.

### 6.1.3. Asynchronous Holy Connector



Idéntico al Holy Connector, con la diferencia de que en lugar de usar un client-server como conector intermedio, utiliza un conector de call asincrónico.

### 6.1.4. Holy Video Connector



Cuenta en ambos extremos con comunicadores de streaming, éstos se comunican mediante un Holy Connector. Además cuenta en ambos extremos con compresores de video para aumentar el throughput de la transmisión. Un detalle importante es que haremos un pequeño abuso de la funcionalidad del Asynchronous Holy Connector, asumiendo que al utilizarlo como conector aquí, en el Holy Video Connector, no se utiliza el componente de checksum. Hacemos esto para evitar definir un componente idéntico al Asynchronous Holy Connector pero sin el componente de checksum. La razón por la que no queremos usar éste componente, es que no nos interesan chequeos de integridad, a lo largo de la comunicación del flujo de datos. Por un lado, este chequeo demanda tiempo, que es escaso en el contexto de streaming. Por otro lado, no es necesario garantizar completa integridad, y se admiten pequeñas diferencias entre los datos enviados y los recibidos.

Los conectores de entrada y salida a los respectivos Comunicadores de Streaming de video son pipes. El buffering que éstos proveen permite que la experiencia del usuario no se vea afectada por las posibles fluctuaciones del caudal de la transmisión (evita que el usuario vea interrumpido su video porque el mismo está “Cargando...”).

### 6.1.5. Holy Data Connector



Análogo al HVC, pero con compresores de texto en lugar de compresores de video. En este caso sí se utiliza el componente de checksum en el conector que vincula los extremos, ya que la integridad de los datos es importante para generar las simulaciones.

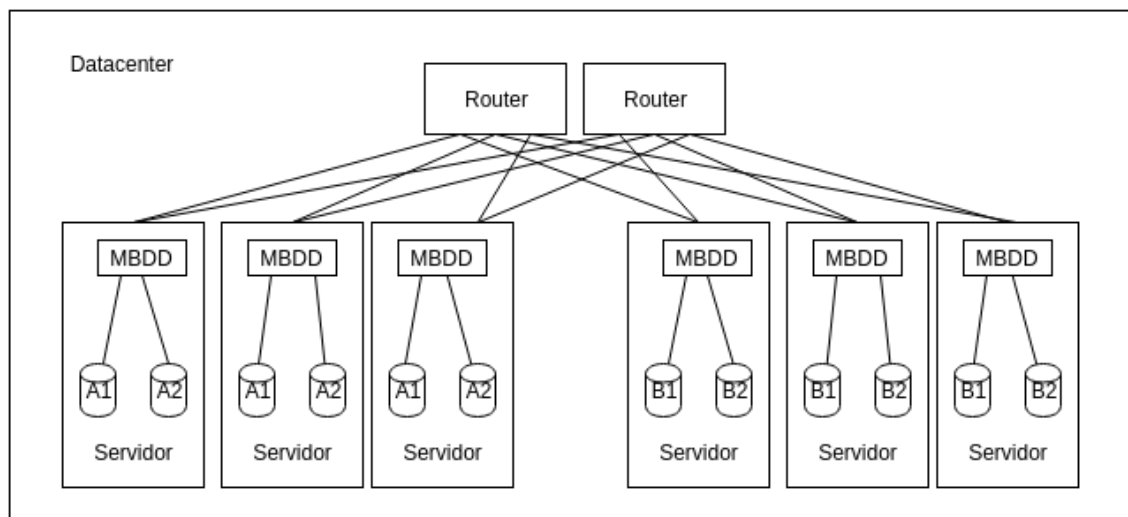
## 6.2. Diagrama

### 6.2.1. Arquitectura orientada a datacenters

Nuestro sistema tiene una arquitectura distribuida, basada en datacenters. Un datacenter contiene una gran cantidad de servidores que proveen el servicio de juego. Dentro de un datacenter, los servidores se dividen en conjuntos, y cada conjunto provee servicio a una región distinta. Una región es uno o más países. En el caso extremo, una región son todos los países del mundo. Notar que esto significa que un datacenter puede proveer servicio a varias regiones.

Todas las peticiones que ingresan en un datacenter son recibidas y procesadas por una máquina de tipo *router*. Un router redirige la petición a algún servidor de la región a la que esté dirigida la petición. Los routers pueden rutear al conjunto de servidores de cualquiera de las regiones servidas en el datacenter. Por ejemplo, podría haber un datacenter en Argentina, que provea el servicio para Argentina, Brasil y Uruguay; en este caso, los routers del datacenter procesan peticiones de las tres regiones.

