DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Número 2

ReciBarFiesta

Ingeniería del Software I+II

Grupo 7

Integrante	LU	Correo electrónico
Ciruelos Rodríguez, Gonzalo	063/14	gonzalo.ciruelos@gmail.com
Costa, Manuel José Joaquín	035/14	manucos94@gmail.com
Gatti, Mathias Nicolás	477/14	mathigatti@gmail.com
Maddonni, Axel Ezequiel	200/14	axel.maddonni@gmail.com
Thibeault, Gabriel	114/13	gabriel.eric.thibeault@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción

El trabajo práctico se basa en el desarrollo de una aplicación llamada ReciBarFiesta. El trabajo consiste de dos entregas, la primera concierne a la planificación y la segunda a la arquitectura. La parte de planificación consiste en:

- un plan del proyecto,
- una lista de casos de uso,
- un análisis de riesgos.

En este trabajo presentamos toda esa información (en un orden que nos pareció más sensato) y justificamos nuestras decisiones.

2. Casos de uso

Código	Caso de uso	Horas Hombre
CU01	Incluyendo un evento cultural	30
CU02	Eliminando un evento cultural	8
CU03	Buscando evento	56
CU04	Rankeando evento	60
CU05	Logueándose	59
CU06	Compartiendo información en redes sociales	8
CU07	Editando un evento cultural	8
CU08	Registrándose	20
CU09	Verificando contenido inapropiado automáticamente	48
CU10	Verificando contenido inapropiado manualmente	8
CU11	Reservando hoteles	56
CU12	Reservando vuelos	48
CU13	Enviando publicidad	20
CU14	Visualizando caminos	48
CU15	Realizando reserva	56
CU16	Filtrando resultados	20
CU17	Obteniendo estadísticas sobre los usuarios	32
CU18	Combinando eventos	32

Tabla 1 Casos de uso

2.1. Diagrama de Casos de uso

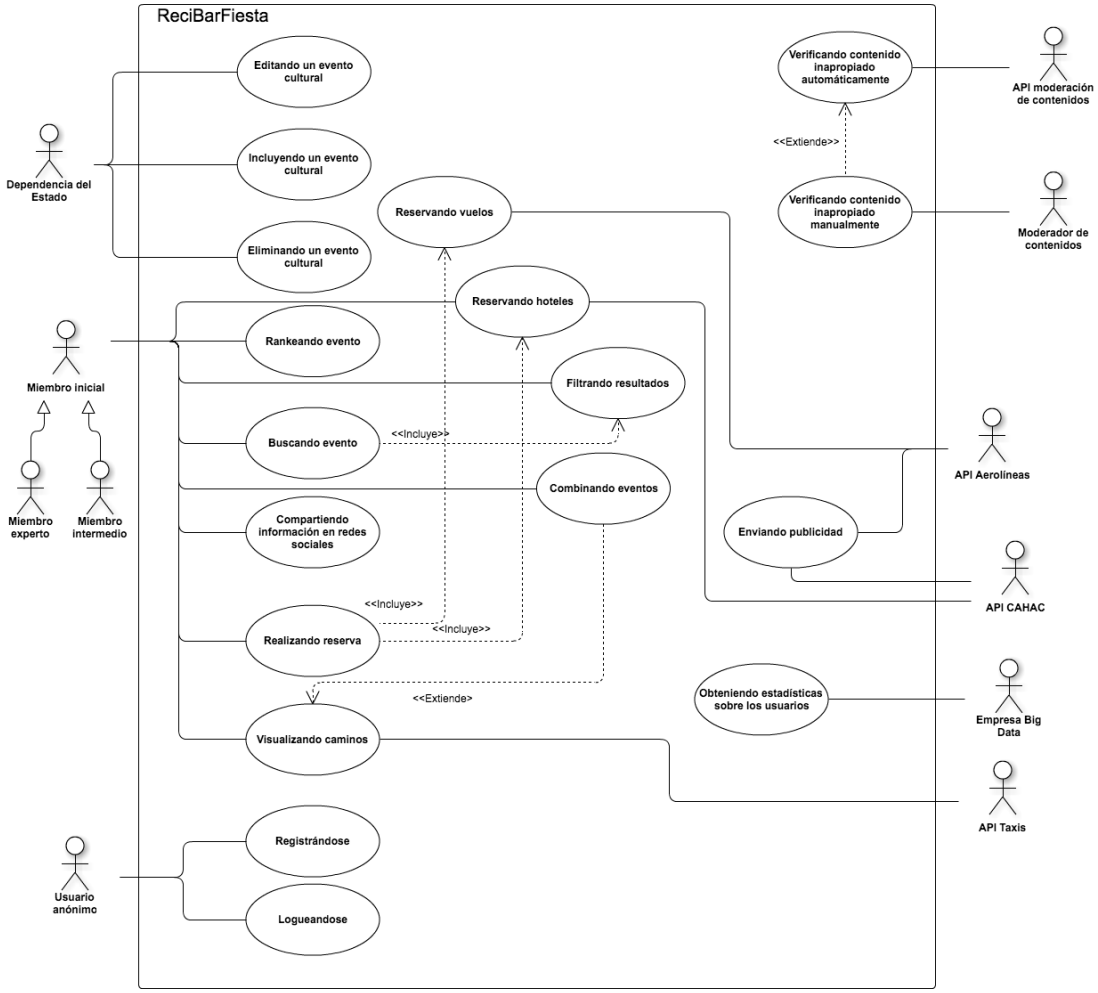

Figura 1: Diagrama de Casos de Uso de ReciBarFiesta

2.2. Descripción de los Casos de uso

1. **Incluyendo un evento cultural:** se refiere a la funcionalidad para agregar nuevos eventos, como recitales, charlas, festivales, etc. Solo lo pueden hacer usuarios autorizados (inicialmente solo miembros de dependencias del Estado).
2. **Eliminando un evento cultural:** se refiere a la funcionalidad por la cual una dependencia del Estado puede eliminar un evento cultural previamente creado.
3. **Editando un evento cultural:** se refiere a la funcionalidad por la cual una dependencia del Estado puede modificar los datos de un evento cultural previamente creado.
4. **Rankeando evento:** se refiere a la funcionalidad que le permite a un miembro identificado de la comunidad calificar un evento en distintas categorías como, por ejemplo, “Organi-

zación", "Calidad" y "Ubicación". Solo usuarios identificados podrán realizar esta acción, y se ponderará la calificación de acuerdo a la categoría del usuario (inicial, intermedio o experto).

5. **Buscando evento:** se refiere a la funcionalidad por la cual los usuarios de la aplicación buscarán eventos, pudiendo filtrar los resultados por una serie de criterios (por ejemplo: fecha, temática, ubicación, etc) usando el caso de uso "Filtrando resultados" (CU16). Una descripción detallada puede hallarse en la tabla 2.

Caso de uso: Buscando evento	
Actor: Usuario de la Aplicación (básico, intermedio, avanzado, anónimo)	
Curso Normal	**Curso Alternativo**
1) El usuario ingresa un texto en el buscador de eventos.	1.1 El usuario en esta instancia puede agregar un filtro de modo que la búsqueda se realice ya con el filtro incluido.
2) El sistema busca eventos cuyo nombre contenga o tenga una relación con el texto ingresado.	2.1 Si el usuario escribió un texto con caracteres inválidos el sistema muestra un error y pide al usuario que ingrese nuevamente un texto.
3) El sistema muestra los resultados obtenidos. Los resultados se muestran en una lista con los eventos y los links que lleven a las vistas detalladas de cada uno de ellos. Además,	3.1 Si el sistema no encuentra resultados para el texto ingresado, muestra la vista de resultados vacía con un mensaje informando que no hubo resultados para el texto ingresado. (Fin del caso)
4) El usuario puede agregar un filtro a la la lista de resultados: fecha, temática, ubicación, etc. seleccionando el filtro correspondiente y con su valor asociado (según el tipo de filtro) (CU14: Filtrando eventos)	
5) El sistema filtra y muestra los resultados según el filtro configurado.	
6) Fin del caso	

Tabla 2 Descripción detallada de CU05:Buscando evento

6. **Compartiendo información en redes sociales:** se refiere a la funcionalidad por la cual los miembros de la comunidad podrán compartir los diferentes eventos disponibles en la aplicación en diferentes redes sociales como Facebook o Twitter, de manera que otras personas que no necesariamente tienen la aplicación puedan conocer su existencia.

7. **Logueandose:** se refiere a la funcionalidad por la cual los usuarios anónimos con una cuenta previamente creada podrán identificarse con el sistema, lo cual les habilitará a tener información personalizada (por ejemplo, según el barrio en el que viven), valorar eventos, y poder realizar reservas de vuelos y hoteles.

8. **Registrándose:** se refiere a la funcionalidad a través de la cual los usuarios anónimos pueden crearse una cuenta en el sistema, para eventualmente poder identificarse y aprovechar los privilegios que esto acarrea.

