



**DEPARTAMENTO
DE COMPUTACION**

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico N°2

Twiteando para ahorrar

Ingeniería de Software II

Grupo N°2

Integrante	LU	Correo electrónico
Bianchetti, Marcelo	378/08	bianchetti.ml@gmail.com
Doldán, Juan	309/09	blacksirius@yahoo.com.ar
Masseroli, Alejandro	399/09	alemasseroli@hotmail.com
Rinaudo, Joaquín	175/09	jrinaudo@dc.uba.ar



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. Resumen del proyecto a construir	3
1.1. Stakeholders	3
1.2. Descripción del contexto	3
1.3. Objetivos: Drivers, restricciones y grado de libertad	3
1.3.1. Drivers	3
1.3.2. Restricciones	3
1.3.3. Grado de libertad	3
2. Diagramas de caso de uso	4
3. Casos de uso y estimaciones de tiempo	6
4. Iteraciones del proyecto	8
4.1. Fase de elaboración	8
4.1.1. Primera iteración de elaboración	8
4.1.2. Segunda iteración de elaboración	8
4.2. Fases de construcción	9
4.2.1. Primera iteración de construcción	9
4.2.2. Segunda iteración de construcción	9
4.2.3. Tercera iteración de construcción	9
5. Tareas de los casos de uso de la primera iteración	10
6. Riesgos del proyecto	11
7. Casos de uso del sistema a desarrollar	16
7.1. Módulo de manejo de la base de datos	16
7.2. Módulo de recolección de información	16
7.3. Módulo de filtro de datos	17
7.4. Módulo de administración de recursos de aplicación	17
7.5. Módulo de predicción de consultas	17
7.6. Módulo de autenticación	18
7.7. Administración de posibles ofertas falsas	18
7.8. Detección de ofertas falsas	18
7.9. Módulo de procesamiento de búsqueda	19
7.10. Módulo de definición de búsqueda de cliente	19
8. Arquitectura del Sistema	20
8.1. Manejador de datos de TpA	21
8.1.1. Manejo de las Bases de Datos	22
8.1.2. User's Information	23
8.1.3. Association Rules	23
8.2. Recolección de la información	23
8.2.1. Interacción con las bases de datos	25
8.3. Update Query Responder	26
8.4. Outsider Query Responder	27
8.5. Conectores	28
8.6. Diagrama completo arquitectura servidor	29
8.7. Justificación arquitectura del cliente	30
8.8. Conexión entre servidor y cliente	35
9. Atributos de calidad	37
9.1. Escenarios de atributos de calidad	37
9.1.1. Usabilidad	37
9.1.2. Rendimiento	38
9.1.3. Extensibilidad	39
9.1.4. Disponibilidad	40

9.1.5. Auditabilidad	41
9.1.6. Seguridad	41
10. Análisis	43
10.1. Programming in the large <i>vs</i> Programming in the small	43
10.2. UP <i>vs</i> SCRUM	44
10.3. Conclusiones del trabajo	45
11. Apéndice I: Referencia de conectores	46

1. Resumen del proyecto a construir

1.1. Stakeholders

1.2. Descripción del contexto

Sistemas con quienes interactúa, interfaces externas

1.3. Objetivos: Drivers, restricciones y grado de libertad

1.3.1. Drivers

Los objetivos principales del proyecto son:

- Brindar a los usuarios promociones de productos que le interese.
- Ofrecer a la comunidad información sobre los mejores lugares y precios para comprar los productos.
- Evitar ofrecer información de venta falsa o desactualizada.
- Proveer un servicio de alta disponibilidad para consultas de productos.

1.3.2. Restricciones

El proyecto cuenta con algunas restricciones como:

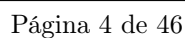
- Hasta implementar nuestro módulo de detección de ofertas falsas, deberemos utilizar el servicio ofrecido por SpamBust.
- La autenticación de usuarios se realizará mediante utilizando mediante OpenId información personal de páginas de terceros.
- La base de datos a utilizar será del la clase NO-SQL
- El sistema debe contar con un módulo para recolectar información desde otros sitios como Facebook, Pinterest.
- Los servidores deben ser construídos a partir de computadoras de escritorio.

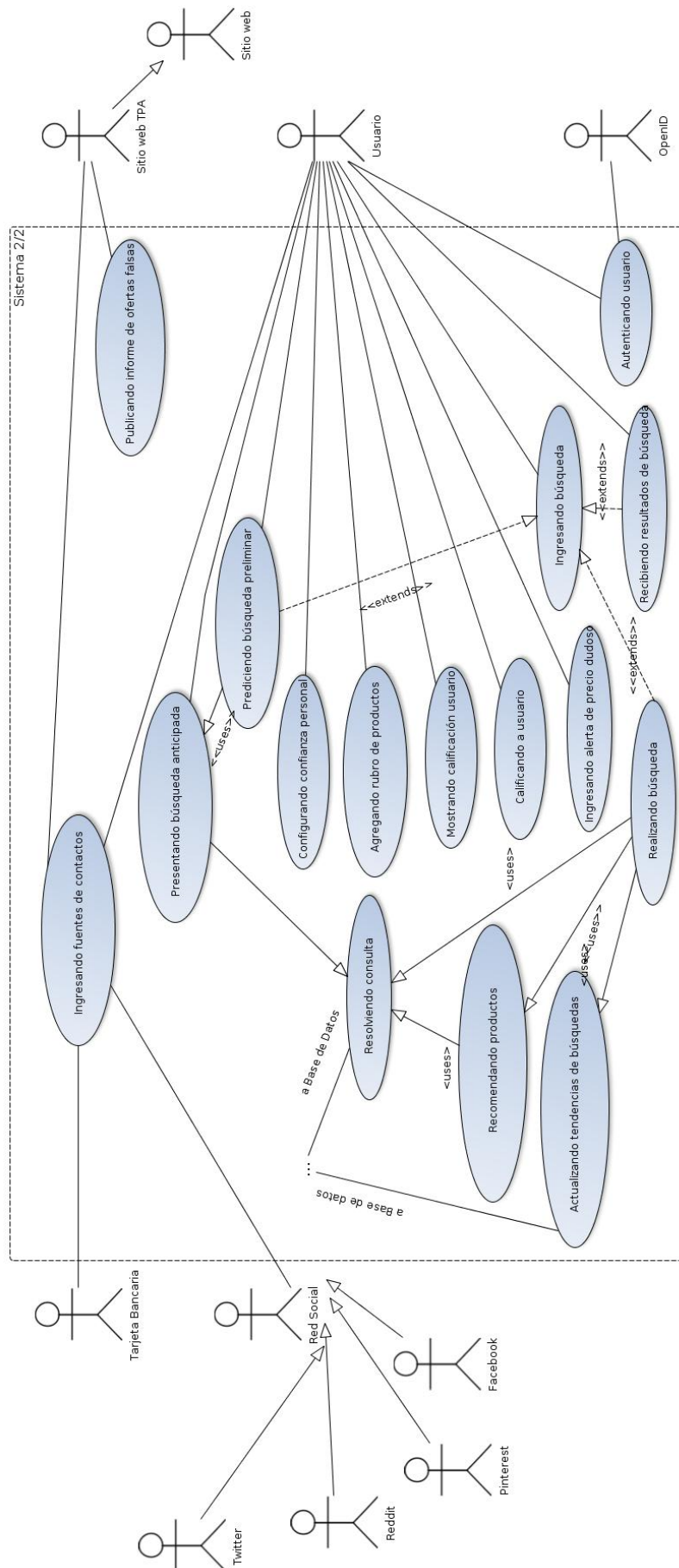
1.3.3. Grado de libertad

Aspectos del proyecto que atribuímos que permiten flexibilidad a la hora de ser desarrollado.

- La cardinalidad del equipo con que se cuenta no fue especificada, ni la composición del mismo como desarrolladores, diseñadores y testers. Por lo tanto, se realizó la asunción de que se cuenta con un equipo de 4 personas, las cuales están habilitadas para realizar tanto tareas de diseño y desarrollo, como de testing.
- El medio mediante el cual se comunicarán los clientes y servidores. Podría ser un medio WiFi, utilizando conexión 3G, SMS, etc. Se definió que los medios serían sólo utilizando WiFi y 3G pues otros medios pagos desalentarían la utilización del producto.
- La duración de las iteraciones para las fases de RUP son siempre de 15 días. Luego teniendo un día hombre de 8 horas y un grupo de cuatro personas, el trabajo que se puede planear para una semana es de alrededor de 320h o 40 días hombre.

Previo a la modularización de los casos de uso, se presentan los diagramas que representan el comportamiento del sistema.





3. Casos de uso y estimaciones de tiempo

<u>Caso de Uso</u>	<u>Estimación en horas</u> ¹
■ Manejo de la base de datos	216
• Almacenando información de venta	112
• Almacenando promociones	24
• Actualizando información de ventas	32
• Eliminando información de ventas falsa	16
• Resolviendo consultas	32
■ Recolección de información	136
• Recolectando información de redes sociales	72 ²
• Recolectando información mediante Web mining	40
• Utilizando API de recolección de publicaciones de venta	24
■ Filtro de datos	136
• Filtrando publicaciones no relevantes	88
• Filtrando promociones desde publicaciones	48
■ Administración de recursos de aplicación	72
• Dando de baja información falsa	16
• Agregando rubro de productos	24
• Modificando reglas de asociación entre productos	32
■ Predicción de consultas	144
• Presentando búsqueda anticipada	48
• Actualizando tendencias de búsquedas	72
• Obteniendo búsqueda preliminar	24
■ Autenticación	72
• Autenticando usuario	32
• Configurando confianza personal	16
• Ingresando fuentes de contactos	24
■ Administración de posibles ofertas falsas	24
• Ingresando alerta de precio dudoso	8
• Publicando informe de posibles ofertas falsas	16
■ Detección de ofertas falsas	128
• Descargando informe de información falsa de SpamBust	24
• Detectando información falsa en la base de datos	104
■ Procesamiento de búsqueda	88
• Realizando búsqueda	40
• Recomendando productos	48

¹En realidad la estimación se hace en base a días hombre de ocho horas

²3 días hombre, siendo 24 horas por cada red social y asumiendo que lo hacemos para Facebook, Twitter y Reddit.

■ Definición de búsqueda de cliente	108
• Mostrando calificación usuario	4
• Calificando a usuario	16
• Ingresando búsqueda	24
• Visualizando resultados de búsqueda	64

4. Iteraciones del proyecto

4.1. Fase de elaboración

Los módulos que se deberían desarrollar en la fase de elaboración son aquellos que eliminan o disminuyen los mayores riesgos del proyecto junto con los que definen la arquitectura base del proyecto. Se decidió que en las fases de elaboración se desarrollarían los módulos de:

- **Manejo de la base de datos:** Consideramos hacer primero este módulo debido a que se tiene un sistema con una tecnología nueva y que debe estar preparado para responder una gran cantidad de consultas. Este punto va a ser, claramente, una parte clave de la arquitectura base.
- **Recolección de información:** Este módulo se desarrolla primero por dos distintas razones. Primero, la recolección de datos presenta un riesgo importante para el software, pues el mismo no puede ser puesto en producción hasta contar con una cantidad importante de ventas. Además, la interacción de los distintos recolectores sobre un recurso compartido como la base de datos forma parte de decisiones importantes a tomar en una arquitectura base.
- **Definición de búsqueda de un cliente:** Este tercer módulo se cree relevante para la definición de la arquitectura base, pues define la comunicación entre el cliente y el servidor de TPA. Además, se puede enfrentar lo antes posible riesgos relacionados con la usabilidad del producto y portabilidad a distintas plataformas.

La primera fase de elaboración se enfrentará al módulo de manejo de base de datos, que creemos que es el módulo que presenta los mayores riesgos. Esto es puesto que se debe construir un sistema distribuido de base de datos complejo con nuevas tecnologías. Además, dicho sistema distribuido debe ser montado sobre computadoras de escritorio y no hardware especializado. Como es un módulo extenso, la primera iteración se enfocará únicamente en él.