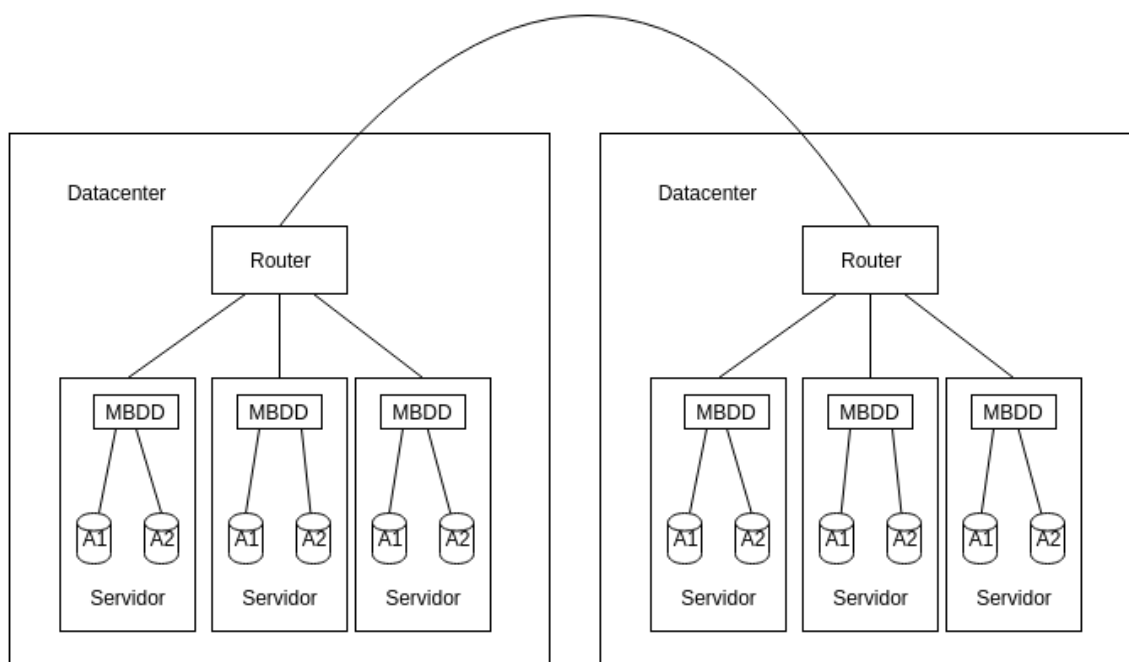
El siguiente diagrama muestra lo descripto.



MBDD = Motor de bases de datos distribuidas

El datacenter de la figura tiene dos routers que procesan pedidos para dos regiones. Cada región se compone de tres servidores. Notar que una de las regiones contiene dos tipos de repositorios, A1 y A2. Estos repositorios aparecen en todos los servidores de la región, debido a que los datos están distribuidos sobre todos estos servidores. Análogamente, los servidores de la otra región, tienen repositorios B1 y B2, distribuidos. Cada servidor tiene un *motor de bases de datos distribuidas*, que administra el acceso a los repositorios distribuidos.

Este esquema de distribución de datos sobre servidores es lo que permite que el sistema sea escalable. Además provee otras bondades como capacidad de replicación de datos, necesario para tener alta disponibilidad. Más aun, es posible implementar distribución de datos entre datacenters, si hacemos que el conjunto de servidores de una región esté distribuido en más de un datacenter. Esto se muestra en la siguiente figura.



MBDD = Motor de bases de datos distribuidas

De esta forma podemos replicar datos entre distintos datacenters, lo cual permite a la arquitectura soportar la caída de uno de ellos sin sufrir pérdidas de datos ni dejar de proveer servicio a una región.

### 6.2.2. Descripción general

Para describir la arquitectura, optamos por explicar cómo se satisface cada una de los requerimientos indicados en el enunciado. Para ésto, decidimos citar textualmente, cada párrafo del enunciado, y describir los aspectos del sistema relacionados. Si bien la descripción no abarca todos los aspectos del sistema (una explicación exhaustiva demandaría demasiado tiempo y espacio), es un buen punto de partida para entender el funcionamiento general del mismo.

*Lo primero que se pretende es que abarque varios de los deportes colectivos más populares del planeta. Además del ya citado básquet, deberá incorporar a las ligas de fútbol, hockey, rugby, béisbol y fútbol americano que se determine tengan mayor número de seguidores (ej. el fútbol español o la NFL estadounidense), y que se consigan empresas que provean datos en tiempo real de la evolución del partido (tanto del desempeño de los jugadores como del partido en general). Esto último es importante porque además de la simulación de los mismos, ahora también se quiere incorporar un modo de liga de fantasía tradicional.*

Para agregar deportes, lo único que se debe hacer es contratar nuevos servicios de transmisión de partidos, agregar nuevos parámetros de juego (i. e. agregar datos al repositorio de *puntajes por acción*), y agregar funcionalidad al *administrador de desafíos* para que se puedan crear desafíos del nuevo deporte.