9. **Verificando contenido inapropiado automatizado:** se refiere a la funcionalidad con la que el sistema utiliza la API de moderación de contenidos. La misma pone puntajes de 1 (contenido inapropiado), 2 (contenido dudoso, proceder a verificación manual) o 3 (contenido seguro)

10. **Verificando contenido inapropiado manualmente:** se refiere a la funcionalidad para que un moderador pueda determinar si un determinado contenido es apropiado o no. Este caso de uso solo se activa para aquellos contenidos que son calificados con 2 por CU9.

11. **Enviando Publicidad:** se refiere a la funcionalidad que le permite a los sponsors económicos de la aplicación enviar su material publicitario para ser mostrado por la página.

12. **Visualizando Caminos:** se refiere a la funcionalidad que permite a los usuarios visualizar el o los caminos (en caso de que se hayan seleccionado múltiples eventos) desde su posición actual hasta cada evento. Se ofrecen diferentes tipos de rutas: caminando, en auto particular, en taxi (usando la aplicación del gobierno de la Ciudad), y en transporte público. Eventualmente pueden combinarse eventos (CU18).

13. **Reservando Hoteles:** se refiere a la funcionalidad por la cual la API del sistema de la Cámara de Hoteles y Afines de la Ciudad (CAHAC) podrá registrar reservas de hoteles efectuadas por usuarios de la aplicación.

14. **Reservando Vuelos:** se refiere a la funcionalidad por la cual la API del sistema de Aerolíneas Argentinas podrá registrar reservas de vuelos efectuadas por usuarios de la aplicación.

15. **Obteniendo estadísticas sobre los usuarios:** se refiere a la API desarrollada para que las empresas de Big Data puedan obtener los datos estadísticos sobre los usuarios.

16. **Filtrando resultados:** se refiere a la funcionalidad que le permite a los usuarios de la aplicación filtrar listas de eventos según diversos criterios a determinar.

17. **Realizando Reserva:** se refiere a la funcionalidad por la cual los miembros de la comunidad podrán seleccionar y reservar vuelos/hoteles para participar de eventos publicados en la aplicación utilizando los servicios provistos por Aerolíneas Argentinas, para vuelos, y de la Cámara de Hoteles y Afines de la Ciudad, para alojamientos. Una descripción detallada de este caso de uso puede encontrarse en la tabla 3.

Caso de uso: Realizando reserva	
Actor: Usuario de la Aplicación (básico, intermedio, avanzado)	
Curso Normal	**Curso Alternativo**
1) El usuario selecciona un vuelo asociado a un evento desde la lista de resultados.	
2) El sistema muestra la interfaz de reserva asociada al vuelo seleccionado.	
3) El usuario completa los datos necesarios para la reserva.	
4) El sistema envía los datos de la reserva a la API provista por Aerolíneas Argentinas.	4.1 Si el usuario ingresó un dato inválido el sistema muestra un mensaje de error y se recarga el formulario de reserva.
5) La API de Aerolíneas Argentinas se encarga de registrar la reserva asociada al usuario de la aplicación.	
6) Fin del caso	
Nota: El comportamiento es análogo para realizar reservas de hoteles.	

Tabla 3 Descripción detallada de CU17: Realizando reserva

18. **Combinando eventos:** se refiere a la funcionalidad que extiende al caso CU10, permitiendo combinar eventos de forma tal que se obtenga el mejor camino que pasa por todas las ubicaciones.

3. Riesgos

Riesgo 1

- **Descripción:** Dado que se pueden dar problemas de conexión y se requiere que todos los días extraigan los datos (por limitaciones de espacio) se podría llegar a fallar en entregar los datos pedidos a las compañías de big data.
- **Probabilidad:** Baja
- **Impacto:** Medio
- **Exposición:** Baja
- **Mitigación:** Tener el espacio necesario para almacenar, como mínimo, el doble de la información que esperamos almacenar en promedio.
- **Plan de contingencia:** En caso de que el servicio de big data este caído por varios días y no queramos perder información, se debería hacer backup manual de los datos antes de que sean sobreescritos.

Riesgo 2

- **Descripción:** No tener la integración con Aerolíneas y CAHAC hecha para el 2017.
- **Probabilidad:** Media
- **Impacto:** Alto
- **Exposición:** Alta
- **Mitigación:** Completar la integración en las primeras dos iteraciones del desarrollo. Asignar a los mejores programadores y más recursos a realizar esta tarea.
- **Plan de contingencia:** En caso que muchos eventos imprevistos suceden y no se llegue a tiempo, se podría pedir una extensión a Aerolíneas con la suficiente anticipación. Además, se podría hacer que el equipo abandone el desarrollo de otras funcionalidades y se aboque en su totalidad a finalizar la integración.

Riesgo 3

- **Descripción:** Incapacidad del sistema de moderación principal para procesar los pedidos en tiempo real (Tanto de congestión como por caída del sistema).
- **Probabilidad:** Alta (Depende fuertemente del sistema).
- **Impacto:** Baja
- **Exposición:** Media
- **Mitigación:** Realizar estudios para tener una excelente aproximación de tráfico, de tal manera de poder comunicarlo a la empresa que provee el servicio. Tener un servicio secundario, de tal manera que si el primario se cae o funciona mal, podamos usarlo.

- **Plan de contingencia:** En caso que todo falle y no podamos analizar comentarios en tiempo real, podemos almacenarlos en una cola y para que sean analizados más adelante cuando todo vuelva a la normalidad. En caso que el sistema tarde mucho en volver a la normalidad, perderemos comentarios.

Riesgo 4

- **Descripción:** Perder la conexión a internet.
- **Probabilidad:** Baja
- **Impacto:** Alto
- **Exposición:** Media
- **Mitigación:** Tener una conexión secundaria (o más de una) de tal manera que si se cae, podamos usar esa.
- **Plan de contingencia:** Deberíamos tener un sistema rotatorio de on-calls, de tal manera que si se cae internet, en cualquier momento se le avise al on-call y este se ocupe de trasladar el servicio a una nueva conexión.

Riesgo 5

- **Descripción:** Filtración de datos privados
- **Probabilidad:** Baja
- **Impacto:** Alto
- **Exposición:** Media
- **Mitigación:** Diseñar un sistema de detección de intrusiones. Enviar datos al servicio de big data de forma encriptada y des-asociada del nombre del usuario.
- **Plan de contingencia:** Almacenar los datos personales de las cuentas de las personas de forma encriptada, de tal manera que si se filtran no sean legibles.

Riesgo 6

- **Descripción:** Usuarios creados con la intención de manipular la valoración de los eventos
- **Probabilidad:** Alta
- **Impacto:** Bajo
- **Exposición:** Media
- **Mitigación:** Pedir datos personales a la hora de crear la cuenta (Por ejemplo DNI o CUIT).
- **Plan de contingencia:** Diseñar un sistema que detecte que un evento está recibiendo muchas valoraciones y avise al on-call para que realice una revisión manual de la situación.

Riesgo 7

- **Descripción:** Renuncia de personal clave del equipo de desarrollo.
- **Probabilidad:** Baja
- **Impacto:** Medio
- **Exposición:** Baja
- **Mitigación:** Distribuir el trabajo de manera de que nadie esté demasiado sobrecargado y todos conozcan todo lo que se hace.
- **Plan de contingencia:** En caso de renuncias, reorganizar el equipo de tal manera que el programador que más conozco el trabajo del programador que renunció retome sus tareas.

Riesgo 8

- **Descripción:** Que el tráfico promedio de usuarios sea más del previsto y los sistemas internos estén continuamente sobrecargados.
- **Probabilidad:** Baja (Suponiendo una buena estimación inicial)
- **Impacto:** Alto
- **Exposición:** Media
- **Mitigación:** Hacer estimaciones lo mejor posibles.
- **Plan de contingencia:** Diseñar un balanceador de cargas y usarlo. Tener varias instancias del servidor corriendo en paralelo y totalmente replicadas.

Riesgo 9

- **Descripción:** Si la cantidad de ventas de hoteles y boletos de avión no fuera la suficiente, Aerolíneas podría retirar su inversión.
- **Probabilidad:** Baja (Muy difícil de estimar).
- **Impacto:** Medio
- **Exposición:** Media
- **Mitigación:** Hacer que la integración con Aerolíneas sea impecable. Diseñar publicidades llamativas pero poco invasivas.
- **Plan de contingencia:** Cuando la aplicación esté funcionando, invertir tempranamente en hardware y ahorrar dinero para poder hacer una inversión grande en caso que Aerolíneas Argentinas retire la suya.