En la segunda fase se desarrollará el módulo de recolección de información por las razones mencionadas previamente, junto con el módulo de procesamiento de búsqueda.

Luego las fases de elaboración son las siguientes:

4.1.1. Primera iteración de elaboración

- Manejo de la base de datos

4.1.2. Segunda iteración de elaboración

- Recolección de la información
- Definición de búsqueda de un cliente

Una vez que se realizaron estas iteraciones, cumplimos con poseer una arquitectura suficientemente estable y confiable debido a que se han dirigido y considerado la mayoría de los elementos riesgosos del proyecto. También, al tener estos módulos desarrollados se tiene que gran parte de la visión del producto se encuentra estable (pues ya está implementada la funcionalidad más general); luego, los demás módulos pueden ser construidos en sucesivas fases de elaboración.

4.2. Fases de construcción

4.2.1. Primera iteración de construcción

Los módulos a construir en la primera fase de elaboración son los siguientes:

- Procesamiento de búsqueda
- Filtro de datos
- Autenticación

La decisión de incluir el módulo de procesamiento de búsqueda se debe a que el motor de la recomendación de productos resulta complejo, por lo que es beneficioso desarrollarlo en un primer paso. Se elige también el módulo que filtra los datos basados en que ambos resuelven los restantes riesgos identificados. Al tener los filtros de datos y las búsquedas, se puede empezar a recolectar informaciones de ventas y promociones desde los medios sociales y la web desde temprano, logrando así tener un volumen de ventas importante al momento de pasar el producto a producción.

Por otro lado, el sistema de autenticación de terceros conlleva problemas de seguridad y disponibilidad que se deben definir lo antes posible.

4.2.2. Segunda iteración de construcción

Para la segunda fase se tendrán que desarrollar los siguientes módulos:

- Administración de posibles ofertas falsas
- Detección de ofertas falsas
- Predicción de consultas

Si bien los módulos de detección de información falsa y su consiguiente administración son bastante importantes para el consumidor final, no se pueden realizar antes ya que cualquier sistema de análisis de la información (que se precie de ser medianamente bueno) necesita de una base sobre la cual comenzar a trabajar. Es por esto que se realiza en medio de la fase de construcción.

El sistema de predicción de consultas es algo bastante simple y que no pareciera presentar riesgos de ningún tipo para el sistema total, ni su arquitectura.

4.2.3. Tercera iteración de construcción

La última fase del proyecto tendrá:

- Administración de recursos de la aplicación

Se pospone dicho módulo para el final ya que las funcionalidades del mismo no presentan ningún riesgo asociado ni parecerían afectar la arquitectura.

5. Tareas de los casos de uso de la primera iteración

En el cronograma se nota la descomposición en tareas de los casos de uso del módulo de manejo de la base de datos. Para no recargar demasiado la descomposición en tareas, se optó por sólo realizar una descomposición minusciosa sobre el caso de uso Almacenando informaciones de venta, que como es el primer caso de uso a desarrollar tarda más que el resto. Esto se debe a que es necesario realizar un análisis y diseño completo de la base de datos. Una vez que la misma fue diseñada e implementada, los otros casos de uso de dicho módulo son fácilmente implementables. Se quiere notar que hay una dependendencia entre casos de usos de una misma iteración, esta dependendencia se debe a dos razones: La primera se basa en que en orden de tener los otros casos de uso implementados en una menor cantidad de tiempo (a pesar de ser similares) se debe de haber completado ya uno que incluyera tareas de diseño y investigación que definan como será el manejo de la base de datos. La segunda razón se debe a cuestiones de recursos, debido al equipo con que se cuenta, todos los casos de uso del módulo no pueden ser desarrollados en paralelo por lo que al menos uno debería ser implementado previamente.

6. Riesgos del proyecto

A continuación se listan algunos de los riesgos del proyecto, ordenados según exposición y clasificación.

P	Descripción	P	I	E	Plan	Clasificación
1	Dado que se planea que mucha gente utilice la aplicación entonces probablemente los tiempos de búsqueda de productos de una semana promedio será mayor a 30 segundos, generando un descontento en los usuarios de la misma	Alta	Alto	Alta	[M] El DBA será responsable de la configuración de un balanceador de carga con una cantidad de servidores calculada para poder manejar el volumen de datos [C] El DBA deberá encargarse de la configuración del sistema distribuido de la base de datos para duplicar el espacio y redistribuir los datos de las antiguas bases para aumentar el nivel de paralelismo	Técnico
2	Dado que se debe recolectar información de venta de otros sitios entonces probablemente se cuente con menos de 1 millón de registros de ventas de productos debido a que se desarrolló el módulo de recolección tarde en el proyecto y la gente esté desconcenta con los resultados obtenidos en sus búsquedas	Alta	Alto	Alta	[M] El PM del proyecto deberá tener funcionando los recolectores antes de pasados dos meses de iniciado el proyecto, y los mismos recolectando información [C] El encargado del área de comercio deberá pactar con empresas competidoras por informaciones de ventas utilizables y seguras a cambio de porcentaje en la ganancia del producto en materias de publicidad	Técnico

Cuadro 1: M: Mitigación , C: Contingencia

P	Descripción	P	I	E	Plan	Clasificación
3	Dado que el proyecto cuenta con poca inversión y es muy complicado entonces la medición de costo de performance a mitad del cronograma del proyecto sea menor a uno, indicando problemas de presupuesto	Alta	Alto	Alta	[M] El grupo de desarrollo implementará un sistema de publicidades para ofrecer a empresas de distintos rubros, aumentando el flujo de capital. El espacio para publicidades se cobrará a un precio más bajo durante el desarrollo del producto [C] El gerente del área comercial de la empresa negociará con empresas ofertas de publicidades para aumentar el nivel de las publicidades.	Comercial
4	Dado que se vive en un país donde se tiene restricciones a las importaciones entonces probablemente los servidores pedidos no puedan entrar al país dejando al proyecto sin un servidor central con redundancia	Media	Alto	Alta	[M] El DBA obtendrá sus servidores de empresas de países donde el nivel de importación hacia Argentina es más bajo, como India, aumentando la probabilidad que dicha importación pueda ser aceptada [C] En caso de no contar con los servidores tres meses antes de comenzar el proyecto, el gerente de base de datos deberá desarrollar un sistema que utilice las computadoras que ya tenga en su posesión	Management

Cuadro 2: M: Mitigación , C: Contingencia

P	Descripción	P	I	E	Plan	Clasificación
5	Dado que hay una gran diversidad tecnológica entonces es probable que utilizando una sola plataforma se obtenga un nivel de carga del servidor menor al 30 % desperdiciando mucho uso y llegando a menor cantidad de usuarios	Media	Alto	Alta	[M] El líder del subproyecto realizará una investigación de cuales son las plataformas más utilizadas por el mercado y determinará el desarrollo del cliente para dicha plataforma [C] El encargado del desarrollo de la parte del cliente deberá seleccionar un conjunto mínimo de plataformas más utilizadas y lograr la portabilidad de la aplicación hacia algunas de ellas	Técnico
6	Dado que es un nuevo producto para los usuarios entonces probablemente la gente tarde más de 1 minuto en poder realizar una búsqueda de productos, generando un descontento hacia la interfaz del usuario de la aplicación por su mal diseño	Media	Alto	Alta	[M] El sub-gerente del área de recursos humanos tomará prototipos previos de la UI y realizará con grupos externos pruebas de uso con ellos [C] El área de desarrollo desarrollará tutoriales de video que ayuden al uso del producto para subir en la página web	Management

Cuadro 3: M: Mitigación, C: Contingencia

P	Descripción	P	I	E	Plan	Clasificación
7	Dado que está planeado tercerizar la autenticación entonces hay posibilidades de que dichos servicios estén caídos por más de 10 horas semanales	Baja	Alto	Media	[M] El gerente del área técnica deberá elegir un sistema de autenticación que garantice un servicio estable, tal que no esté caído más de 10 horas semanales [C] El gerente de desarrollo deberá desarrollar un sistema de login propio con utilidades mínimas para utilizar en esos casos y buscar un servidor más estable	Técnico
8	Dado que se planea utilizar autenticación de sitios de terceros entonces probablemente se tenga una cantidad de ataques promedio mayores a 100 semanales para obtener los datos privados de los otros sitios sin tener que atacarlos a ellos directamente, disminuyendo la performance del sistema	Media	Medio	Media	[M] El gerente del área técnica deberá formar un grupo encargado de la seguridad a fin de garantizar una baja tasa de ataques efectivos. [C] En caso de que los ataques sean incontrolables por el departamento de seguridad, el gerente del área técnica buscará contratar servicios externos encargados de tal tarea..	Técnico

Cuadro 4: M: Mitigación, C: Contingencia

P	Descripción	P	I	E	Plan	Clasificación
9	Dado que SpamBust es una empresa privada externa al proyecto entonces probablemente la eficacia de detección de ofertas falsas sea menor a 8 de cada 10 ofertas válidas en promedio y el sistema contenga mucha información falsa que produzca descontento en los clientes	Baja	Medio	Baja	<p>[M] El gerente del área técnica le pedirá a SpamBust que provea un servicio con la calidad deseada y lo amenazará con sacarle la exclusividad en el proyecto de no cumplirlo</p> <p>[C] El gerente del área de desarrollo deberá formar un grupo encargado de desarrollar una alternativa al software externo</p>	Externo
10	Dado que Independiente ha venido teniendo campañas malísimas este último tiempo entonces probablemente descienda haciendo que Juan se enoje y no quiera trabajar más	Alta	Alta	Alta	<p>[M] Le pedimos a Don Julio que nos de una manito</p> <p>[C] Si el rojo desciende, armamos bardo</p>	Del Mundo real

Cuadro 5: M: Mitigación, C: Contingencia

7. Casos de uso del sistema a desarrollar

7.1. Módulo de manejo de la base de datos

Caso de uso: Almacenando información de venta

Descripción: El sistema almacena en un medio no volátil la información de venta generada.

Actores participantes: Sistema, Base de datos

Usa: Actualizando información de venta

Caso de uso: Almacenando promociones

Descripción: El sistema almacena en un medio no volátil las promociones encontradas.

Actores participantes: Sistema, Base de datos

Caso de uso: Resolviendo consulta

Descripción: La base de datos obtiene los productos de una consulta particular hacia una tabla particular, sea de productos, promociones o estadísticas de búsquedas más probables.

Actores participantes: Sistema, Base de datos

Caso de uso: Actualizando información de ventas

Descripción: El sistema se encarga de actualizar información de ventas que quedó deprecada en base a nuevas obtenciones.

Actores participantes: Sistema, Base de datos

Caso de uso: Eliminando información de ventas falsa

Descripción: Dado un informe de ventas falsas, el sistema se encarga de eliminarla de las bases de datos.

Actores participantes: Sistema, Base de datos.

7.2. Módulo de recolección de información

Caso de uso: Recolectando información de redes sociales

Descripción: El sistema se conecta a cada red social y obtiene las más recientes posibles publicaciones de ventas provenientes de cada una de ellas.

Actores participantes: Red social (Facebook, Pinterest, Reddit, Twitter), Sistema

Usa: Filtrando publicaciones no relevantes, Almacenando información de venta, Recolectando promociones

Caso de uso: Recolectando información mediante Web mining

Descripción: El sistema recorre la web en busca de ofertas de productos y las almacena en su base de datos.

Actores participantes: Sitio de búsqueda Google, Sitio de búsqueda Bing, Sistema

Usa: Filtrando publicaciones no relevantes, Almacenando información de venta, Recolectando promociones

Caso de uso: Utilizando API de recolección de publicaciones de venta

Descripción: Usuario del sitio Web suben información de venta a través del mismo. Y esta información es enviada al sistema utilizando nuestra API.

Actores participantes: Sitio Web, sistema

7.3. Módulo de filtro de datos

Caso de uso: Filtrando publicaciones no relevantes (INTERNO)

Descripción: El sistema filtra aquellas publicaciones, descargadas de un medio social, que no presenten ventas de productos válidas y genera la información de venta para aquellas que sí lo hagan.