Los datos de la evolución del partido provienen de los mismos servicios que nos transmiten partidos reales. Tanto los datos de un partido, como la transmisión del mismo nos son transmitidos en conjunto. Ver componente *Servicio de transmisión de partido real*.

***En este nuevo modo los ganadores de los desafíos ya no se deciden a través de la simulación directa de los partidos, sino del desempeño de los jugadores en partidos reales en las ligas en cuestión. Por ej. usando el caso del básquet, se podría otorgar 1 punto por cada punto que haga ese jugador en el partido, 1.5 puntos por cada asistencia, 2 por cada bloqueo o robo, -1 por cada pérdida de balón, etc... El equipo que tenga mayor suma, gana el desafío.***

El puntaje que se asigna a cada acción de cada deporte es un parámetro que se encuentra en el repositorio *puntajes por acción*.

***Con respecto a los desafíos, ahora pueden incluir un gran número de partidos, no sólo uno. En el caso de este nuevo modo “liga fantasía tradicional”, podrían incluirse todos los partidos de una fecha determinada de una liga, o un salteado de un conjunto de fechas cualesquiera, o incluso la liga o torneo completo. En el caso de la simulación, se arman torneos de diferentes formatos (se elige al momento de plantearse el desafío, playoff, liga, combinado zonas, etc.) con todos los equipos participantes. En este último caso se agrega la duración total “artificial” del torneo en semanas, pues se pretende que los partidos no se resuelvan más de forma inmediata si no que puedan “vivirse” como los reales (más información luego).***

Los desafíos son creados por un cliente (un jugador o un administrador) desde la pantalla de lista de desafíos, mediante el módulo *creador de desafíos*. Éste se comunica con un *administrador de desafíos*, del lado del servidor, el cual, mediante el *administrador de fixtures*, crea el fixture correspondiente al desafío. El fixture puede tener diversas formas, dependiendo del tipo de desafío creado, por ejemplo una liga, un torneo o un partido individual. El fixture es escrito en el repositorio *fixtures de desafíos*.

Para que los desafíos se puedan extender a lo largo del tiempo, y se puedan vivir como si fuesen reales, existe la posibilidad de que los partidos de un desafío comiencen en distintos momentos. Esta información se incluye en el fixture.

Como los partidos comienzan en distintos momentos, el sistema contiene un módulo *dispatcher de partidos simulación* que sensa regularmente los fixtures, buscando los próximos partidos a ejecutarse. Cuando llega la hora de ejecutar un partido, le informa al *simulador de partido* que debe comenzar a ejecutar una simulación. Le transmite la información de los equipos y los datos de los usuarios involucrados. Cuando el simulador concluye la ejecución, le informa el resultado al *procesador de resultados*.

Por un lado, este módulo le comunica al *administrador de pagos* el ganador y el perdedor del partido, para que éste efectúe el pago correspondiente, según las apuestas. Las apuestas se encuentran en el repositorio *apuestas por partido en ejecución*. Con toda esta información, el administrador de pagos le indica al sistema de pagos externo, las transacciones a realizarse.

Por otro lado, al terminar la simulación, se le informa al administrador de fixtures cómo debe modificar el fixture. Otros repositorios también son modificados adecuadamente.

***No sólo los jugadores pueden crear desafíos, sino que administradores propios del sitio pueden crearlos. Queda claro, que ahora los desafíos pueden ser aceptados por miles de jugadores, que competirán cada uno con su equipo para salir vencedor.***

Los desafíos pueden ser creados desde un programa cliente hecho específicamente para administradores.

***Cada desafío tendrá un chat general para intercambiar mensajes con otros participantes, además de las conversaciones que podrá tener con competidores amigos a modo de “Instant Messenger” a través de la plataforma.***

Se incluye la posibilidad de chatear en las pantallas de partido. Un *receptor/enviador de mensajes de chat* se encarga de recibir mensajes de chat, distribuirlos a todos los usuarios conectados a una sala, y filtrar y analizar aquellos que se refieran a jugadores de alguno de los equipos del partido, para modificar sus estadísticas en tiempo real. Las modificaciones de las estadísticas de los deportistas son almacenadas en el repositorio *estadísticas de deportistas*.