4. Plan de Proyecto

4.1. WBS

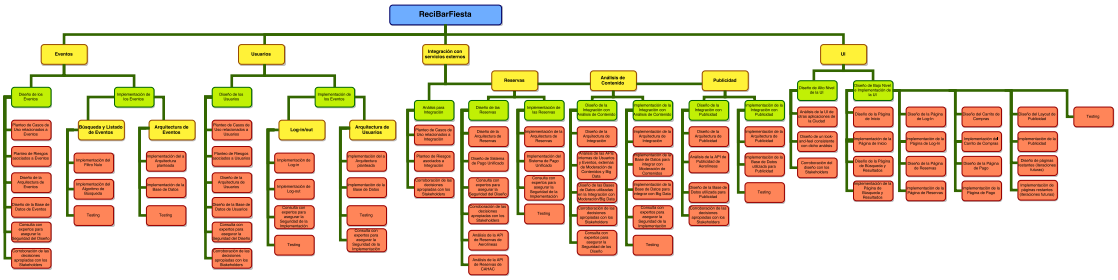

Figura 2: WBS

En la figura 2 puede observarse el diagrama del WBS. Los nodos amarillos representan productos; los verdes, procesos; los rojos, tareas, independientemente de que sean productos o procesos. Cabe realizar algunas aclaraciones respecto del diagrama. El diseño de la arquitectura del sistema se realizará para la próxima entrega; algunas de las subdivisiones en tareas se basan en dicho diseño, por lo que se observan múltiples instancias de tareas placeholder de la forma "Implementación de la Arquitectura de X". Tras realizar dicha arquitectura, éstas se deberían separar en las tareas más granulares a realizar. Es decir, en el WBS nos limitamos a presentar una visión de alto nivel de las subdivisiones del proyecto, que serían refinadas al proceder con el proceso de diseño. El enfoque más específico se reserva para los productos, procesos y tareas correspondientes a los Casos de Uso que fueron designados a la primera iteración.

4.2. Primera iteración

Para reflejar el orden en que llevaremos a cabo la primera iteración realizamos un diagrama de Gantt (figura 3) el cual modela, entre otras cosas, las dependencias, duraciones y fechas de inicio de cada tarea; sirviendonos de guía al llevar a cabo el proyecto y para identificar puntos delicados en nuestro plan, utilizando criterios como el de camino crítico.

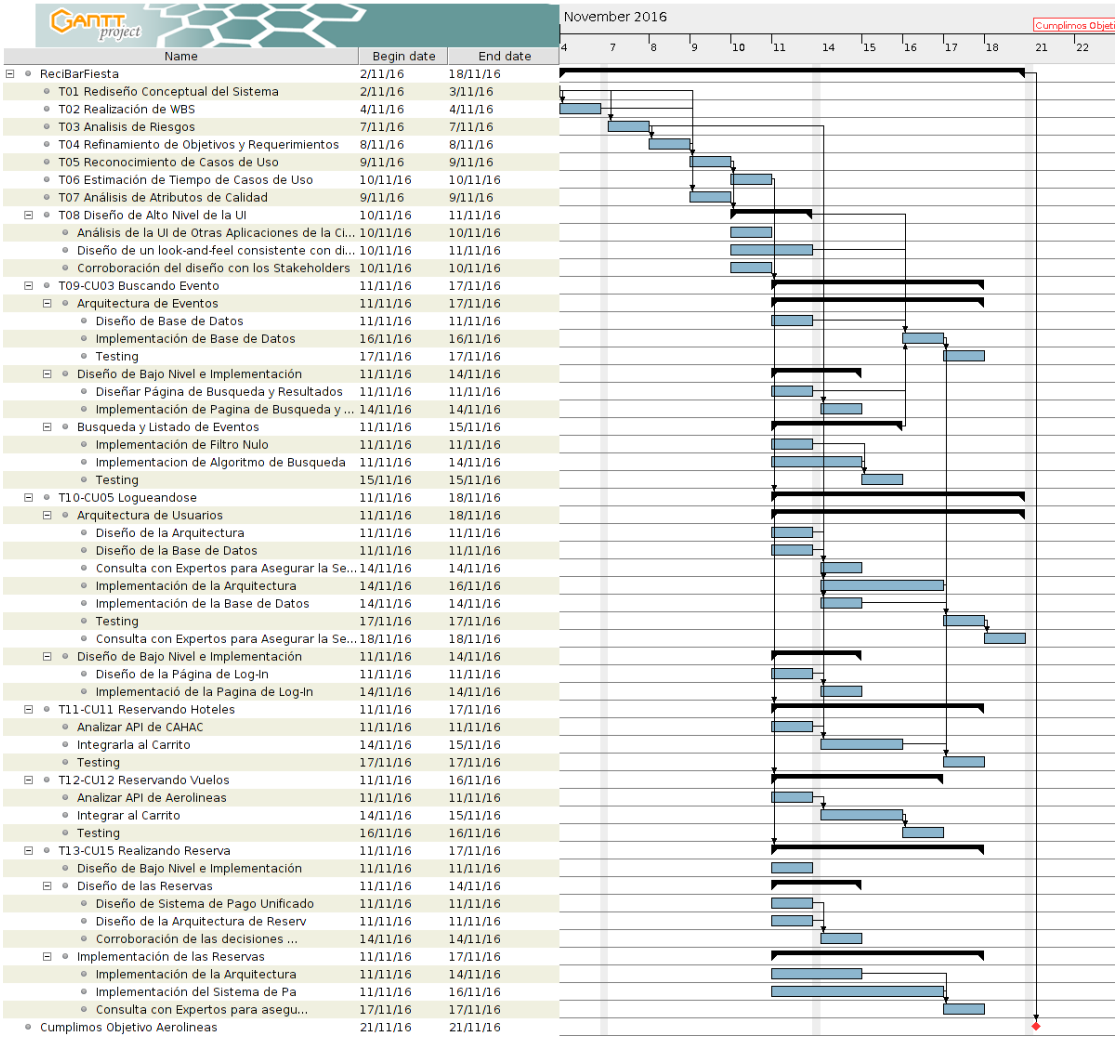

Figura 3: Diagrama de Gantt para la primer iteración del proyecto

4.2.1. Justificación

Como vimos antes, deberíamos priorizar tener la integración con Aerolíneas Argentinas terminada lo más rápido posible, de tal manera de mitigar el riesgo de no tenerla terminada para 2017. Nótese que es el único riesgo que encontramos al cual tenemos exposición alta. De esta manera, pusimos en la primera iteración todo lo relacionado con la compra de pasajes de avión y reserva de hoteles. Esta es la parte más delicada de la integración. El cumplimiento de este objetivo se puede ver reflejado como un hito en nuestro diagrama de Gantt

Además, incluimos los otros dos casos de uso a la iteración porque son fundamentales para toda la aplicación en general, y para la integración con Aerolíneas en particular.

4.3. Resto de las iteraciones

En las siguientes iteraciones deberíamos ir completando el resto de las funcionalidades, prestando atención a implementar los mecanismos de mitigación de los riesgos de exposición media (y bajo, pero más adelante), además de sus planes de contingencia, siempre en caso de que sean por software.

En la Segunda iteración, deberíamos finalizar la integración con Aerolíneas que comenzamos en la primera iteración. Esto se basaría en diseñar y desarrollar la integración de las publicidades de Aerolíneas en nuestra aplicación. Esto no fue incluido en la primera iteración porque el equipo no hubiera dado a basto a completar el trabajo, que hubiera requerido demasiadas horas hombre. Recordemos que está el riesgo de que algún ingeniero clave renuncie. También se agregan funcionalidades como el filtrado de resultados y visualización de caminos para lograr tener un entregable que ponga a prueba algunos de los objetivos principales de la aplicación. En esta iteración dejamos algo de tiempo dedicado al rediseño y reimplementación de la aplicación luego de escuchar el feedback del entregable.

La Tercera y última iteración de Elaboración comprende el diseño e implementación de las interacciones entre los eventos y los usuarios. La creación, edición y eliminación de eventos, rankings y demás.

La Cuarta iteración es de construcción, se espera dedicar gran parte del tiempo a implementar, testear y realizar deployments. Los casos de uso de esta iteración requieren bastante la interacción con agentes externos, ya que entre ellos están la capacidad de compartir información en redes sociales, verificar contenido inapropiado automática y manualmente. También se realiza la obtención de estadísticas de los usuarios y posterior manejo de su envío a las Empresas de Big Data.