Actores participantes: Sistema

Caso de uso: Filtrando promociones desde publicaciones (INTERNO)

Descripción: A partir de publicaciones obtenidas, el sistema determina cuáles son promociones y las almacena en la base de datos.

Actores participantes: Sistema

Usa: Almacenando promociones

7.4. Módulo de administración de recursos de aplicación

Caso de uso: Dando de baja información falsa

Descripción: Un administrador ingresa la baja de una información que fue detectada como falsa y posteriormente analizada. El sistema elimina dicha información de su base de datos.

Actores participantes: Sistema, Administrador, Base de Datos

Caso de uso: Agregando publicidad

Descripción: Un administrador una nueva publicidad de producto por la cual se modificarán las reglas de recomendación para que nuevas consultas devuelvan recomendaciones de productos publicitados.

Actores participantes: Sistema, Administrador, Base De Datos

Usa: Modificando reglas de asociación de producto

Caso de uso: Agregando rubro de productos

Descripción: Un administrador agrega un nuevo rubro, a partir de este momento, nuevos usuarios pueden consultar dichos rubros y también los recolectores deberán obtener publicaciones de productos pertenecientes al mismo.

Actores participantes: Administrador, Sistema, Base de Datos

Caso de uso: Modificando reglas de asociación entre productos

Descripción: El administrador puede realizar altas, bajas y modificaciones de las reglas del sistema para las recomendaciones de promociones y productos.

Actores participantes: Administrador, Sistema

7.5. Módulo de predicción de consultas

Caso de uso: Presentando búsqueda anticipada

Descripción: A medida que el usuario ingresa parámetros para su búsqueda, el sistema exhibe aquellas búsquedas que predice, a partir de las consultas con mayor probabilidad de la base de datos, que son la que el usuario desea.

Actores participantes: Usuario, Sistema

Usa: Resolviendo consulta

Caso de uso: Actualizando tendencias de búsquedas

Descripción: A partir de una búsqueda, el sistema renovará sus estadísticas para predecir

futuras búsquedas.

Actores participantes: Sistema, Base de datos

Caso de uso: Obteniendo búsqueda preliminar

Descripción: El sistema capta la búsqueda parcial del usuario y envía búsqueda predicha.

Actores participantes: Sistema, usuario

Usa: Presentando búsqueda anticipada

7.6. Módulo de autenticación

Caso de uso: Autenticando usuario

Descripción: Un usuario se autentica en el sistema utilizando OpenID, quien realizará las verificaciones pertinentes para asegurar la identidad.

Actores participantes: Sistema, usuario, OpenID

Caso de uso: Configurando confianza personal

Descripción: Un usuario configura la confianza que tiene en diversas fuentes para futuros resultados de búsqueda

Actores participantes: Sistema, usuario

Caso de uso: Ingresando fuentes de contactos

Descripción: Un usuario autenticado inicia migración de contactos desde un medio hacia el sistema.

Actores participantes: Sistema, Usuario, Tarjeta bancaria, Red Social, Sitio de búsqueda

7.7. Administración de posibles ofertas falsas

Caso de uso: Ingresando alerta de precio dudoso

Descripción: Un usuario ingresa una denuncia de precio sobre una información de venta a partir de los resultados que le fueron presentados en una búsqueda y el sistema almacena la misma.

Actores participantes: Sistema, usuario

Caso de uso: Publicando informe de posibles ofertas falsas

Descripción: Los martes el sistema publica en su página web un informe con la cantidad de ofertas falsas detectadas, productos con precios dudosos, información contradictoria y alertas de engaños reportados por los usuarios.

Actores participantes: Sistema, Sitio Web TPA

7.8. Detección de ofertas falsas

Caso de uso: Descargando informe de información falsa de SpamBust

Descripción: SpamBust ingresa al sistema el informe en un formato xml de las promociones e ventas que cree falsas. El sistema almacena las mismas en una base de datos para luego publicarlas a fin de mes.

Actores participantes: Sistema, SpamBust, Base de datos

Caso de uso: Detectando información falsa en la base de datos

Descripción: El sistema escanea la base de datos y analiza la información cargada en el mismo con el fin de detectar posibles informaciones de ventas y promociones falsas. Luego las almacena para publicarlas a fin de mes.

Actores participantes: Sistema, base de datos

7.9. Módulo de procesamiento de búsqueda

Caso de uso: Realizando búsqueda

Descripción: Dada una búsqueda de un usuario, el sistema obtiene de sus bases de datos la información de ventas de la búsqueda y las promociones relacionadas por las reglas de asociación.

Actores participantes: Sistema

Usa: Recomendando productos, Resolviendo consulta

Caso de uso: Recomendando productos

Descripción: A partir de una búsqueda, el sistema busca en su base de datos aquellas promociones que son relacionadas según las reglas de asociación de productos. Dichas promociones son mostradas al usuario.

Actores participantes: Sistema, base de datos

Usa: Resolviendo consulta

7.10. Módulo de definición de búsqueda de cliente

Caso de uso: Mostrando calificación usuario

Descripción: Un usuario solicita la reputación de otro y el sistema se la devuelve.

Actores participantes: Sistema, usuario

Caso de uso: Calificando a usuario

Descripción: Un usuario ingresa una calificación positiva o negativa sobre un usuario a partir de los resultados que le fueron presentados en una búsqueda y el sistema almacena la misma.

Actores participantes: Sistema, usuario

Caso de uso: Ingresando búsqueda

Descripción: El usuario ingresa los parámetros de una búsqueda en su dispositivo, el sistema toma las informaciones parciales de su búsqueda y predice búsquedas más probables. Cuando el usuario indica que quiere obtener los resultados, el servidor realiza la búsqueda de los productos y muestra los resultados obtenidos en el dispositivo del usuario.

Actores participantes: Sistema, usuario

Extiende: Obteniendo búsqueda preliminar, Realizando búsqueda ,Visualizando resultados de búsqueda

Caso de uso: Visualizando resultados de búsqueda

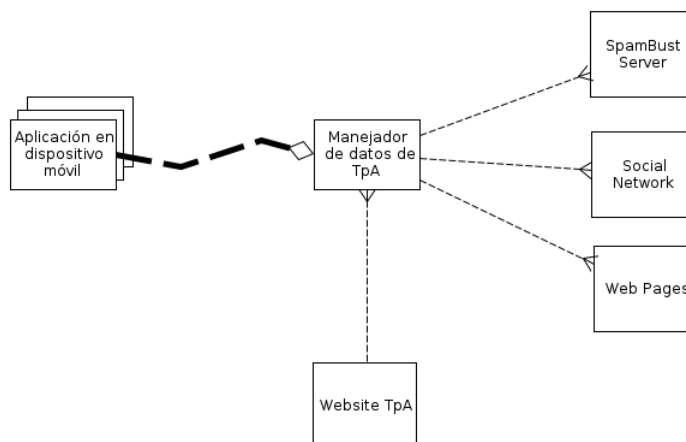
Descripción: Dados los resultados de una búsqueda, el sistema envía los mismos al dispositivo del usuario para que este los vea.

Actores participantes: Sistema, usuario

8. Arquitectura del Sistema

El diseño de la arquitectura fue bastante difícil ya que el software debía poder cumplir con muchos requisitos especificados sobre una base de hardware no preparada para ello.

Lo primero que hicimos fue considerar cuáles eran los aspectos importantes del problema. De esta forma, logramos identificar los tres componentes principales.



El conector representa al canal de comunicación entre los dispositivos móviles y la base de datos

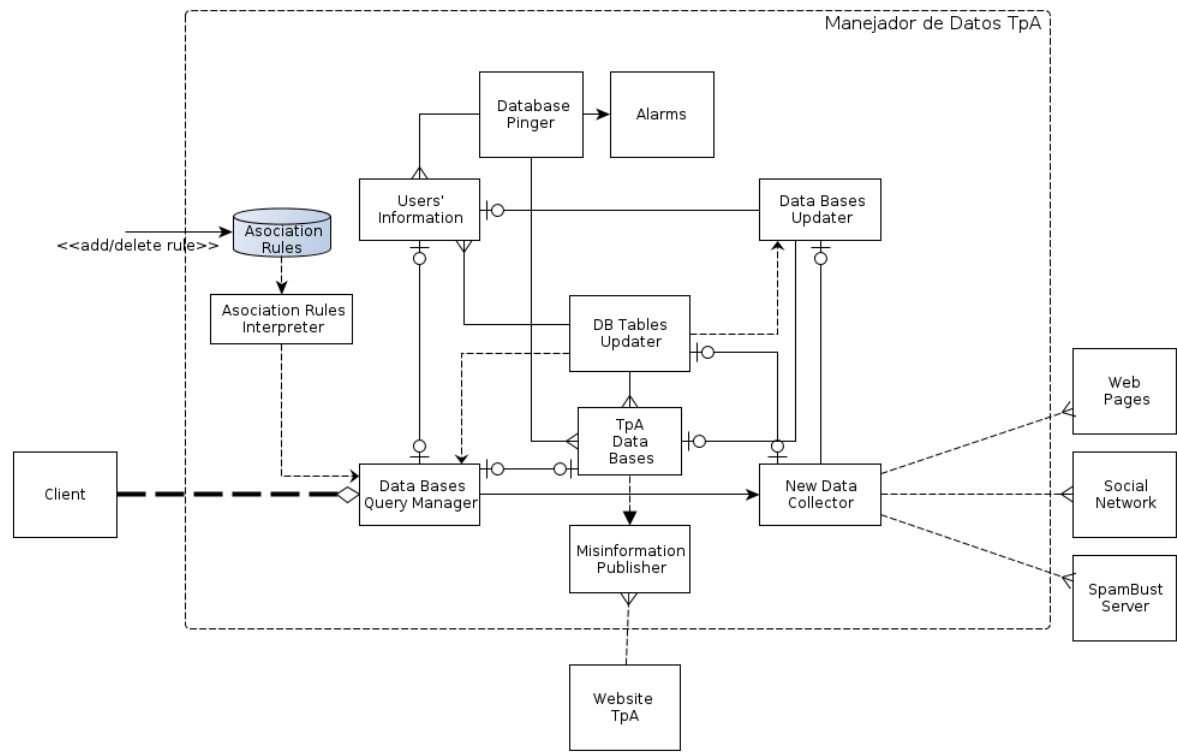
Como se puede ver, tenemos dos componentes principales:

- La aplicación que se instalará en los dispositivos móviles y permitirá que los usuarios del sistema puedan realizar sus consultas.
- El manejador de los datos del sistema, que se encargará de recoger la información y almacenarla, y responder a todos los pedidos de los usuarios.

Además, tenemos un conector que también tiene gran importancia, ya que representa la comunicación de muchos dispositivos de distinta índole con nuestro manejador de datos. Por lo tanto, también deberá ser analizado en profundidad.

8.1. Manejador de datos de TpA

Veamos primero a grandes rasgos qué es lo que hace el manejador de datos.



Tener en cuenta que los conectores de este gráfico son abstracciones de las muchas conexiones que se producen entre los distintos módulos

La idea es simple. Por un lado, tenemos clientes que se comunicarán con el Data Bases Query Manager. Luego, le realizará todas las consultas que considere necesarias y el Manager se encargará de obtener las respuestas de las distintas bases de datos. Estas están separadas en cuatro grandes grupos; por un lado tenemos los productos, las tendencias de búsqueda de los usuarios, las promociones (y dentro de las mismas las publicidades) y por último toda aquella información que se considere nociva. Una vez que el Manager obtiene la respuesta de cada consulta realizada a las distintas bases, le envía al cliente la información obtenida.