***La idea principal con mayor consenso en los inversores, es que las fichas de apuesta***

*pasan a ser reemplazadas por dinero real. Todos los participantes deberán poder ingresar datos de una tarjeta de crédito o cuenta corriente en entidades bancarias de los países participantes para que el sistema pueda debitar o acreditar dinero de forma inmediata y ser registrados por el sistema de forma segura.*

Justo antes de comenzar un partido, ocurre una ronda de apuestas entre los jugadores involucrados. Las apuestas, solicitadas por un *administrador de apuestas*, se crean en el cliente por medio de un *creador de apuestas*. Recibidas las apuestas en el servidor, el administrador las almacena en el repositorio *apuestas por partido en ejecución*.

*Si bien el juego sigue permitiendo jugar absolutamente gratis, existirán desafíos que tengan una cuota de entrada para participar y repartirán un monto de dinero en premios en base a una cantidad mínima de participantes que deben anotarse para que el mismo pueda realizarse. Los premios que se pagarán serán en relación al rango de posiciones en el que queda el equipo del jugador al terminar el desafío (ej: 1er puesto, 2do, 3ro, 4to a 10mo, 10 a 50, 50 a 200, 200 a 1000, etc.). Los desafíos “gratis” pueden otorgar premios como créditos no retirables para jugar, productos exclusivos de “partners”, tickets para eventos deportivos, etc.*

La cuota de entrada y los premios no monetarios se fijan a la hora de crear un desafío, y son procesados por el administrador de desafíos. Esta información es almacenada en el repositorio *premios por desafío*.

*Una vez que comienza el primer partido de un desafío ya no se pueden hacer cambios en el equipo registrado o salirse del mismo o ingresar nuevos participantes. Cada desafío deberá tener una cuenta regresiva para indicar el tiempo restante para el comienzo. En todo momento, incluso mientras los partidos se están desarrollando, se debe poder ver quiénes son los participantes del mismo y el puesto actual del equipo del participante en el desafío actualizado al minuto a minuto de los partidos.*

Para inscribirse a un desafío, un usuario utiliza el *listador de desafíos* que, además de mostrar todos los desafíos disponibles, permite inscribirse a ellos. La información de la inscripción es enviada a un administrador de desafíos, que le ordena al administrador de fixtures modificar el fixture con el nuevo inscripto. Comenzado un desafío, el administrador de fixtures no aceptará ningún cambio sobre el fixture. Por lo tanto, no habrá nuevas inscripciones.

Asimismo es posible recuperar la información sobre los participantes de un desafío en curso, a través del fixture del desafío. Los rankings de un desafío son confeccionados por un *calculador de rankings de desafíos* que, basándose en un fixture, calculan la posición de cada jugador.

*De hecho, el estado de los partidos de los desafíos (en cualquiera de sus modos) debería poder seguirse en vivo a través del sitio, como cualquier sitio de noticias deportivo. Además, se está muy avanzado en conversaciones para acceder a streams de videos de diferentes ligas y que el partido pueda seguirse legalmente a través de la plataforma en algunas regiones habilitadas. Los dueños de los derechos de transmisión, a cambio, quieren poder vender espacios de publicidad en el sitio y acceder a algunos datos de preferencias/comportamiento de los usuarios, además de sus cuentas de mails para efectuar campañas de marketing más específicas.*

Cuando un usuario ingresa a la sala de un desafío, se conecta a un servidor que esté simulando o proveyendo una transmisión de partido real, y éste comienza a transmitirle datos.

En el caso del modo fantasía, el permiso de acceso a una transmisión está almacenado en el repositorio *regiones habilitadas para transmisión de partidos fantasía*. Cuando un usuario solicita acceso a la sala de un desafío, primero debe resolverse la ubicación de un datacenter que contenga la información sobre ese desafío. Ubicado un tal datacenter, se verifica el permiso de acceso a la transmisión, y en caso que la región del usuario esté habilitada, se le concede acceso a un servidor que esté proveyendo la transmisión.

A los dueños de los derechos de transmisión se les puede dar acceso a los repositorios de datos de usuarios fácilmente, consultando cualquiera de los datacenters. A su vez, tenemos almacenadas publicidades que serán mostradas en pantallas de desafíos. Las publicidades son recibidas a través de proveedores de publicidad externos, y almacenadas en un repositorio *publicidades*.