4.3.1. Casos de uso por iteración

Primera Iteración (Elaboración)

- CU03 (56 Horas Hombre)
- CU05 (59 Horas Hombre)
- CU11 (56 Horas Hombre)
- CU12 (48 Horas Hombre)
- CU15 (56 Horas Hombre)

Duración: Dos semanas y tres días

Segunda Iteración (Elaboración)

- CU13 (20 Horas Hombre)
- CU14 (48 Horas Hombre)
- CU16 (20 Horas Hombre)

- CU08 (20 Horas Hombre)

Duración: Dos semanas y tres días

Tercera Iteración (Elaboración)

- CU18 (32 Horas Hombre)
- CU04 (60 Horas Hombre)
- CU07 (8 Horas Hombre)
- CU01 (30 Horas Hombre)
- CU02 (8 Horas Hombre)

Duración: Dos semanas y tres días

Cuarta Iteración (Construcción)

- CU06 (8 Horas Hombre)
- CU09 (48 Horas Hombre)
- CU10 (8 Horas Hombre)
- CU17 (32 Horas Hombre)

Duración: Tres semanas

5. Especificación de Escenarios

5.1. Escenarios de Seguridad

1. Máxima confidencialidad respecto a la información sobre los usuarios
   - **Fuente:** Un atacante interno o externo no autorizado
   - **Estímulo:** Intento no autorizado de obtener datos sobre los usuarios
   - **Artifact:** Comunicación entre el sistema central y el usuario, o datos internos del sistema sobre los usuarios
   - **Entorno:** Online
   - **Respuesta:** Los datos no son accesibles al atacante en tiempos razonables
   - **Medida de respuesta:** El tiempo requerido para que el atacante consiga la información deseada está, al menos, en el orden de los siglos
2. Los datos de las estadísticas son anónimos
   - **Fuente:** Un atacante externo potencialmente autorizado (como las empresas de big data)
   - **Estímulo:** Intento de asociar las estadísticas de uso con un usuario o grupo de usuarios en concreto
   - **Artifact:** Estadísticas de actividad de los usuarios
   - **Entorno:** Online
   - **Respuesta:** La identidad de los usuarios está oculta, i.e. las estadísticas son anónimas
   - **Medida de respuesta:** El ataque no logra asociar correctamente a un usuario con sus estadísticas en un 99.99 % de los casos.
3. Exclusividad de las estadísticas sobre usuarios para la empresa de big data
   - **Fuente:** Un individuo (u organización) externo no autorizado
   - **Estímulo:** Intento de acceder a información privilegiada
   - **Artifact:** Estadísticas de actividad de los usuarios
   - **Entorno:** Online
   - **Respuesta:** Los datos no son accesibles al atacante en tiempos razonables
   - **Medida de respuesta:** El tiempo requerido para que el atacante consiga la información deseada está, al menos, en el orden de años
4. Solo usuarios autorizados pueden subir información de eventos
   - **Fuente:** Individuo identificado no autorizado
   - **Estímulo:** Intento de subir información sobre eventos
   - **Artifact:** sistema ReciBarFiesta
   - **Entorno:** Online
   - **Respuesta:** El sistema impide la acción
   - **Medida de respuesta:** El intento fracasa el 99.999 % de las veces

5. Detección de usuarios falsos creados solo para generar popularidad de algún evento o desprestigiar a otro

   - **Fuente:** Individuo con múltiples identificaciones
   - **Estímulo:** Intento de forzar el nivel de popularidad de un evento maliciosamente
   - **Artifact:** Sistema de rankeo de eventos
   - **Entorno:** Online
   - **Respuesta:** El sistema previene que dichos individuos puedan abusar del sistema, y si falla en prevenirlo puede detectar su existencia y posteriormente castigarlos
   - **Medida de respuesta:** El tiempo requerido para crear una cantidad masiva de cuentas falsas que consigan el privilegio suficiente de poder modificar sensiblemente la popularidad de un bar está en el orden de las semanas (considerar que la duración de los eventos suele estar en un orden similar). De los casos que logran alterar el curso normal de una votación, el 90 % de las veces se los detecta

5.2. Escenarios de Disponibilidad

1. Comunicación constante con la API de moderación de contenidos

   - **Fuente:** Externa
   - **Estímulo:** Crash del servicio de moderación de contenidos
   - **Artifact:** Sistema de comentarios
   - **Entorno:** Operación normal
   - **Respuesta:** El sistema pasa a un estado degradado en el cual los comentarios se encolan, quedando pendientes de moderación hasta que se recupere el sistema
   - **Medida de respuesta:** El 99.99 % de las veces no se pierde ningún comentario

2. El sistema idealmente no debe dejar de funcionar nunca

   - **Fuente:** interna
   - **Estímulo:** falla por omisión reiterada (crash)
   - **Artifact:** servicio del sistema
   - **Entorno:** Operación normal
   - **Respuesta:** el sistema detecta la falla y se pasa a modo degradado (perdiendo la funcionalidad que falló) durante el tiempo que toma la recuperación
   - **Medida de respuesta:** la disponibilidad de un servicio debe ser de al menos el 99.99 %

5.3. Escenarios de Modificabilidad

1. Extensible para agregar visualizaciones de caminos

   - **Fuente:** Desarrollador
   - **Estímulo:** Intención de agregar y/o modificar los algoritmos de visualización de caminos utilizados

- **Artifact:** La funcionalidad y/o la performance del sistema
- **Entorno:** Tiempo de diseño
- **Respuesta:** Cambio efectuado sin efectos secundarios
- **Medida de respuesta:** Se requieren menos de 200 horas hombre para llevar a cabo el cambio

2. Modificación de la API para moderación de contenidos por mejoras continuas:

- **Fuente:** Desarrollador
- **Estímulo:** Intención de adaptar el sistema a una nueva interfaz de la API de moderación de contenidos
- **Artifact:** Módulo de moderación de contenidos del sistema
- **Entorno:** Tiempo de diseño
- **Respuesta:** Adaptación realizada sin efectos secundarios para el sistema
- **Medida de respuesta:** La modificación pudo realizarse afectando únicamente a un módulo del sistema

5.4. Escenarios de Performance

1. El sistema no debe tener ningún tipo de demoras en las búsquedas de eventos con un nivel de tráfico normal

- **Fuente:** Externa
- **Estímulo:** Llegada periódica de múltiples búsquedas de eventos
- **Artifact:** Servicio de búsqueda de eventos
- **Entorno:** Carga normal
- **Respuesta:** Procesamiento de las búsquedas
- **Medida de respuesta:** Latencia total de la operación de a lo sumo 0.5 segundos

2. El sistema no debe quedarse cargando aún con un tráfico elevado

- **Fuente:** Externa
- **Estímulo:** Llegada periódica de múltiples búsquedas de eventos
- **Artifact:** Servicio de búsqueda de eventos
- **Entorno:** Sistema sobrecargado
- **Respuesta:** Procesamiento de las búsquedas
- **Medida de respuesta:** Latencia total de la operación de a lo sumo 1.5 segundos

6. Arquitectura

6.1. Diagramas de Componentes y Conectores con Deployment

6.1.1. Nivel 0

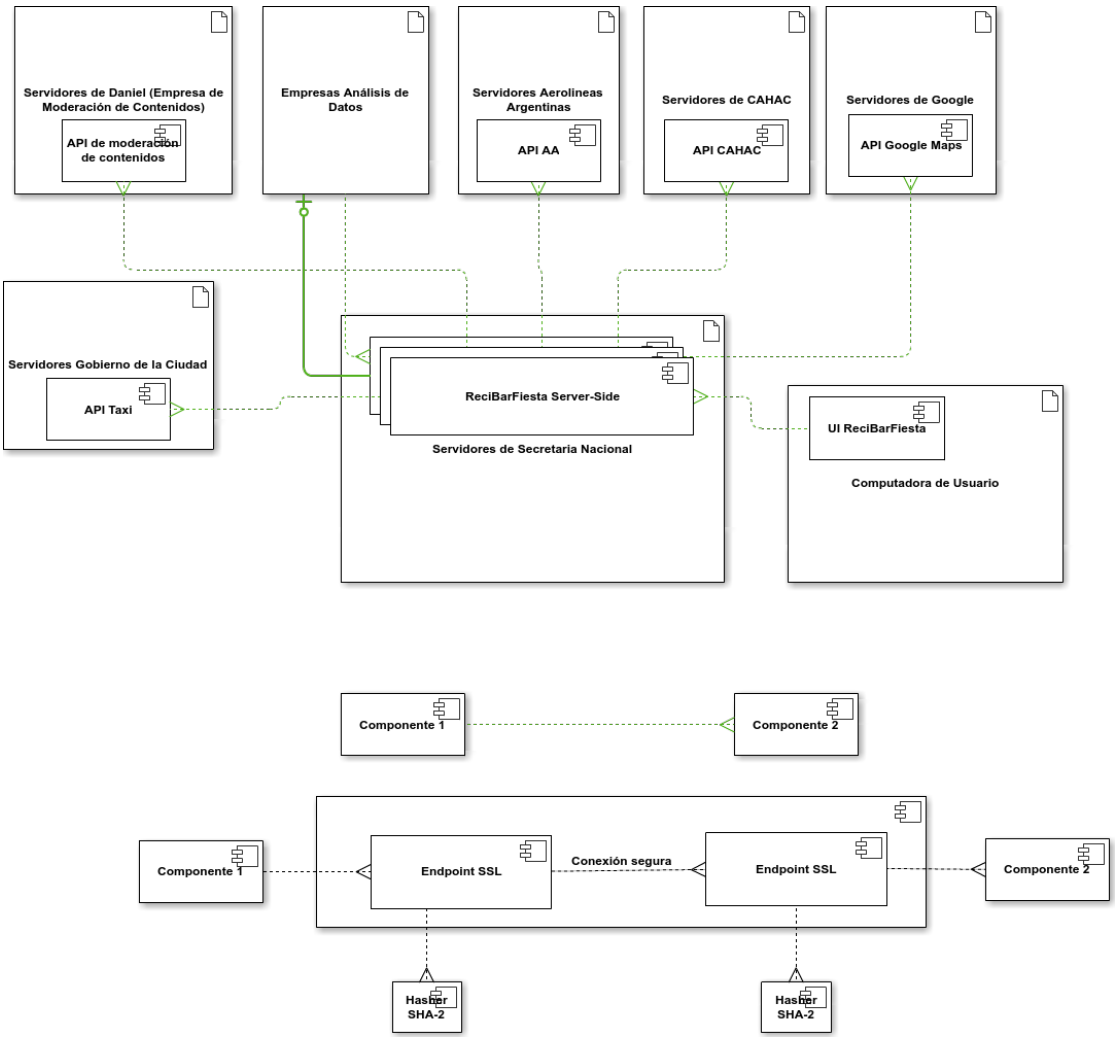

Figura 4: Vista de Nivel 0. Explicación del conector seguro. (Hacer zoom para visualizar correctamente)