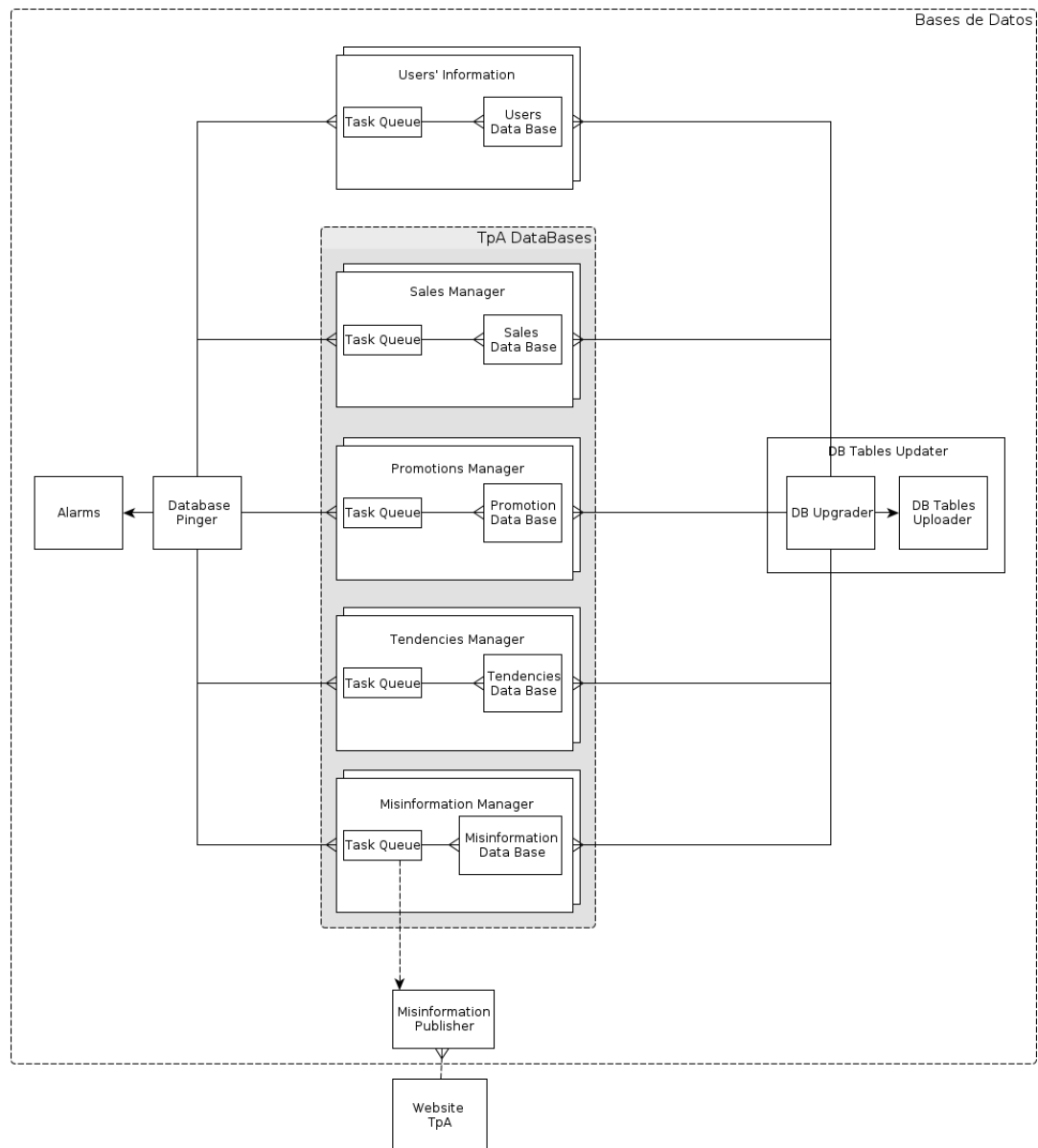
Para cargar la información a las distintas bases de datos contamos con un módulo llamado New Data Collector, que se encarga de recabar la información de las distintas fuentes, ya sea tanto nueva información de ventas o promociones, como nueva información acerca de las posibles estafas y engaños que se estén dando. Una vez que recibe información de este tipo, el Collector se comunica con el Data Bases Updater, quien se encargará de efectivamente persistir la información en las tres de las cuatro bases de datos antes mencionadas (la de tendencias se actualiza cuando se completa una búsqueda).

Al final de la sección se incluye un diagrama de componentes y conectores donde se ven en mayor detalle las conexiones entre los distintos módulos que componen al manejador de datos.

8.1.1. Manejo de las Bases de Datos

Decidimos dividir a las bases de datos en cuatro para mejorar la performance del sistema. Consideramos que tener por separado cada uno de los cuatro grandes grupos de posibles queries, nos permitiría paralelizar de manera más efectiva para así lograr programación concurrente.

Cada una de las bases de datos cuenta con una Queue que funcionará de la siguiente manera. Cuando le llega un pedido por pipeline, el mismo es un paquete que contiene el identificador del OQR³ que hizo el pedido y la consulta en sí. Luego, esta Queue se encargará de realizar la consulta en el repositorio que tiene asociado, y una vez que obtiene la información pedida, se la enviará a quien la pidió. Esto nos permite tener una asociación de muchos a muchos con los distintos OQRs y al mismo tiempo poder responderle a cada uno lo que pidió.



Las bases de datos de TpA y las bases de datos de los usuarios a grandes rasgos.

³Outsider Query Responder, más adelante se habla de la función del mismo

Como queremos tener bases de datos distribuidas, necesitamos tener un componente encargado del mantenimiento de las mismas. Luego de investigar un poco sobre el tema, decidimos utilizar una metodología similar a la que usan en Facebook Inc. El mismo consiste en tener una buena función de hash y una forma de asignar esos hashes por rango a cada base de datos, por ejemplo, en la *BD1* van del 0 al 10; en la *BD2*, del 21 al 31; etc. De esta forma, cuando llega un elemento nuevo, calculamos el hash y la base de dato en la que debe estar. A la hora de agrandar la base de datos, se divide el rango de datos que va a ir a cada base, y se copian desde la original aquellos que deban copiarse, luego se eliminan y ahora los rangos de cada hash estarán asociados de distinta manera.

Por ejemplo, si a *BD1* van de 0 a 3 y quiero duplicar la capacidad, ahora a *BD2*, van los hashes 1 y 3, como *BD1* tenía información del hash 1, la misma es copiada a la *BD2* y eliminada de *BD1*; una vez que se hace esto, la función que mapea cada hash a cada base de datos, se modifica. El DB Tables Updater es el encargado de realizar toda esta funcionalidad. El mismo consiste del DB Upgrader, a quien le van llegando las órdenes de agrandar las distintas bases y realizar los cambios en las funciones; luego, se encarga de hacer efectivos los cambios, copiar y eliminar la información de las bases de datos, etc. Para terminar, le avisa al DB Tables Uploader de todos los cambios, y este se encarga de hacer un broadcast para avisarle de los mismos a todos los módulos que utilizan los hashes.

El Misinformation Publisher es un módulo que se encarga de leer la información de la última semana de la base de datos de engaños, estafas, etc, y los envía al WebSite de TpA cada vez que este se los pide.

Además, contamos con un Pinger que nos indica si las bases de datos no están caídas, y en caso contrario, se encarga de hacer saltar una alarma para que quien esté encargado, revierta la situación.

8.1.2. User's Information

La información de los usuarios también tiene sus propias bases de datos, separadas del resto. Aunque también son muy importantes y pueden tomar interesantes proporciones, decidimos que no tenían tanto que ver con las otras bases de datos, y por lo tanto las distanciamos de las mismas en la arquitectura, considerándolas una estructura separada. Asimismo, almacenarán sólo los datos de las distintas entidades en las que confían a la hora de recibir mejores promociones, y el estado de cada usuario (ie, si está bloqueado o no).

8.1.3. Asociation Rules

Como las reglas de asociación de productos van a ser muy cambiantes, decidimos tener módulos completamente separados para esto. De esta forma, tenemos un repositorio que sirve como archivo de configuración, el cual alguien (externo al sistema) estará encargado de modificar cada vez que sea necesario. Una vez que se realizan los cambios, otro módulo se encarga de leerlos y enviárselos a cualquiera que los necesite.

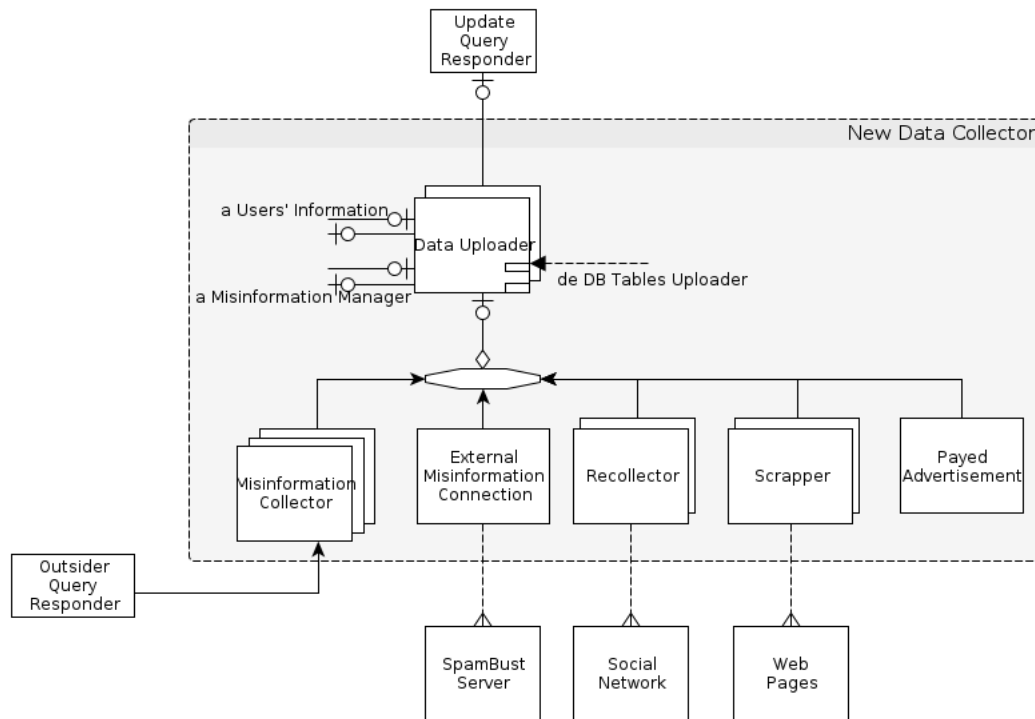
8.2. Recolección de la información

Tal como ya dijimos, para recolectar la información de las distintas fuentes, contamos con un módulo llamado New Data Collector. Veamos cómo funciona este en profundidad. Se tienen cinco tipos de recolectores distintos:

- **Misinformation Collector:** Recoge la información que dan los distintos usuario acerca de los precio que corroboran. Pueden ser tanto calificaciones negativas tomadas al azar, como denuncias.
- **External Misinformation Collector:** Este módulo toma la información de estafas y engaños desde un módulo externo. De momento, la obtendrá de Spambust, un sistema

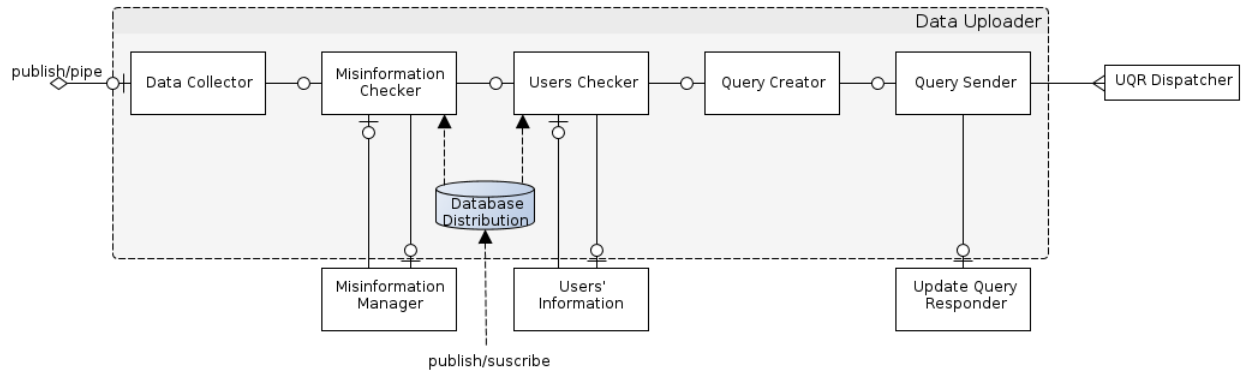
externo a nuestro desarrollo, estableciendo la comunicación con Spambust Server; aunque la idea es desarrollar a futuro otro módulo propio que se encargue de detectar las ofertas falsas.