*Con respecto al modo de simulación, se pretende extender las reglas utilizando a un comité de expertos (jugadores, técnicos, estadísticos, periodistas deportivos, etc.) de cada*

*deporte para intentar que se asemeje aún más a la realidad. En el caso del básquet incorporará fouls, tiros libres, cambios de jugadores (suplentes), minutos en cancha + cansancio, tiempos con cambios de lado de la cancha, estadios locales y visitantes, condiciones climáticas, movimiento de los jugadores y sus posiciones en la cancha al efectuar acciones, etc. para extrapolar más certeramente la actuación del jugador en base a sus estadísticas. Al mismo tiempo se busca modificar el simulador para que en vez de un log de salida, provea un stream continuo de “minuto a minuto” mucho más detallado que el log de la versión anterior. La razón de esta decisión es que se está por llegar a un acuerdo por la compra de un motor gráfico que permita a los participantes ver el partido a partir del stream de salida del simulador. En este momento se negocia por al menos dos motores diferentes.*

Las reglas de los juegos simulados están completamente contenidas en el simulador. Su modificación no afecta al funcionamiento del resto de nuestro sistema.

Un simulador de partido, ubicado en el servidor, envía un flujo de datos a todos los clientes conectados al mismo. Cuando un cliente recibe datos de la simulación, son procesados por un motor gráfico, el cual posteriormente los dibuja en la interfaz gráfica.

*El primero es un motor gráfico 3d de última generación que muestra a los jugadores con un gran nivel de realismo pero que puede ser muy pesado para la mayoría de los dispositivos con los que los participantes accederían al portal (que va desde PCs de escritorio a dispositivos móviles de todo tipo). La empresa desarrolladora habla de que se le podría diseñar un módulo para transmitir en video el partido a los dispositivos no soportados por el motor gráfico. Otra empresa presenta un motor 2d más humilde pero mucho menos demandante, donde los partidos se ven “desde arriba” con un diseño caricaturesco y que funciona para dispositivos móviles con varios años de antigüedad. Se cree que se va a llegar a un acuerdo con ambas compañías para incorporar ambos motores en la primera versión, según se necesite.*

Se decidió incluir ambos motores gráficos en el programa cliente. Por lo tanto, cuando se detecte que el dispositivo cliente no tiene prestaciones suficientemente buenas para utilizar el motor 3D, se utilizará el motor 2D. No se enviarán datos de video 3D preprocesados desde el servidor.

*Cualquier participante, incluso si no forma parte del desafío debería poder ver un partido simulado de su región. Al ser ahora “visibles” las simulaciones, las mismas dejan de ser instantáneas, y se intenta que las duraciones sean similares a las de un partido normal. De esta forma se puede vender publicidad específica a través del sitio. También se quiere aumentar el caudal de redes sociales que se utilizan para “afectar” las estadísticas (no sólo Twitter, sino incorporar Facebook, Google+, etc.) y que incluso los comentarios de los usuarios en los chats de los desafíos o incluso los IMs (Instant Messages) privados de los usuarios puedan ser utilizados para afectar los resultados. Como las simulaciones ahora duran similarmente a un partido real, los resultados de menciones pueden verse como un “rating minuto a minuto” de cada jugador.*

Un usuario puede solicitar ver cualquier partido de la lista de desafíos y, si su región está habilitada, un servidor adecuado se lo transmitirá. La duración del partido estará dado por el flujo de datos transmitido del servidor al cliente.

Las redes sociales son constantemente sensadas por un *administrador de redes sociales*, que se encarga de extraer datos relevantes relacionados con deportes. Estos datos son almacenados en un repositorio *información de redes sociales*. Esta información es sensada periódicamente por un *analizador de información de redes sociales*, el cual procesa la información y se la transmite al simulador de partido. Además, actualiza el repositorio de estadísticas de jugadores.

*Se espera que el sistema sea utilizado por millones de personas a lo largo y ancho del planeta y que se llegue a ese número de suscriptos muy rápidamente (hay mucho interés de que uno de los países incluidos en la versión inicial sea China). Algunos especialistas en redes plantearon entonces la necesidad de regionalizar la plataforma en varios niveles y que los participantes de cada región sólo compitan entre sí en desafíos locales o nacionales (que puede ser de las ligas del mundo más seguidas para esa región). Además, se cree, esto debería facilitar la logística con los bancos, el idioma de los chats, los pagos en la moneda local, la entrega de premios, etc.*

Como vimos, la arquitectura del sistema está orientada a datacenters. Si se quisiera habilitar el servicio en China, una buena idea sería construir un datacenter en el continente asiático. Dicho datacenter contendría servidores que sirven, en particular, al antedicho país.

Ya hemos hablado de la organización de servidores por regiones. Para tener ligas locales, continentales y mundiales, podemos crear regiones que abarquen los países que deseamos que conformen la liga. Para nuestro sistema, cualquier conjunto de países es idéntico, con lo cual podemos servir un campeonato mundial de la misma forma que servimos un torneo local. El único detalle es que en el caso de una liga local, lo más razonable es ubicarla en un datacenter cercano al país en el que se desarrolla la liga, mientras que en una liga mundial, la mejor elección del (o los) datacenter que proveerá el servicio no es tan obvia. Puede que cualquier datacenter sea una buena elección.