6.1.2. ReciBarFiesta

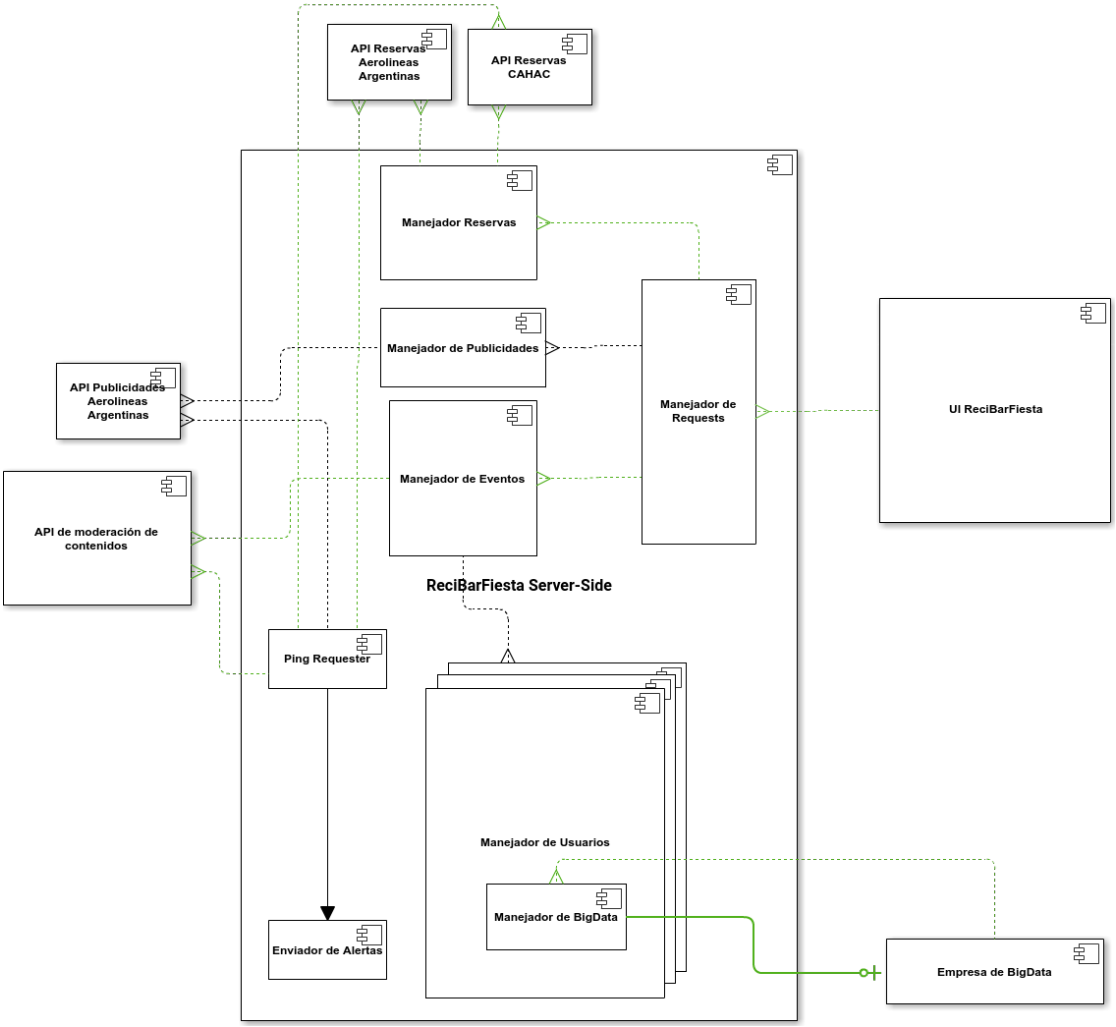

Figura 5: Vista de Nivel 1 de ReciBarFiesta del lado del servidor. (Hacer zoom para visualizar correctamente)

6.1.3. Manejador de Eventos

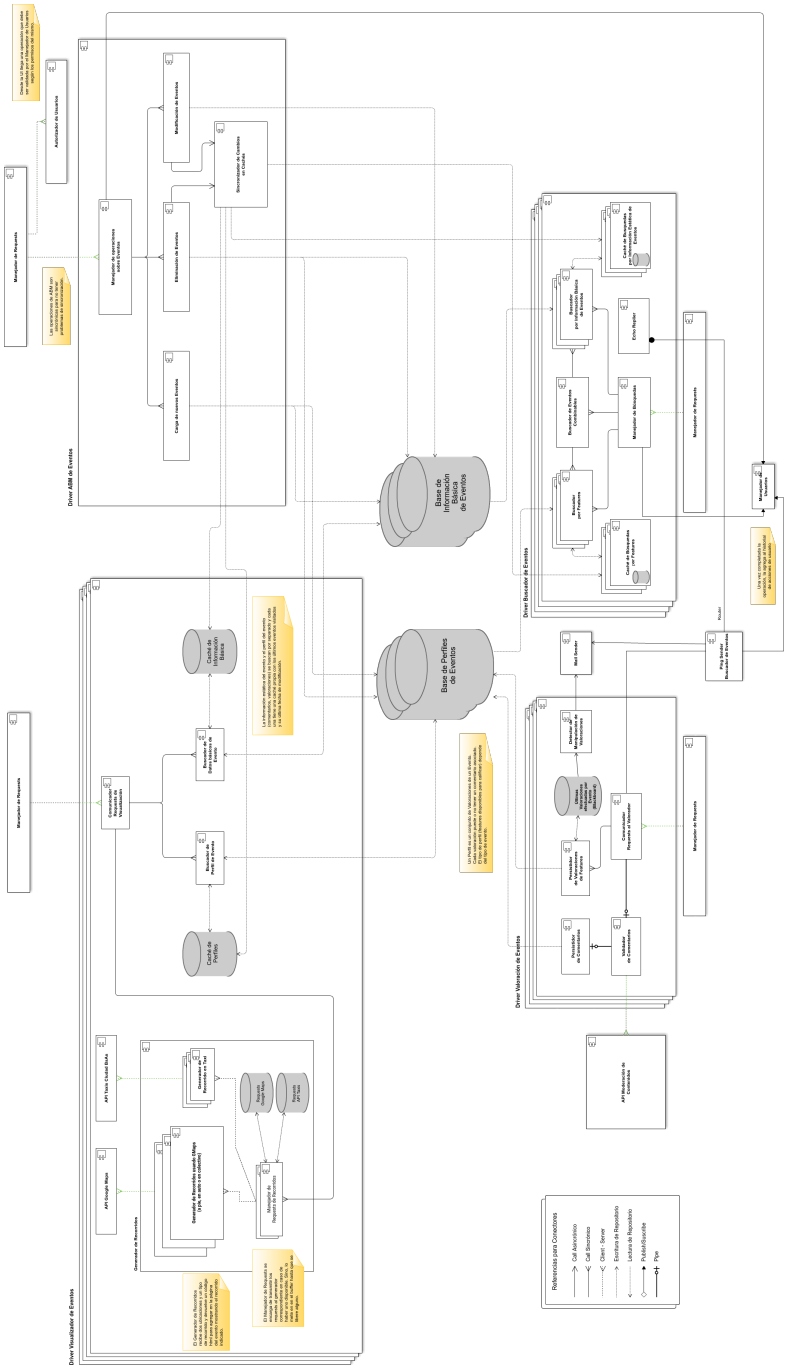

Figura 6: Vista del Manejador de Eventos. (Hacer zoom para visualizar correctamente)