- **Recolector:** Recoge información de la web de las distintas ofertas presentes en diversas redes sociales.
- **Payed Advertisement:** Sirve para que alguien especializado pueda subir las publicidades pagas que vamos a promocionar.
- **Scraper:** Vamos a tener también algunos módulos haciendo web scraping con la idea de obtener información



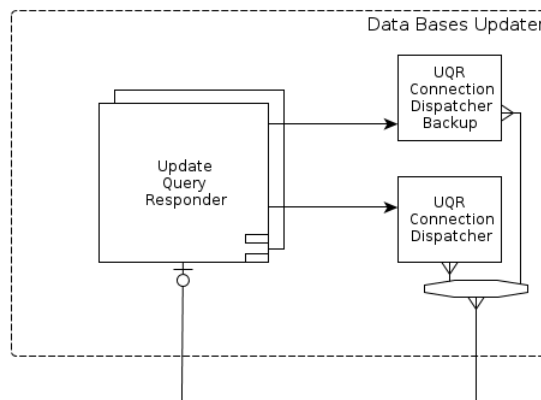
Todos estos módulos mandan su información a un publish/suscribe modificado (llamado publish/pipe), que es leído por algún Data Uploader. La modificación del publish/suscribe consiste en que cada información es leída por sólo uno de todos los receptores. El Data Uploader se encarga de verificar que la información no es espuria según los datos obtenidos anteriormente y, en caso de que sea buena, mandarla al Data Bases Updater, el encargado de persistir la información en la BD. La idea de realizar la arquitectura de esta manera se basa en que al tener los módulos por separado, si el día de mañana queremos modificar la forma en que recibimos cierto tipo de información, por ejemplo a los recolectores, basta con sólo modificarlos a estos dejando al resto inalterado. Asimismo, en caso de que alguno de los elementos dejara de funcionar, como tienen múltiples instancias, tampoco habrá grandes problemas.

El Data Uploader es un simple pipe & filter que se encarga de lo siguiente. Recibe varios pedidos desde alguno de los recolectores; primero chequea que la información no haya sido dada de baja como un engaño; luego, mira que la información de cada pedido no provenga de un usuario que actualmente esté bloqueado; después se crean las queries correspondientes y se envían al Data Bases Updater.



8.2.1. Interacción con las bases de datos

Tanto el Data Base Updater como el Data Base Query Manager funcionan de la misma manera, así que vamos a explicarlos una vez. La idea consiste en que vamos a tener muchos Responders de pedidos, lo que nos permite atender pedidos en simultáneo. El problema se da a la hora de dirigir el tráfico de manera eficiente. Para realizar esto, decidimos incluir un módulo llamado Dispatcher, que posee una lista de los Responders que están libres, y se encarga de atender cada pedido de conexión nueva devolviendo un elemento de esta lista.

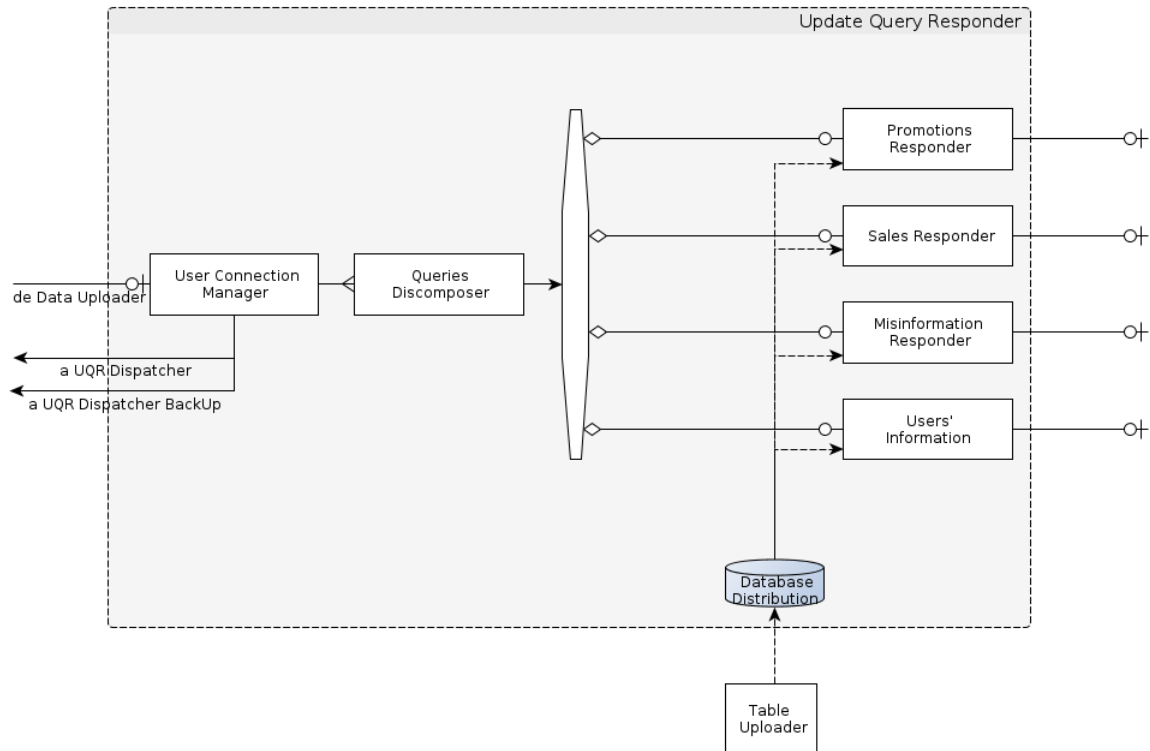


Luego, el Responder se encarga de resolver la consulta, ya sea guardando información, como modificando, leyendo o borrando. Una vez hecho esto, contesta el pedido y además le avisa a Dispatcher que ahora estará disponible. De esta manera, cuando venga un nuevo pedido, se podrá derivar al Responder.

Para evitar problemas con la posible caída del Dispatcher, decidimos duplicarlo, de forma tal que el Backup va leyendo todos los mensajes que manda el original, de forma tal que va poniendo en su lista cuáles son los Responder utilizados. Además, los Responders también le mandan a él que ya están libres. Si llega un pedido y el Dispatcher no le asigna un Responder en 5 segundos, el Backup lo considera caído, hace saltar una alarma y responde. Esto nos ayuda a mantener la disponibilidad del sistema, al tiempo que se evitan grandes problemas de concurrencia. Si bien el Dispatcher es un cuello de botella, al hacer cosas tan sencillas, podemos suponer que va a responder siempre con mucha rapidez. Debido a la cantidad de información que el sistema maneja en sus bases de datos y que el tipo de información no es crítica, no parece una buena decisión implementar sistema de redundancia de dichas bases.

8.3. Update Query Responder

El funcionamiento del Update Query Responder es por demás sencillo, ya que se encarga de subir a las bases de datos, la información que le fueron dando los distintos recolectores. Por lo tanto, cuando le llegan pedidos, tan sólo debe descomponerlos en cada una de las queries reales e ir tirándolas en un publish/suscribe. Luego, dependiendo de qué tipo sea cada una, la van a ir leyendo los distintos componentes e interactuando con las bases de datos pertinentes.



Por ejemplo, llegan dos pedidos, los cuales dicen lo siguiente:

1. Eliminar Usuario “Ramon14”.
2. Agregar “Papa \$5 kilo Lavalle 15”.

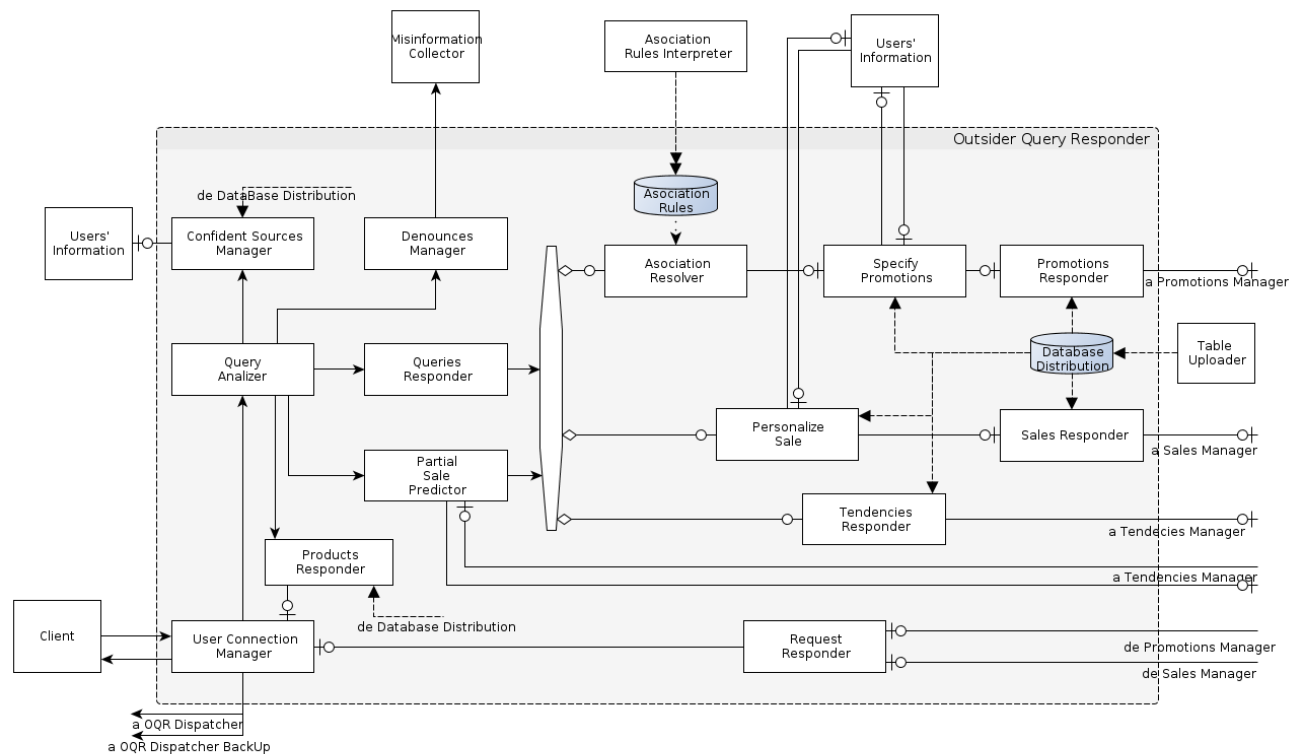
Entonces, el primero de los pedidos interactuará con las cuatro bases de datos; eliminando los datos de Ramon14 de las bases de datos de Información de Ventas y de Promociones, baneándolo temporalmente como usuario en la Información de Usuarios, y agregando lo ocurrido al módulo de Engaños.

El segundo pedido, por otra parte, simplemente agregará a la Información de Ventas, el hecho de que se consigue papa a \$5 el kilo en Lavalle 15.

Una vez que el Queries Discomposer manda todas las queries a ser realizadas, responde al User Connection Manager que ya terminó. Si no queda más por hacer, este envía una señal a cada Dispatcher para avisarles que ya está libre.

8.4. Outsider Query Responder

Este módulo es bastante más complicado, así que veamos primero la vista de componentes y conectores para luego ir analizándolo.