*El ranking de los jugadores más ganadores del prototipo se mantiene pero con algunos cambios, ya que desafíos con muchos usuarios o muy largos en tiempo otorgarán mayor puntaje que los “mano a mano” convencionales. Dicho ranking servirá para acceder a torneos de niveles regionales o internacionales. Por ejemplo, en base a las limitaciones del hardware y conexiones con el que se cuenta para servidores, los mejores 10.000 participantes de Argentina podrán acceder a una nueva lista de desafíos con los restantes 10.000 mejores jugadores de cada país del resto de Latinoamérica. Y los mejores 1.000 de esta nueva región podrán competir en un nivel mundial con los 1.000 mejores de Europa, Asia, Norteamérica, África y Oceanía. Para un participante mantenerse en los rankings de cada nivel, los resultados de la región mundial, no influirán en los del nivel regional (por ej. Europa), ni en los de nivel nacional o local, aunque dejar de competir por un tiempo en alguno hará restar posiciones, lo que eventualmente puede significar perder acceso a los niveles superiores. Se quiere tener campeones mundiales anuales, y que las finales sean verdaderos eventos a transmitirse en vivo a los millones de usuarios del sitio de todas las regiones para luego hacer promociones y giras con ellos alrededor del mundo. Esto implica que los administradores pueden hacer visibles desafíos de niveles superiores a usuarios que no tienen normalmente acceso para que el resto de los usuarios puedan seguirlos en vivo. Se plantea que cada vez que comience una nueva temporada o año, se reinicie el ranking del sitio.*

Cuando un administrador crea un desafío internacional, se computa el ranking de todos los jugadores de las regiones intervinientes en el desafío. Ésto lo hace un *calculador de rankings de usuarios*. El resultado se almacena en el repositorio *rankings de usuarios*.

Con esta información, cada vez que un usuario desea inscribirse a dicho desafío, el administrador de desafíos consulta el ranking, y verifica si el usuario se encuentra entre los primeros  $n$  del ranking.

Observar que los resultados de la región mundial no afectan a las estadísticas de otras regiones, gracias a que los servidores de partido sólo modifican las estadísticas de usuario de su conjunto. En otras palabras, los servidores de región mundial sólo modifican la base de datos distribuida de región mundial.

*Es importante que no sólo los participantes puedan acceder a su estado de cuenta, sino que administradores del más alto nivel de privilegios también puedan hacerlo para cualquier jugador. Además, deberían ser capaces de ver los movimientos y el balance del sitio de juegos en cada región o a nivel global. Como se quiere evitar potenciales problemas con entidades gubernamentales de control (fundamentalmente del fisco) en los países que opere el sitio, se desea que estén diseñados e implementen mecanismos de tal manera que se pueda brindar la transparencia deseada, tanto en movimientos de dinero, como en el funcionamiento correcto de las simulaciones.*

Todos los programas cliente de tipo administrador cuentan con un *administrador de cuentas de usuario*, que permite visualizar toda la información disponible sobre los usuarios.

Se garantiza la seguridad de las transacciones utilizando conectores Holy, que garantizan seguridad. Todas las transacciones pueden ser listadas tanto por usuarios como administradores, a través del *listador de transacciones*, el cual consulta el repositorio de *transacciones*.

*Por último, se sabe que la legislación de varios países prohíbe el acceso a este tipo de juegos. Dependiendo del caso, se deberá evitar que: directamente el usuario acceda al sitio o que no puedan registrarse o bien registrarse, pero sólo jugar los desafíos gratuitos sin poder ganar premios o pudiendo ganar premios que no sean sumas de dinero.*



Cuando un usuario solicita registrarse en el sistema, un componente *registrador* verifica que el usuario pertenezca a una región habilitada para jugar. Dicha información se encuentra en el repositorio *regiones habilitadas para jugar/apostar*. Análogamente, cuando un usuario solicita inscribirse a un desafío, el administrador de desafíos chequea contra mismo el repositorio, que el usuario efectivamente tenga permiso para apostar.

### 6.2.3. Repositorios distribuidos

A continuación listamos la información contenida en cada uno de los repositorios. Recordemos que todos ellos forman parte de un esquema de almacenamiento de datos distribuido. Conocer exactamente qué información tiene a disposición el sistema, es útil para comprender sus capacidades.