6.1.4. Manejador de Usuarios

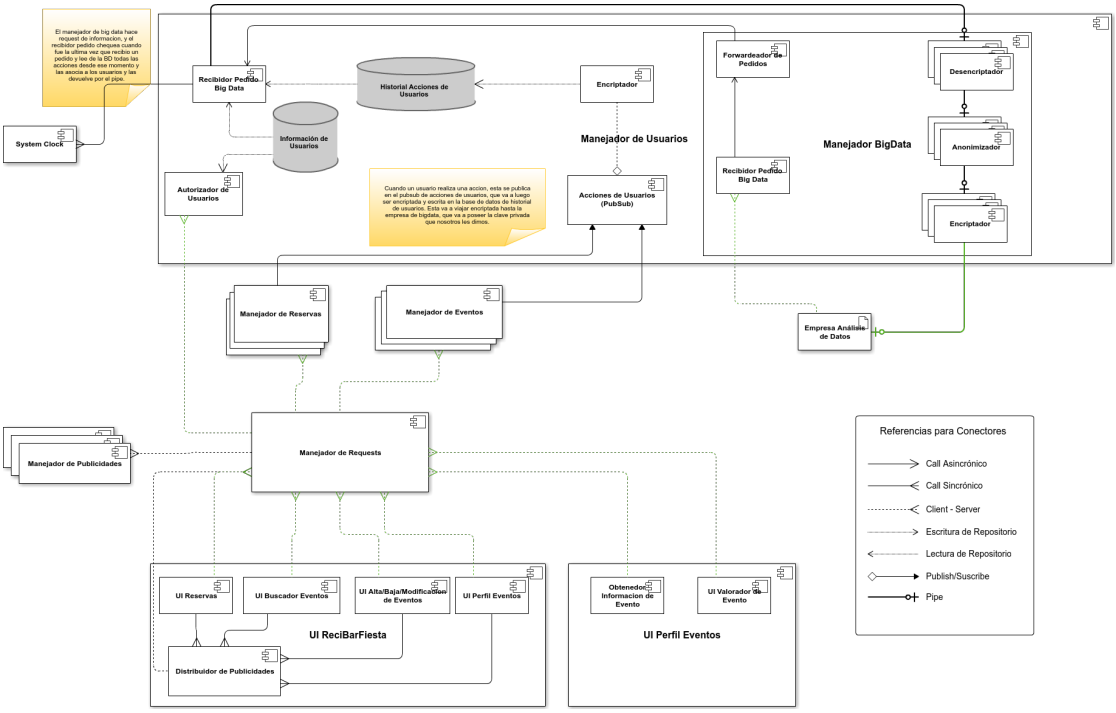

Figura 7: Vista del Manejador de Usuarios (con Manejador BigData). Vista de UI ReciBarFiesta. (Hacer zoom para visualizar correctamente)

6.1.5. Manejador de Reservas

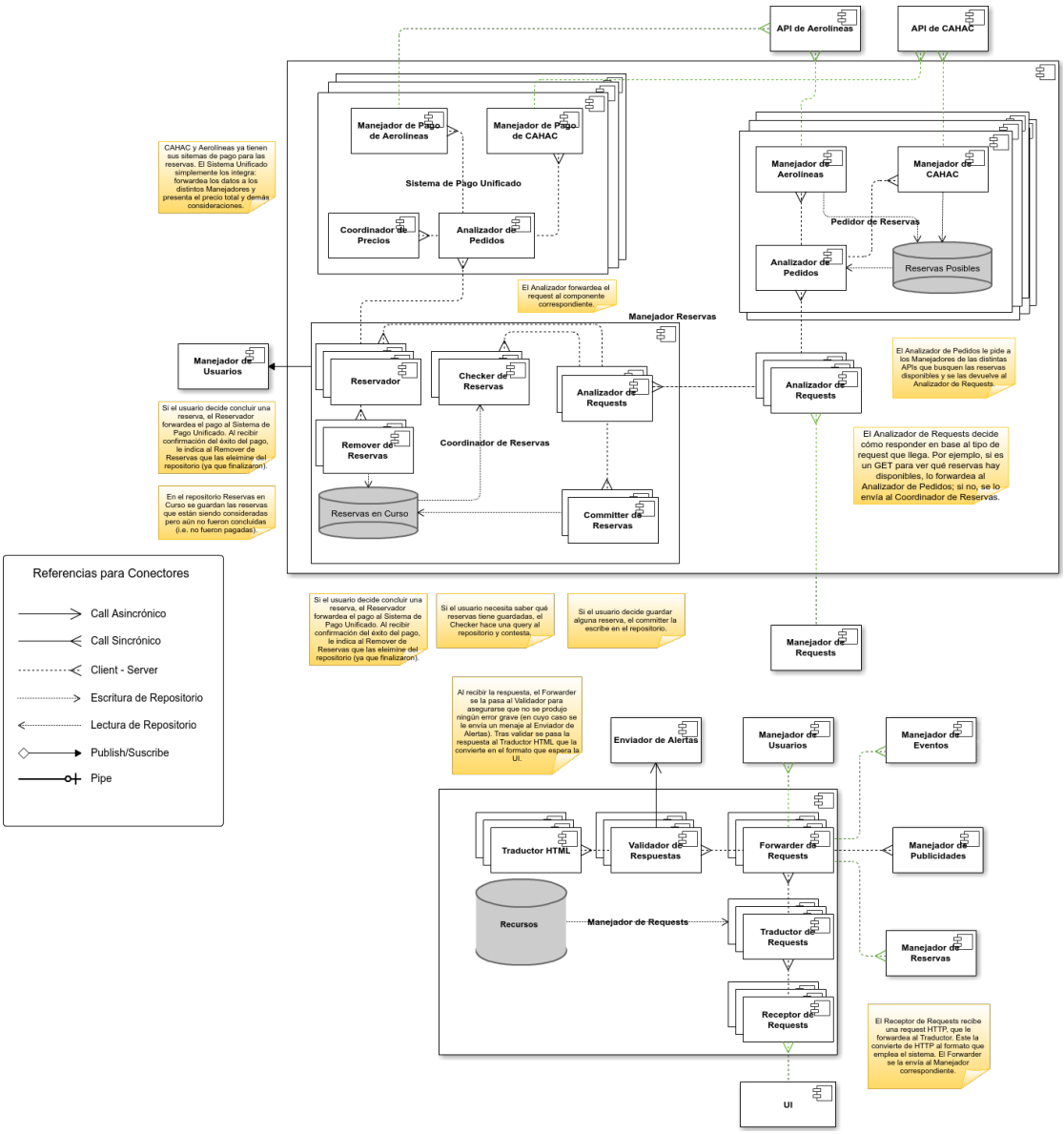

Figura 8: Vista del Manejador De Reservas. Vista del Manejador De Requests. (Hacer zoom para visualizar correctamente)

6.1.6. Manejador de Publicidades

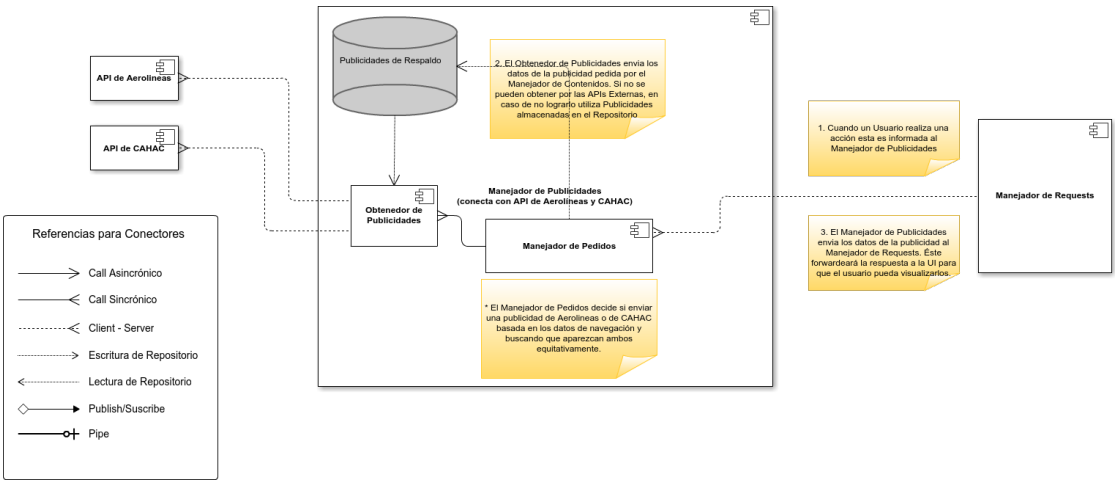

Figura 9: Vista del Manejador de Publicidades. (Hacer zoom para visualizar correctamente)

6.2. Explicación de la Arquitectura

6.2.1. Manejador de Requests

La comunicación entre el usuario, desde la página web de la aplicación, y el server se realiza mediante HTTP. El Receptor de Requests recibe un request HTTP y se lo forwardea al Traductor, que lo convierte al formato del sistema. Luego el Forwarder lo envía al componente correspondiente.

Una vez que se recibe la respuesta del componente, se valida. Si se le encuentra algún error, se envía un mensaje al Enviador de Alertas. Si no, se la manda al Traductor HTML, que la convierte en el formato que espera el cliente.

El propósito de este componente es limitar la interacción con el cliente a un solo punto del server. Adicionalmente, los demás componentes se pueden desligar de la representación HTML de sus respuestas.

6.2.2. Manejador de Eventos

El manejador de eventos es el encargado de responder a las distintos requests que envía un usuario de la aplicación desde la interfaz para ejecutar una acción sobre un evento: agregar uno nuevo, modificar, eliminar, valorar, visualizar y búsqueda de eventos.

Notar que los componentes de Visualización, Valoración y Búsqueda de Eventos están replicados para poder realizar muchas operaciones concurrentemente. Sin embargo, el componente de ABM no está replicado para evitar que se generen inconsistencias en la información de los eventos.