Primero el User Connection Manager recibe un pedido, que manda al Query Analyzer para saber qué debe hacer. De ahí, puede ser enviado a uno de cuatro módulos:

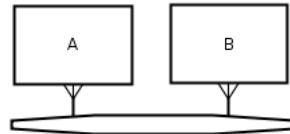
1. **Confident Sources Manager:** Se encarga de modificar las opciones de la confianza que tiene el usuario en distintas fuentes.
2. **Denounce Manager:** Se encarga de mandar las denuncias realizadas por los usuarios.
3. **Partial Sale Predictor:** El cliente manda consultas no completadas para realizar una búsqueda en base a las tendencias de las búsquedas pasadas.
4. **Queries Responder:** Es el encargado de resolver las consultas terminadas del usuario.
5. **Products Responder:** Cada vez que un cliente se conecta, va a preguntar por los productos sobre los que tenemos información, este componente se encarga de responder rápidamente cuáles son, leyendo de la distribución de las bases de datos.

El Queries Responder se encarga de mandar al publish/suscribe el pedido del usuario. Luego, los tres suscriptores leen el pedido. Para buscar las promociones, primero se deben resolver las reglas de asociación en base al producto; luego, hacer una búsqueda más específica en base a los datos del usuario, y por último mandar la query al Promotion Manager. Para hacer la búsqueda de los productos, se personaliza la búsqueda en base a resultados que le habían resultado convenientes al usuario otrora, la información que hayamos recabado del mismo, y demás; y luego se manda la query al Sales Manager. Al mismo tiempo, se guarda en el Tendencias Manager lo buscado, ya que las búsquedas totales pasan a formar parte de las tendencias⁴.

⁴No tiene sentido que las búsquedas parciales se tomen en cuenta para las tendencias, ya que daría lugar a inconsistencias. Por ejemplo, si el “tomate perita” es el más buscado en un momento, cada vez que busco “tomate”, la búsqueda parcial irá al perita; si la tendencia cambiara y la gente pasara a buscar “tomate redondo”, igualmente cada vez que escriban “tomate” se

Luego, los resultados de las búsquedas pedidas, son enviadas hacia el Request Responder, que enviará las respuestas al usuario, mediante el Connection Manager, aunque la búsqueda no terminó de completarse. Por ejemplo, puede resolverse la consulta de productos aunque todavía no se hayan obtenido las promociones. Del lado del cliente se presentará dicha información constantemente mientras llegue desde el servidor.

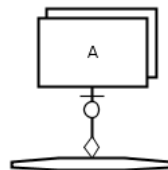
8.5. Conectores



Este publish/suscribe nos permite la asignación de Responders por parte de los dos Dispatchers. La idea principal es que uno de los Dispatchers es el primario mientras que el otro funciona como back up en caso de caída del primero; vamos a llamarlos primario y secundario respectivamente. Por lo tanto, si bien ambos van a estar escuchando los pedidos, la respuesta será realizada por el primario. En caso de que pasaran 5 segundos desde el pedido, y el primario no respondiera, el secundario tomará la posta, asignando un Dispatcher al pedido y enviando una señal a la alarma para notificar el evento.

Cada pedido se responderá de forma incremental y de manera consecutiva; por lo tanto, nunca podrá responderse un pedido hasta que no se hayan respondido todos los anteriores. Al mismo tiempo, cada asignación de Responder enviada al publish/suscribe funcionará como una interrupción, evitando que el otro Dispatcher siga trabajando en la misma o en cualquiera de las anteriores.

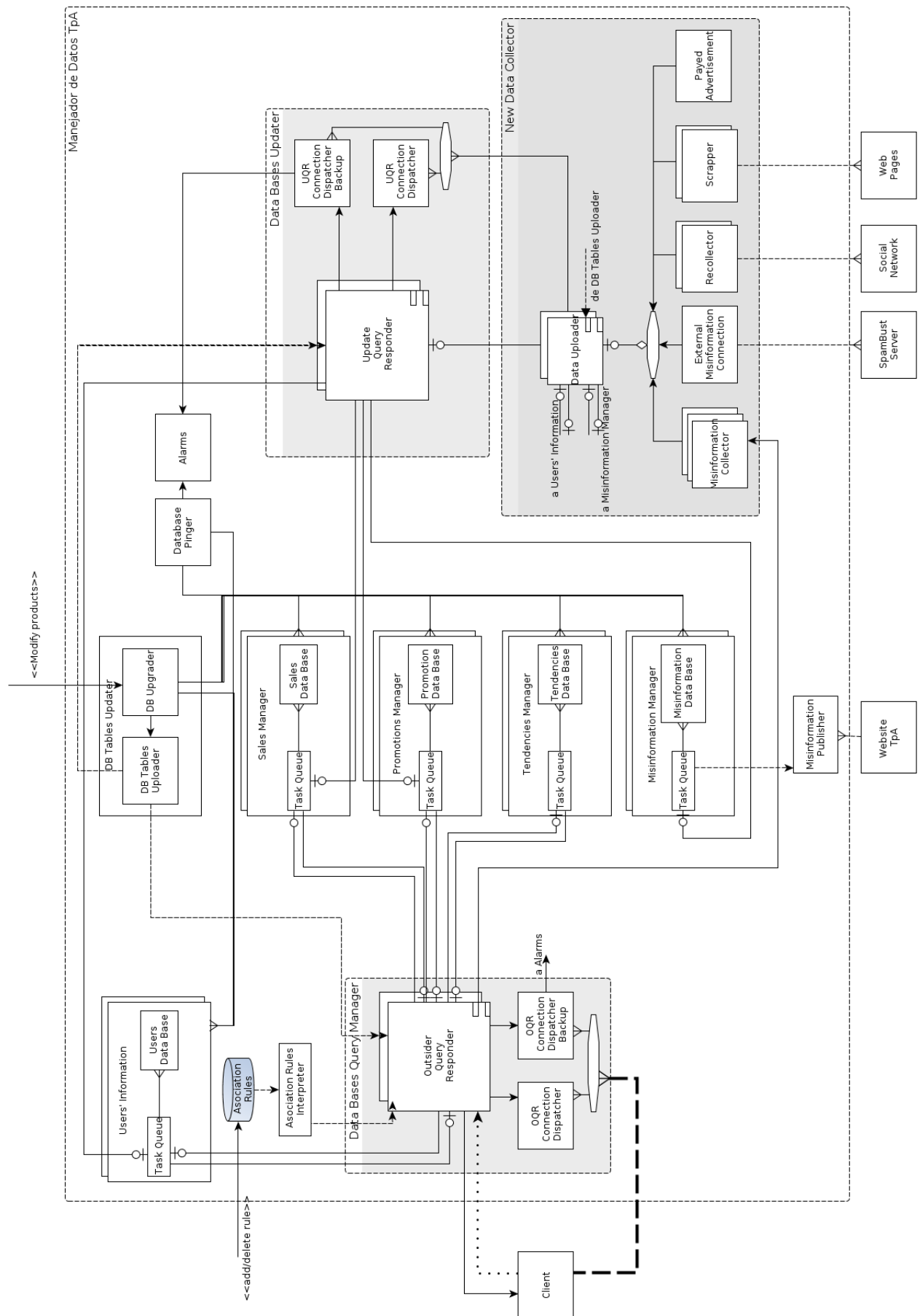
Por ejemplo, llega el pedido 12005 y el primario tarda más de diez segundos en responder ya que se bloquea por cierto motivo; luego, el secundario asigna un Responder al pedido 12005 y también a los pedidos 12006 al 12014. En ese momento, el primario se desbloquea, pero tiene que atender a las interrupciones, y una vez que termina, descarta la asignación que estaba realizando para el 12005 y se pone a esperar al siguiente pedido.



Este publish/pipe va de muchos publicadores a muchos suscriptores, con la particularidad de que todos los suscriptores pueden atender lo que se publica. Sin embargo, no lo harán; la idea es que sólo uno de los suscriptores atienda cada pedido. Se trata de una cola de pedidos, los cuales pueden ser atendidos por uno de muchos suscriptores.

aumentará la cantidad de veces buscada al perita, por más que esto no fuera cierto.

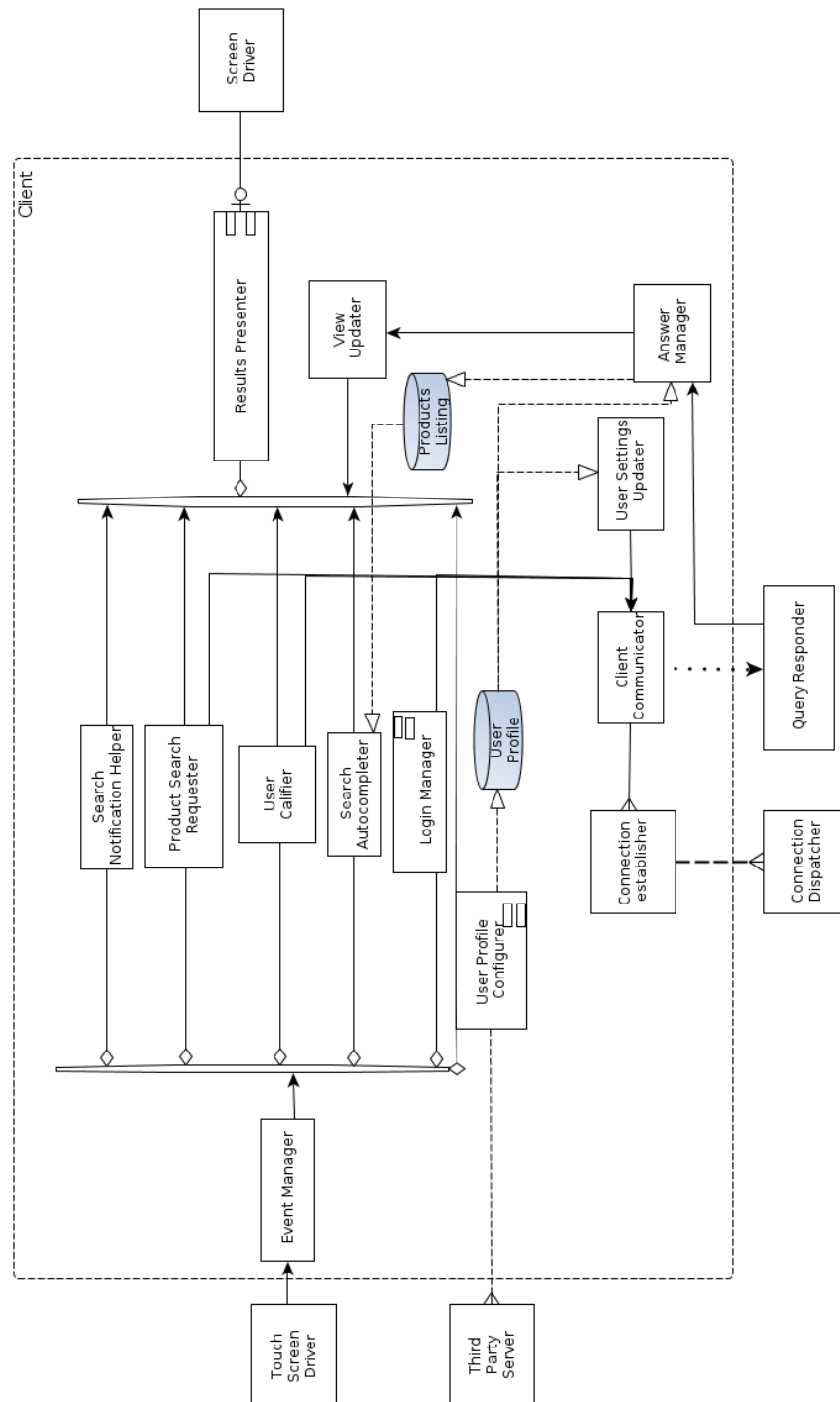
8.6. Diagrama completo arquitectura servidor



8.7. Justificación arquitectura del cliente

La arquitectura del cliente es bastante simple, por un lado, se tiene que tener varios componentes asociados a eventos del touch screen mediante un manejador de eventos, que se adapta a la plataforma. Dichos eventos se relacionan con distintas funcionalidades que el usuario desea utilizar del sistema. Luego, cada componente que responde a una funcionalidad particular tiene como responsabilidad realizar cambios en la vista del usuario y actualizar la misma. Aquellos componentes que necesiten de obtener información del servidor, se comunicarán con el mismo utilizando un componente encargado de la conexión.