- **Ranking de usuarios.** Contiene el ranking de los usuarios de la región.
- **Apuestas por partido en ejecución.** Contiene las apuestas involucradas en cada partido en ejecución.
- **Servidores simulando o transmitiendo partido.** Contiene los servidores que están simulando o transmitiendo cada partido.
- **Partidos pendientes y en juego.** Contiene los partidos (tanto de desafío simulación como fantasía) que están pendientes y en ejecución.
- **Regiones habilitadas para jugar/apostar.** Contiene las regiones a las que se les puede prestar servicio.
- **Estadísticas de usuario.** Contiene información estadística, como por ejemplo la cantidad de partidos ganados y perdidos, de cada jugador.
- **Información de eventos reales.** Contiene toda la información sobre eventos reales en los que se basan partidos fantasía.
- **Usuarios conectados a partido.** Contiene ID e IP de los usuarios actualmente conectados a la sala de un desafío.
- **Regiones habilitadas para transmisión de partidos fantasía.** Contiene las regiones a las que se les puede transmitir un partido real desde ésta región.
- **Datos de cuenta de usuario.** Contiene los datos de un jugador de la región.
- **Estado de partido en ejecución.** Contiene los datos de estado, por ejemplo puntaje de cada equipo, de los partidos que se están ejecutando en este momento.
- **Fixtures de desafíos.** Contiene los fixtures actualizados de los desafíos que aún no acabaron. En particular, contiene los fixtures de los desafíos que aún no han empezado.
- **Puntaje por acciones.** Contiene información sobre los parámetros que se utilizan para puntuar las acciones que ocurren en un partido fantasía.
- **Métodos de pago.** Contiene la información sobre los servicios de pago que se aceptan en ésta región.
- **Información de redes sociales.** Contiene la información más reciente descargada de redes sociales.
- **Publicidades.** Contiene las publicidades más recientes recibidas desde los proveedores de publicidad.
- **Transacciones.** Contiene registro de todas las transacciones (debitos y acreditaciones) realizadas por el sistema.
- **Premios por desafío.** Contiene los premios (monetarios y no monetarios) que se otorgan a los participantes desafíos que aún no han acabado.

- **Estadísticas de deportista.** Contiene la información estadística de los deportistas que los simuladores usan durante su ejecución.

### 6.3. Atributos de Calidad vs Arquitectura

A continuación listamos todos los atributos de calidad detallados en la sección anterior y explicamos como hicimos para resolverlo en el diagrama de componentes y conectores.

- **Atributo:** El sistema debe evitar que los servidores superen su límite de clientes y se caigan.  
**Justificación:** Los componentes *router* que reciben las peticiones de un cliente mantienen contadores acerca de la carga de los servidores. Cuando detectan una posible sobrecarga consultan la cantidad de usuarios conectados a los servidores de una región. Lo hacen consultando el repositorio *usuarios conectados a desafío*. Si hay sobrecarga, no se responde a las peticiones.
- **Atributo:** Desea que haya un esfuerzo por respetar a los países en los que su legislación no permite que directamente se ingrese al sitio.  
**Justificación:** Los *router* que reciban peticiones provenientes de regiones no habilitadas, no serán resueltas. Las regiones no habilitadas se consultan en el repositorio *regiones habilitadas para jugar*.
- **Atributo:** Se requiere que sea fácil desactivar una cuenta por un tiempo, para ayudar a los adictos en recuperación.  
**Justificación:** Se incluye un módulo para desactivar una cuenta, en el programa cliente.
- **Atributo:** El simulador debe poder extenderse para soportar reglamentaciones nuevas ya que se espera poder incluir países adicionales.  
**Justificación:** La modificabilidad el simulador corre por cuenta de los desarrolladores del mismo. En el caso del modo fantasía, cuya operatoria depende de nuestros desarrolladores, se incluye un repositorio con datos de configuración para los partidos de dicho modo.
- **Atributo:** El sistema debe correr en la mayor cantidad de plataformas posible. El sistema debe poder extenderse fácilmente para incluir nuevas plataformas.  
**Justificación:** La arquitectura del sistema no asume requerimientos de hardware o software (salvo algunos muy básicos) sobre el cliente. Para poder soportar todo el abanico de dispositivos, desde aquellos con recursos de hardware y software limitados como aquellos de altas prestaciones, se incluyen dos motores gráficos, uno 3D (de altos requerimientos) y otro 2D (de bajos requerimientos). La decisión sobre cuál utilizar se toma en base a los recursos disponibles al momento de la recepción de los resultados de la simulación.
- **Atributo:** Se quiere poder controlar las publicidades en las simulaciones y el sitio en general en base a el tipo de audiencia.  
**Justificación:** El Motor de bases de datos distribuida tiene la capacidad de hacer un analisis de datos para poder saber, dado un usuario que publicidad mostrarle.
- **Atributo:** La simulación debe poder mejorarse para que sea más realista de manera incremental sin que los cambios sean costosos.  
**Justificación:** Las mejoras de la simulacion asumimos que se hacen dentro del componente "Simulador de partido".
- **Atributo:** Los datos de pago deben guardarse para que el usuario sólo tenga que actualizarlos esporádicamente.  
**Justificación:** Contamos con un repo "Metodos de pago" donde guardamos los metodos de pago de cada usuario. Un usuario puede mediante el "Persistidor Metodo de Pago" guardarlos en ese repo.
- **Atributo:** Se debe utilizar el engine 3D siempre que sea posible. Si el cliente no lo soporta adecuadamente debe utilizarse el 2D.  
**Justificación:** Del lado del cliente contamos con un multiplexor, el cual, chequeando los requerimientos y funcionalidades del sistema cliente decidirá que engine utilizar