Visualización de Eventos

Es el encargado de obtener la información cargada en las bases de datos asociada a un evento en particular, que incluye los datos básicos del evento (así llamaremos a datos como el Nombre, Ubicación, Horario, Tipo de evento, etc.) y los datos del "Perfil del Evento"(que incluye valoraciones por cada feature según el tipo de evento y comentarios asociados a esas valoraciones realizados por los usuarios).

Supongamos que un usuario quiere acceder a la Vista de un evento en particular, al hacer click en un evento el sistema envía el ID del mismo al Manejador, en particular al componente Driver Visualizador de Eventos. Este componente recibe el request y procede a buscar la información asociada al evento (perfil y datos básicos). Para ambos casos, hay un componente encargado de realizar las consultas en la base correspondiente. Además, cada uno tiene una caché asociada con los últimos eventos visitados por los usuarios. Una vez obtenida la información, en caso de que el usuario desee ver el recorrido hasta el evento, se envía al generador de recorridos la dirección actual del usuario y la dirección del evento. Este componente se encargará de interactuar con la API necesaria: GoogleMaps en caso de que sea recorrido a pie, en auto o en colectivo, o la API de Taxis de la Ciudad, para recorridos en taxi. A través de dichas APIs generará un código HTML que se enviará al Comunicador. Luego, el Comunicador del Driver de Visualización envía toda la información al Renderer HTML para generar la página correspondiente y enviársela a la interfaz para que el usuario pueda verla.

Alta/Baja/Modificación de Eventos

Desde la interfaz, un usuario decide realizar una de las operaciones de ABM para algún evento en particular. Primero, el usuario debe ser autorizado por el Autorizador de Usuarios si es que tiene los permisos necesarios para realizar la operación. Una vez chequeados sus permisos, el request llega al Driver ABM de Eventos del Manejador. El Manejador de operaciones deriva el request parseado con los datos necesarios para realizar la operación al componente correspondiente: Carga de nuevos Eventos, Eliminación, o Modificación. En caso de tratarse de Carga de un Evento, el componente chequea que el evento sea válido, y en caso afirmativo procede a agregarlo a la base de datos de Información básica y a la base de Perfiles, generando un perfil vacío. En caso de que sea una baja de un evento, el componente elimina todas las entradas del mismo en las bases de perfiles e información básica, y luego envía un request asincrónico al Sincronizador de Cambios en Cachés, encargado de actualizar todas las cachés de los componentes del Manejador de Eventos, para que no queden con datos inconsistentes. Por último, en caso de ser una modificación, se actualiza la información básica del evento una vez que haya sido validada, y luego se avisa al sincronizador de cachés para que ejecute las actualizaciones. Una vez finalizada la operación, la misma se guarda en el historial de acciones por usuario.

Valorador de Eventos

Este componente es el encargado de almacenar las valoraciones de los usuarios en el perfil del evento correspondiente. Una valoración debe incluir al menos un valor seleccionable para los features de un evento, que dependen del tipo del mismo, y además puede o no incluir un comentario asociado a dicha valoración. En caso de contener un comentario, éste se encolará para que sea validado interactuando con la API de moderación de contenidos, y luego guardado por el persistidor de Comentarios. En caso de que la conexión con la API se vuelva lenta, se guardará por un lado la valoración y el comentario asociado quedará encolado hasta que sea validado, y luego almacenado. En caso de ocurrir esto se avisará al usuario con mensaje al retornar la respuesta. Si el comentario no es apropiado, se devolverá un mensaje de error. Por otro lado, al persistir las valoraciones, éstas son guardadas temporalmente en un repositorio de últimas valoraciones efectuadas. Este repositorio es leído continuamente por un componente encargado de detectar si se están produciendo demasiadas valoraciones similares al mismo

evento; y, en caso de detectarlo, enviará un mail a través del Mail Sender para que la situación sea revisada manualmente.

Buscador de Eventos

El Driver Buscador de Eventos manejará las búsquedas realizadas por los usuarios, dividiéndolas en búsquedas por Features o búsquedas por Información Básica del evento. El Manejador de Búsquedas parseará el request del usuario y enviará la búsqueda al componente Buscador correspondiente según el tipo de la misma. Cada uno de estos componentes cuenta con una Caché donde se almacenarán las búsquedas más populares realizadas por los usuarios, que serán actualizadas periódicamente. Las cachés cuentan con tres niveles (primario, secundario, y terciario) según el nivel de popularidad de las búsquedas para garantizar performance. Una vez realizada la búsqueda, en caso de ser requerido por el usuario, el manejador enviará los resultados al Componente buscador de Eventos Combinables, para que genere búsquedas de eventos combinables (cercanos, temáticas relacionadas, etc.), que luego serán efectuadas por el Buscador correspondiente según se trate de búsquedas en la base de datos de perfiles o de información básica. Una vez obtenidos los resultados, se actualizan las cachés, se notifica al historial de usuario la acción realizada, y se devuelve una página con los resultados y links a las vistas de los eventos encontrados.

6.2.3. Manejador de Usuarios

El manejador de usuarios es el componente que se ocupa de todo lo referido a usuarios. Se ocupará, entre otras cosas, de almacenar la información de los usuarios y sus acciones relevantes en la aplicación.

Interactuará con la empresa de análisis de datos y con la UI de ReciBarFiesta.

Acciones de Usuarios (PubSub) y Encriptador

Estos dos componentes son los más importantes a la hora de guardar las acciones de los usuarios en la aplicación. Cada vez que un usuario haga una accion, el componente que la realiza deberá publicarlo en el PubSub. Hay un encriptador que se ocupa de encriptar los datos y guardarlos en la base de datos de historial de acciones de usuarios. De esta manera la información queda resguardada ante eventuales leaks de información.

Manejador BigData

Este sub-componente de Manejador de Usuarios se ocupa de recibir los pedidos de la empresa de análisis de datos y responderlos. Los pedidos los recibe un Recibidor de Pedidos de Big Data, y es forwardeado por un Forwardeador. Este pedido es recibido por el Recibidor Pedido de Big Data, que chequea cuando fue la última vez que recibió un request y devuelve todas las acciones desde ese momento por un pipe, asociando las acciones con los usuarios, que extrae del repositorio llamado Información de Usuarios.

Luego, siguiendo un estilo de pipes and filters, esa información es desencriptada, luego anonimizada (se borra la información que permita identificar personas) y luego se encripta nuevamente de manera qu esolo la empresa de análisis de datos pueda leer esos datos. Esto puede hacerse por ejemplo, con un sistema de clave público-privada, muy fácilmente.

Autorizador de Usuarios

Simplemente recibe un usuario y una contraseña y la chequea contra la base de datos de los usuarios, y devuelve un token que podrá utilizar la UI (token de sesión, utilizado por el

conector encriptado).

6.2.4. Manejador de Reservas

El Analizador de Requests decide a que subcomponente forwardear un request entrante. Si simplemente se busca saber qué reservas están disponibles para un evento dado, se lo envía al Pedidor de Reservas. El Analizador de Pedidos busca en el cache de reservas; si no se encuentra lo buscado, forwardea las requests correspondientes a los distintos manejadores de APIs externas.

El Coordinador de Reservas maneja las reservas que cada usuario está considerando pero aún no finalizó y pagó. El Analizador forwardea el request al subcomponente correspondiente. Si el usuario desea agregar una reserva, el Committer la escribe en el repositorio. El checker indica qué reservas un usuario está mantienendo actualmente. Si desea concretarlas, el Reservador forwardea el request al Sistema de Pago Unificado; si la transacción procede correctamente, el Reservador le envía un mensaje al Remover para que borre las reservas del repositorio, pues ya fueron concretadas.

Aprovecharemos que tanto CAHAC como Aerolíneas tienen sus propios sistemas de pago para las reservas; sin embargo, sería altamemente inconveniente para el usuario tener que entrar sus datos separadamente en cada sistema. Este componente unifica el pago, forwardeando los datos que entra el usuario a las diversas APIs, concretando las reservas sólo si ninguna tuvo problemas. El Coordinador de Precios presenta un sumario unificado de la transacción.

6.2.5. Manejador de Publicidades

El Manejador de Publicidades es el componente encargado de utilizar las APIs de Aerolíneas y CAHAC para conseguir las últimas Publicidades de estas compañias y entregarselas a la Interfaz de Usuario ReciBarFiesta cuando esta lo requiera, la UI comunica esta a traves de un conector client Server enviando la acción que esta realizando el usuario, su ubicación y demás.

La lógica interna consiste de un componente Manejador de Pedidos que modera entre el envío de publicidades de Hoteles y Vuelos para no excederse en el envío de un solo tipo de publicidad al mismo tiempo que intenta enviar los anuncios que mejor se adapten a las acciónes y caracteristicas del usuario.

Como se mencionó previamente es importante que la publicidad este siempre presente en el sitio web. Para lograr esto se tiene, en caso de que fallen las APIs, un repositorio de Publicidades de Respaldo que se va actualizando periódicamente.