Una primera vista general del cliente sería:



Las cinco operaciones principales, que necesitan del servidor, de un cliente son:

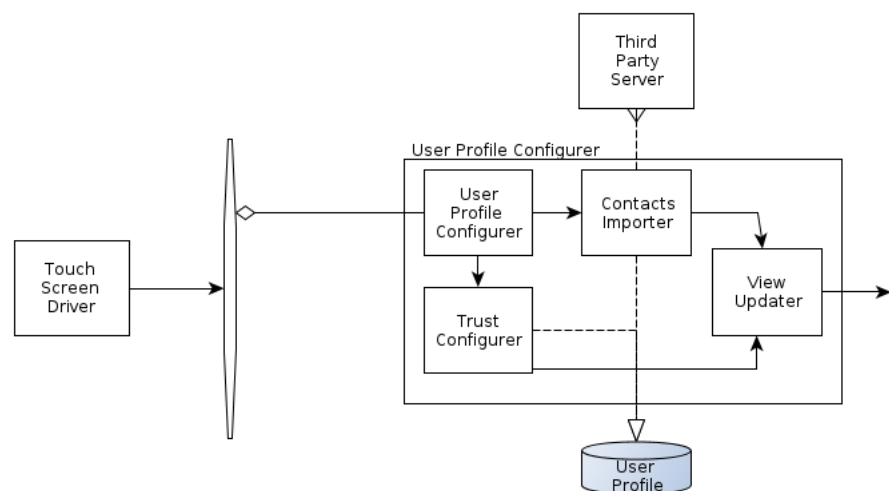
- Autenticación
- Pedido de listado de productos del sistema
- Consulta de un producto (podría ser una consulta no completada)
- Denuncia de información de venta
- Envío de actualización de profile de usuario

A estas funcionalidades, se le agregan aquellas funcionalidades que no necesitan de la comunicación con el servidor de TpA. Las mismas son la configuración del profile del usuario, las notificaciones de ayuda para los elementos de pantalla del cliente y, por último, el sistema de autocompletado de los productos para la búsqueda.

Inicialmente, las respuestas del servidor eran dirigidas hacia esos componentes que originaron el pedido para que pudieran actualizar sus vistas de acuerdo a esos datos. Esto se cambio debido a que llevaba a problemas de performance, pues siempre se tenía que esperar respuesta del servidor para completar la funcionalidad, conllevando también problemas de usabilidad pues no se actualizaría la vista hasta obtener información para mostrar. Esto va en contra de la usabilidad por que un usuario desea tener un feedback inmediato al interactuar con el sistema. Posteriormente, se decidió que cada componente seguiría actualizando la vista del usuario para tener ese feedback que aumenta la usabilidad y por otro lado las respuestas del servidor serían manejadas por otro componente del cliente exclusivamente encargado de actualizar la vista en casos que haga falta. Ocurre que cuando se pide el listado de los productos del sistema, no es necesario de actualizar la vista pues esto se hace de forma transparente al usuario.

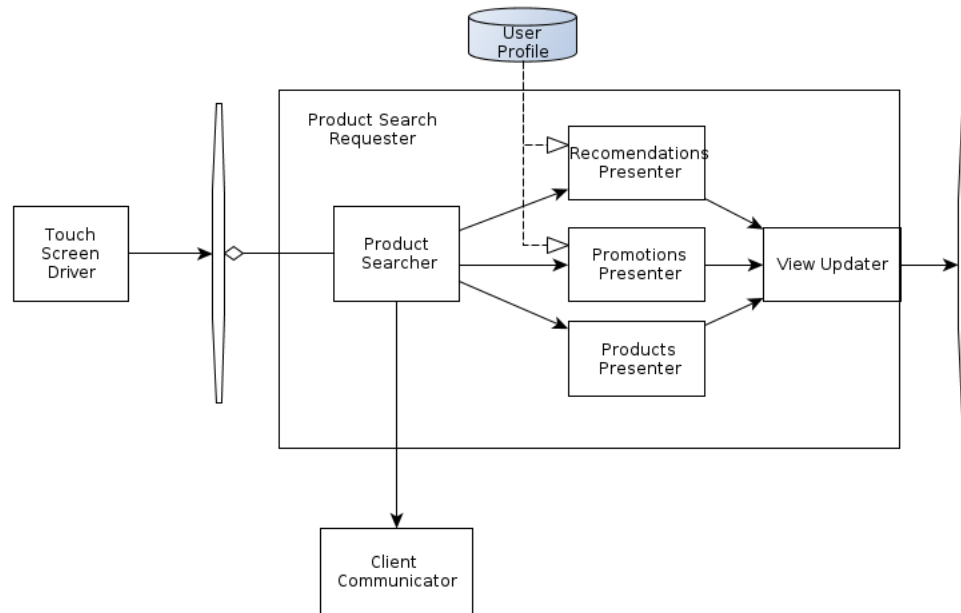
Por otro lado, la configuración de las preferencias de búsqueda del cliente (como por ejemplo de confianza, de querer no ver información de contactos de LinkedIn) se lleva a cabo en el cliente pero la misma es enviada al servidor para que las consultas puedan ser personalizadas según los datos de cada uno. Esta decisión se tomó para disminuir la cantidad de ataques al servidor en busca de claves de las redes sociales de los usuarios, utilizadas para importar los contactos del usuario. Tener dichas claves almacenadas en los clientes disminuiría ese tipo de ataques hacia el servidor.

Sistema de configuración de profile de usuario:



La denuncia de alguna información presentada como resultado se realiza también en el cliente y la misma es enviada al servidor para que se la tenga en cuenta en su análisis de información falsa.

El Presentador de resultados sería responsable de la correcta visualización de los cambios realizados en pantalla por los componentes.



Los atributos de calidad impuestos para el sistema deben ser reflejados en la arquitectura del cliente, así los principales atributos a cumplir en este subsistema son los de usabilidad y performance. Los escenarios de atributos de calidad de usabilidad pueden ser satisfechos agregando componentes independientes y paralelos encargados de la funcionalidad que ayudaría al usuario a aprender a utilizar el sistema o a un uso mejor del mismo como es el caso de las notificaciones de ayuda o el autocompletado de productos. Por otro lado, para mostrar promociones y recomendaciones es necesario tener en cuenta el profile del usuario para tener en cuenta su antigüedad y limitar la aparición de información que sobrecargue la visualización de los resultados.

Por otro lado, para satisfacer escenarios de performance, se valió de aumentar la concurrencia de los componentes del cliente. Teniendo un componente especializado para el pedido de resultados de búsqueda, el cual por cada producto que se ingresa se ocupa de consultar al servidor para obtener resultados del mismo.

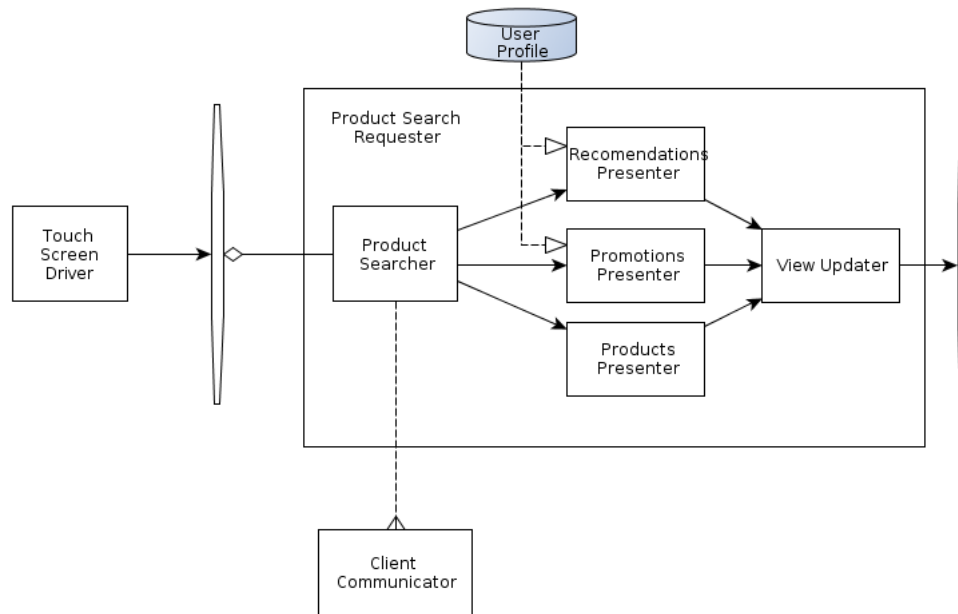
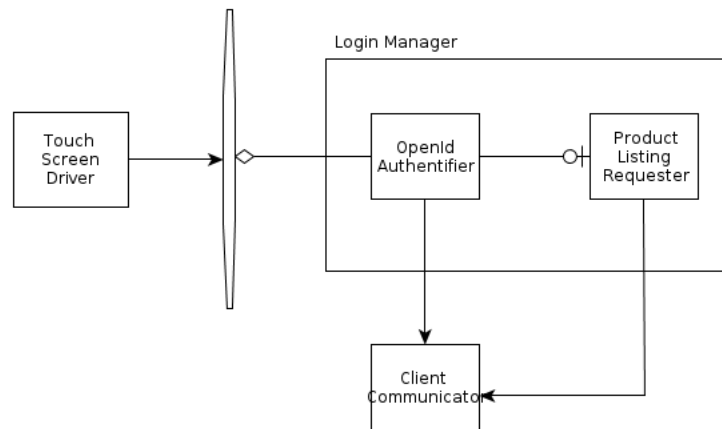


Figura 1: Componente encargado de la búsqueda de productos

Respecto a otros atributos que tendrían que ser tenidos en cuenta del lado del cliente, la extensibilidad de rubros se logra mediante la actualización de rubros de productos que se tiene en el cliente durante el proceso de login del usuario. Al mismo tiempo, la portabilidad de la plataforma del cliente se logra mediante la encapsulación de la visualización de los datos y del manejo de los eventos disparados por el usuario, por lo que para lograr portabilidad de la plataforma se cambia el componente de presentador de resultados por otro adecuado para la plataforma objetivo y el manejador de eventos por uno correspondiente a la otra plataforma.

La confiabilidad de los datos se puede llevar a cabo utilizando un mecanismo de encriptación en el comunicador con el servidor para enviar los datos de login del usuario mediante OpenId al servidor. Al principio la autenticación del usuario mediante OpenId estaba siendo realizada en el componente del cliente pero se decidió que el mecanismo debería ser realizado por el servidor para aumentar la seguridad de la aplicación al permitir la actualización del componente de login de forma invisible para el usuario. El problema con este enfoque se encuentra en que es necesario permitir un canal seguro y confidencial entre el cliente y el servidor sólo para fines de autenticación. Se decidió que el resto de la comunicación entre el servidor y el cliente no se encripte para favorecer el rendimiento del sistema.