- **Atributo:** Los desafíos globales que requiere el streaming a múltiples regiones simultáneamente debe funcionar satisfactoriamente. La experiencia del usuario no debe verse afectada.

**Justificación:** Contamos con un conector de streaming que contiene un buffering para poder transmitir con mayor velocidad. Y por otro lado contamos con muchos servidores para que transmitan al mismo tiempo.

- **Atributo:** Los pagos y las credenciales de pago de los usuarios deben manejarse con seguridad, debe impedirse que hackers redireccionen los pagos o interfieran con las transacciones.

**Justificación:** Para esto utilizamos el conector Holy, el cual es un conector seguro explicado en la sección anterior, entonces la comunicación con el Servidor de Pago es segura.

- **Atributo:** Los datos de usuarios deben estar protegidos contra robos. Tiene que asegurarse la confidencialidad e integridad de los mismos.

**Justificación:** Para esto utilizamos el conector Holy, el cual es un conector seguro explicado en la sección anterior, entonces la comunicación con el Servidor de Pago es segura.

- **Atributo:** Todas las transacciones de dinero deben estar logueadas de manera segura para poder presentarlas como evidencia a las autoridades de cada región.

**Justificación:** Contamos con un repo de transacciones donde se loguean las transacciones. A la hora de realizar un pago el “Administrador de pagos” se comunica con el “Motor de bases de datos distribuidas” para que actualice el repo.

## 7. Conclusión

En retrospectiva, el diseño de la arquitectura resultó mucho más desafiante que la planificación del proyecto y la identificación de los casos de uso relevantes. El desarrollo de las primeras instancias es relativamente estructurado, mientras que construir una arquitectura para el sistema completo brinda una flexibilidad mucho mayor, y es por lo tanto más compleja y se presta a la creatividad del diseñador y la evaluación de múltiples trade-offs. En algunos casos, esto conllevó debates extensos entre los integrantes del grupo con la finalidad de decidir entre sendas alternativas sin parámetros claros para compararlas en términos de calidad.

Distintos principios vistos en clase se manifestaron a lo largo de este trabajo, permitiéndonos apreciar en forma directa los siguientes puntos:

- El proceso de decisión. La principales decisiones arquitectónicas deben ser tomadas por un pequeño grupo de arquitectos en forma centralizada para garantizar la integridad conceptual de la arquitectura desarrollada. Nuestro equipo constó de cuatro personas elaborando una estrategia en conjunto y discutiendo en múltiples ocasiones para lograr alinear las visiones de cada integrante; de haber dividido el diseño y desarrollado cada parte independientemente, la calidad del sistema se hubiera visto perjudicada.
- Identificación de atributos de calidad. Determinar el orden relativo de relevancia entre los atributos de calidad resultó ser una tarea compleja, remarcando el valor en el uso de herramientas como el QAW. Sin este espacio para encontrarnos con las necesidades de los stakeholders, hubiera sido muy difícil estimar cuáles atributos de calidad eran prioritarios y, en consecuencia, se habría dificultado el criterio para discriminar entre las múltiples variantes presentadas en la etapa de diseño.
- Enmarcar el proceso como un aprendizaje incremental. La arquitectura final resultó ser el producto de varias iteraciones que refinaron y resignificaron a las anteriores. Por ejemplo, la introducción de un nuevo componente nos instaba a revisar el nivel de detalle con el que habíamos definido otro.

Otro desafío incidental que se presentó fue a la hora de transmitir la arquitectura desarrollada; en particular, la determinación del mejor nivel de detalle para cada una de las partes que componen la arquitectura. Se intenta simultáneamente ser completo y preciso en la descripción, pero de gran importancia es no perder claridad en la transmisión de la información que se quiere compartir mediante el diagrama. En separadas ocasiones, se diagramaron partes con excesivo nivel de detalle, que luego consideramos no era relevante dentro de la arquitectura general y se dejaron de lado con la finalidad de obtener un diagrama más efectivo.