6.2.6. UI ReciBarFiesta

La UI ReciBarFiesta tiene varios subcomponentes que se ocupan de realizar distintas acciones. Estos son:

- **UI Reservas.** Permite hacer reservas de vuelos y hoteles. Envía requests a Manejador de Requests, que luego serán forwardeadas a un Manejador de Reservas.
- **UI Buscador Eventos.** Permite buscar eventos.

- **UI Alta/Baja/Modifiación de Eventos.** Permite borrar, crear y modificar eventos. El Manejador de eventos va a chequear que la request provenga de un usuario autorizado, y en caso de que así no sea, el request va a fallar.
- **UI Perfil Eventos.** Permite visualizar eventos. Este componente a su vez tiene dos subcomponentes internos, que son **Obtenedor de Inforamción de Evento**, que se ocupa de obtener la información de un evento y mostrarla, y **UI Valorador de Evento**, que se ocupa de enviar una nueva valorización del evento.

Como la presencia de las publicidades de Aerolíneas Argentinas debe ser constante, la UI tiene dentro un distribuidor de publicidades. Cada vez que un componente de la UI entra en acción, le pide una publicidad al Distribuidor de Publicidades, que se ocupará de enviarle el respectivo request al sistema.

6.3. Aspectos particularmente importantes y cómo los resolvimos

6.3.1. ¿Cómo se cumplen los requerimientos de seguridad respecto de los servicios para las empresas enfocadas en los grandes datos?

Utilizamos varias tácticas para cumplir con los requerimientos de seguridad.

Primero tenemos todos los datos de las bases de datos encriptados, tanto de la base de datos de Información de Usuarios como de Acciones de Usuarios. Esto permite que, aunque haya una intrusión en el sistema y las bases de datos se vean expuestas, el atacante no podrá obtener ningún tipo de información sobre los usuarios. Esto es una táctica de resistencia a ataques.

Otra táctica que usamos es la desanonimización de los datos de los usuarios. Esto, desde algún punto de vista, puede verse como la táctica de separación de entidades. Esto nos permite que, por ejemplo, si hay alguna intrusión del lado de la empresa de grandes datos, no se pueda recuperar la información personal de los usuarios.

Por último, todos los enlaces entre la empresa de grandes datos y nuestro sistema son a través de una conexión segura como la que está explicada en los diagramas, utilizando por ejemplo TLS, que nos brinda autenticación y no repudio, que son dos características muy deseables para el tipo de conexión que queremos establecer.

6.3.2. ¿Cómo es la interacción con los servicios externos? (API de moderación, Aerolíneas, Cámara de Hoteles) ¿Hay disponibilidad? ¿Seguridad?

La interacción con los servicios externos es siempre mediante manejadores de esos servicios externos que nos abstraen y nos permiten reemplazarlos fácilmente.

En cuanto a disponibilidad, implementamos un Ping Requester, que cada 10 minutos envía pings a los servicios para verificar que estén disponibles y funcionando correctamente. Si no lo están, se envía un mail al on-call del equipo, reportando el problema. Esta táctica es de detección de errores, y permite disminuir el downtime.

Por otra parte, en servicios como las publicidades, implementamos caches de publicidades, que ante un problema de conexión con la API de Aerolíneas Argentinas que nos provee de publicidades, carga una publicidad de la cache de publicidades y la muestra.

En cuanto a seguridad, implementamos el conector seguro para conectarnos a todos los servicios externos. Estos conectores usan TLS, que brinda integridad, autenticación y no repudio, todo lo deseable en una conexión segura. Esto nos permite resistir muy bien a los ataques. Además, nunca se envía información personal de los usuarios. Cada vez que información de usuarios va a ser envíada fuera del sistema, es anonimizada (se le quitan los datos personales y se la deja inidentificable a una persona física) y encriptada.

6.3.3. ¿Cómo se logra perfomance para responder las búsquedas de los usuarios?

La performance se logra utilizando variadas tácticas.

La táctica más simple es la de replicación. Tenemos nuestro servidor replicado y tenemos un manejador de requests entre la UI y el server side, que actúa como distribuidor de requests y balanceador de carga.

Además, dentro del Manejador de Eventos, en la sección que concierne a la búsqueda de eventos, tenemos múltiples caches que permiten obtener la información a muchisima más velocidad que haciendo la búsqueda de cero.

Además, tenemos replicadas nuestras bases de datos, lo cual permite atender varias requests de lectura al mismo tiempo sin problemas.

7. Conclusión

7.1. Métodos usados (UP vs. Ágil)

UP y los métodos ágiles (como Scrum, que usamos para el primer TP) son iterativos incrementales. Sin embargo tienen varias diferencias.

UP se centra en la arquitectura, y facilita su refinamiento progresivo.

En el primer TP teníamos tres artefactos que guiaban el desarrollo de software: un Product Backlog, un Sprint Backlog y un Burndown Chart. Un backlog es simplemente una lista de prioridad de cosas a hacer. El Burndown Chart sirve para graficar el progreso durante la iteracción actual. Estas tres herramientas son lo que se usa para planificar y hacer el seguimiento del proyecto en Scrum.

UP, por otro lado, contiene una larga lista de documentos y herramientas para planificar el proyecto. Por ejemplo, una planificación de una iteración de UP es muchísimo más detallada que para Scrum, dado que incluye riesgos, tareas, recursos asignados a tareas y dependencias entre tareas.

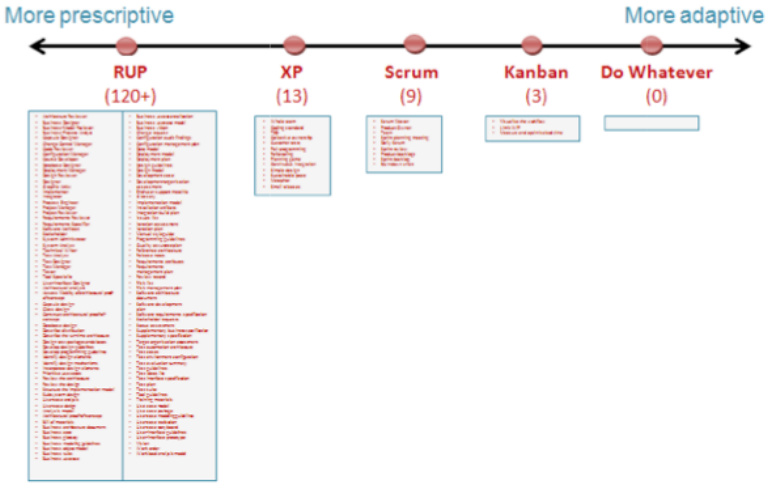

Figura 10: Comparación de varias metodologías de desarrollo.

7.2. Programming in the small y Programming in the large

Programming in the large y programming in the small describen dos aproximaciones diferentes al diseño y desarrollo de software.

Programming in the large involucra en general productos de software desarrollados por grupos a lo largo de extensas cantidades de tiempo. Con programming in the large, los diseñadores hacen hincapié en particionar el trabajo en modulos con interacciones bien definidas. Esto requiere una cuidadosa planificación además de una cuidadosa documentación. Programming in the large en general se refiere a diseñar el sistema desde un alto nivel, donde solo importa el estado de los módulos y sus interacciones, y no su implementación real.

Debido a todas estas características, en programming in the large es importante planificar a largo plazo, tener en cuenta riesgos y darle mucha importancia a la arquitectura del software. Es debido a esto que en general se utiliza una metodología como UP.

Por otro lado, programming in the small describe la actividad de escribir pequeños programas, es decir, cuyo desarrollo toma poco tiempo. En general programming in the small es llevado a cabo por personas individuales o pequeños equipos.

Debido a que en general se busca velocidad en el desarrollo de estas pequeñas piezas de software, y mejorarlas de manera incremental, programming in the small suele asociarse con métodos de desarrollo ágiles, como Scrum.

7.3. Conclusiones generales

Como conclusión general, podemos decir que los dos trabajos prácticos nos permitieron apreciar dos métodos de desarrollo y planificación de software muy distintos, que aplican a situaciones distintas.

UP permite llevar a cabo proyectos enormes, que se extenderán durante mucho tiempo. Por eso se requiere planificar todo al detalle, pensar en los riesgos, diseñar la arquitectura del software puntillosamente, de tal manera que la gran inversión de tiempo y dinero no sea en vano.

Por otro lado, las metodologías ágiles que vimos para el primer trabajo práctico, permiten desarrollar de forma muy rápida softwares pequeños. Es muy buena para desarrollar prototipos o borradores de software (por ejemplo, Most Viable Products) para luego incrementarlos de a poco, y en general proyectos donde no vale la pena pagar un alto overhead de planificación. Sin embargo, los equipos tienen que ser más pequeños que para UP (que por su alto nivel de planeamiento soporta equipos muchísimo más grandes).

Referencias

[1] Len Bass et al., *Software Architecture in Pratice*. Addison-Wesley, 2nd edition.

[2] Paul Clements et al., *Documenting Software Architecture*. SEI Series in Software Engineering, 2nd edition.