Así el sistema de login es el siguiente:

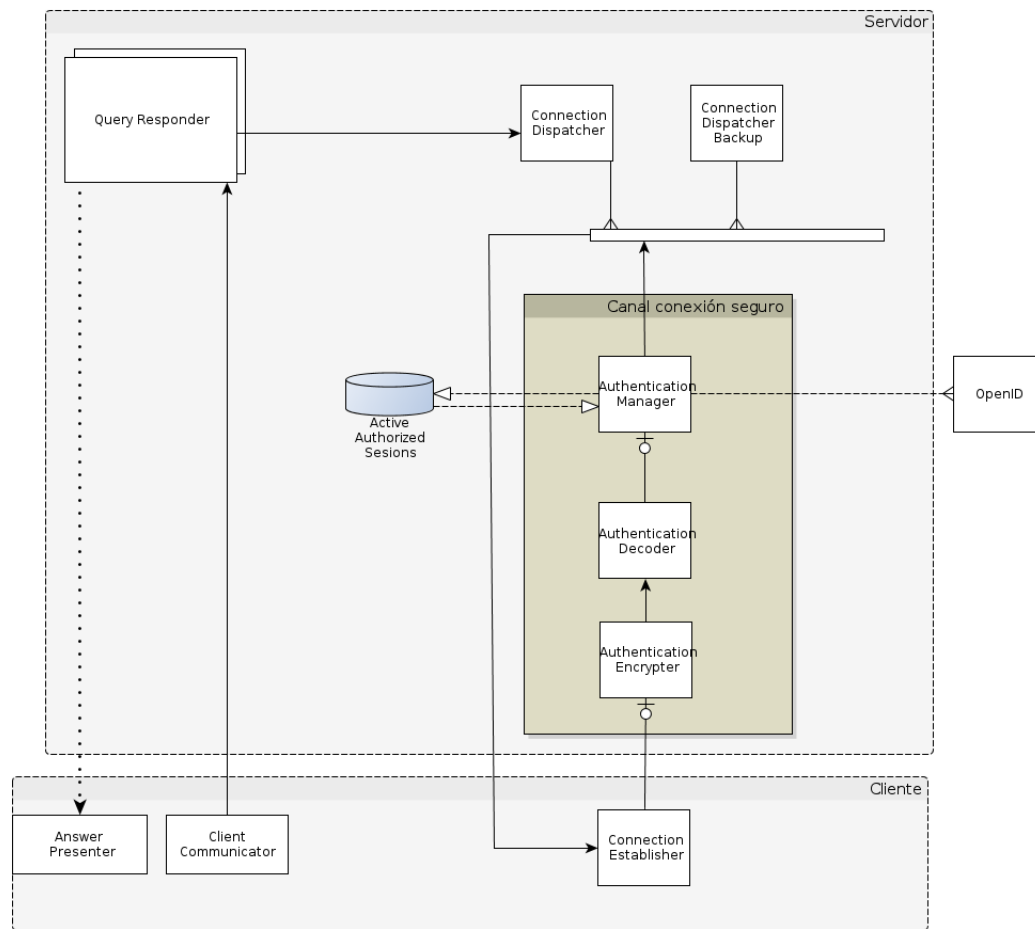


Finalmente, el motor de reglas de asociación no se encuentra del lado del cliente porque, si bien favorecía la performance, al modificar las reglas todos los clientes deberían actualizar sus reglas. De la otra forma, al tenerlo en el servidor, la actualización de las reglas de asociación se realiza de forma invisible e inmediata para las nuevas consultas.

8.8. Conexión entre servidor y cliente

Hasta el momento, se evitó detallar demasiado acerca de la comunicación entre los clientes y el servidor del sistema. Esto se hizo para evitar complicar demasiado ambas arquitecturas previamente descritas. Pero es necesario ahondar en este tema si se quiere terminar con la arquitectura. Lo primero a decir acerca de la comunicación es que siempre que el cliente desee interactuar con el servidor, lo hará mediante los Query Responder ubicados en el servidor. El Query Responder que se encarga de responder una interacción de usuario es asignado por el Connection Dispatcher.

La arquitectura de la comunicación sería la siguiente:



Ahora, previo a cualquier otro tipo de pedido del usuario al servidor debe ser necesario la autenticación del mismo. Para esto, el usuario envía mediante un canal de conexión seguro los datos necesarios para asegurar su autenticación. Para esto se encuentra el canal de conexión segura, que si bien encripta aquellos datos utilizados para la autenticación, no lo hace para el restantes tipos de pedidos (consultas, calificaciones, etc), esto se decidió para favorecer la performance al momento de realizar una consulta. Notar que estos pedidos del Connection Establisher tratan de obtener un Query Responder y no tienen ningún tipo de información crítica. Luego de descifrar los datos de autenticación el servidor realiza una consulta al servidor de OpenId para realizar la propia autenticación y logea al usuario como sesión activa.

Para corroborar cuales usuarios ya se autenticaron, por lo que no es necesario realizar una nueva consulta a OpenId, se consulta si se trata ya de una sesión activa. Por lo tanto, aquellos pedidos de usuarios ya utilizados de respondedores disponibles no requieren de una nueva autenticación o de la encriptación de datos.

Una vez que se tiene el Query Responder al que se le va a consultar, se envía los distintos tipos de consulta al servidor para que el mismo la responda. El único tipo de comunicación que se encripta es cuando se envía el profile del usuario al servidor. Nuevamente, para favorecer la performance de las consultas, se decidió no encriptar el restante tipo de comunicaciones.

9. Atributos de calidad

9.1. Escenarios de atributos de calidad

Los escenarios de calidad elicitados durante el QAW fueron los siguientes:

- Usabilidad
- Rendimiento
- Integrabilidad / extensibilidad
- Disponibilidad
- Auditabilidad
- Seguridad

En ese orden de prioridad entre los atributos. A continuación se realiza un listado de los escenarios relevantes de cada atributo y la táctica utilizada en la arquitectura para poder satisfacerlo:

9.1.1. Usabilidad

Escenario 1:

- **Descripción:** Autocompletado de búsqueda en menos de 1 segundo
- **Fuente:** Usuario
- **Estimulo:** Escribe parcialmente un producto en el buscador
- **Artefacto:** Search autocompleter
- **Entorno:** En tiempo de corrida
- **Respuesta:** El módulo de cliente muestra por pantalla un autocompletado de los productos posibles a partir de ese ingresado
- **Medida:** La notificación aparece dentro del segundo desde que el usuario empezó a escribir el producto

Para este escenario seguimos una táctica de Support User Initiative. Para esto utilizamos el componente Search Autocompleter dentro del cliente que le facilita al usuario el ingreso de sus búsquedas a partir de los productos que se tienen en el servidor. Suponemos que el componente descripto cumple los límites temporales especificados.

Escenario 2:

- **Descripción:** Notificar funcionalidad en menos de 1 segundo
- **Fuente:** Usuario
- **Estimulo:** Consulta acerca de información de elemento en pantalla
- **Artefacto:** Results presenter
- **Entorno:** En tiempo de corrida
- **Respuesta:** El módulo cliente muestra una pequeña notificación acerca de la funcionalidad del elemento consultado
- **Medida:** La notificación aparece dentro del segundo desde que el usuario realizó la consulta

Nuevamente utilizamos la misma táctica, brindándole al usuario la posibilidad de conocer las distintas funcionalidades a las que puede acceder. Suponemos que este proceso se puede realizar dentro de los límites impuestos.

Escenario 3:

- **Descripción:** Ofrecer publicidades limitadas a usuarios nuevos
- **Fuente:** Usuario
- **Estimulo:** Sentirse comodo con la UI
- **Artefacto:** Promotions presenter
- **Entorno:** En tiempo de corrida
- **Respuesta:** El sistema presenta una cantidad de publicidades limitada a los nuevos usuarios para no impedir su proceso de aprendizaje
- **Medida:** El 90 % de los primeros usuarios de prueba consultados muestra un nivel satisfactorio del uso del sistema al evaluarlo

En este caso utilizamos la táctica de Support System Initiative, en donde usamos un modelo del cliente para concer su antigüedad en el sistema. Este modelo se corresponde con el Profile de usuario, y como se puede ver en la búsqueda de productos, los presentadores de resultados toman en cuenta dicho profile.

9.1.2. Rendimiento

Escenario 4:

- **Descripción:** Almacenar nuevas promociones en menos de 1 segundo
- **Fuente:** Recolector de red social
- **Estimulo:** Encuentra nueva información de venta de un producto
- **Artefacto:** Módulo servidor
- **Entorno:** Puesto en funcionamiento
- **Respuesta:** La nueva información de venta es almacenada en el servidor y puede ser vista por nuevas consultas
- **Medida:** El almacenamiento de nuevas promociones se realiza en menos de 1 segundo desde que fue encontrada

Para este escenario aplicamos una táctica de Resource management introduciendo concurrencia a los componentes Recolector que obtienen los datos nuevos y a los Update Query Responder que son los que los almacenan. Asumimos que con esto logramos los tiempos expuestos.

Escenario 5:

- **Descripción:** Enviar resultados de búsquedas parciales en menos de 10 ms
- **Fuente:** Usuario
- **Estimulo:** Ingresa nueva búsqueda parcial de una cantidad de productos
- **Artefacto:** Sistema
- **Entorno:** Normal
- **Respuesta:** Dada la búsqueda parcial, el servidor devuelve resultados para la búsqueda
- **Medida:** La búsqueda se realiza en menos de 10 ms

Nuevamente seguimos la misma táctica implementando concurrencia sobre los Outsider Query Responder que atienden a cada uno de los pedidos. Volvemos a asumir que resulta en que puede cumplir las restricciones temporales explicitadas.

Escenario 6:

- **Descripción:** Predecir resultados de búsqueda al cliente en menos de medio segundo
- **Fuente:** Query Analyzer
- **Estimulo:** Predice resultados de búsqueda de cliente
- **Artefacto:** Partial Sale Predictor
- **Entorno:** Normal
- **Respuesta:** El Query Analyzer envía a Partial Sale Predictor una búsqueda parcial, y este se encarga de resolver la predicción para enviarla al usuario
- **Medida:** El usuario puede visualizar la predicción dentro del medio segundo

Para aumentar la performance, en este caso decidimos utilizar un mecanismo para predecir resultados de búsquedas de los clientes, en vez de esperar a que las ingresen en su totalidad. Continuamos asumiendo que los componentes utilizados me permiten cumplir la medida elegida.

9.1.3. Extensibilidad

Escenario 7:

- **Descripción:** Cambiar reglas de asociación entre productos sin modificar el código del sistema en menos de 1 hr
- **Fuente:** Administrador de negocio
- **Estimulo:** Decide cambiar reglas de asociación de promociones
- **Artefacto:** Association Rules
- **Entorno:** En tiempo de corrida
- **Respuesta:** Las reglas de asociación entre productos es cambiada y el sistema comienza a utilizar dichas reglas para las nuevas consultas
- **Medida:** El cambio se realiza sin realizar modificaciones en el código del sistema en menos de 1 hr

Para poder lograr que se cambien las reglas de asociación sin tener que realizar modificaciones en el código, utilizamos una táctica de Defer Binding Time mediante un archivo de configuración que provee una forma de acceder directamente al conjunto de reglas para que alguien especializado las modifique. También contamos con un módulo intérprete, que sabe manejar dicho archivo. De esta forma, nos estamos abstrayendo de las reglas en sí de manera que cambiarlas no impacte en el código del sistema.

Escenario 8:

- **Descripción:** Agregar nuevo rubro de productos en menos de un día sin modificar código del sistema
- **Fuente:** Administrador de negocio
- **Estimulo:** Decide agregar nuevo rubro de productos
- **Artefacto:** DB upgrader
- **Entorno:** En tiempo de corrida
- **Respuesta:** Se agrega el nuevo rubro de productos y a partir de ese punto los recolectores tendrán en cuenta información asociada a productos de ese rubro
- **Medida:** El cambio se puede lograr dentro de un día sin realizar modificaciones en el código del sistema

Esta posibilidad de extensión la conseguimos principalmente por anticiparnos a los posibles cambios tratando de que los mismos afecten la menor parte posible de todo el sistema. Para ello acumulamos los rubros en una especie de lista denominada Database Distribution. Es decir utilizamos una táctica de Localize Changes, en particular anticipate expected changes.

Escenario 9:

- **Descripción:** Poder modificar el cliente para una nueva plataforma
- **Fuente:** Administrador de negocio
- **Estimulo:** Decide portar cliente a nueva plataforma
- **Artefacto:** Módulo cliente
- **Entorno:** En tiempo de corrida
- **Respuesta:** El equipo de desarrollo realiza una modificación testeable del cliente para lograr ser utilizado por la nueva plataforma especificada
- **Medida:** La portabilidad de la plataforma del cliente se logra con un esfuerzo de dos meses hombre

En este caso logramos la portabilidad de la plataforma gracias al encapsulamiento de la visualización de los datos. Se cambia el componente de presentador de resultados por otro adecuado para la plataforma objetivo, y se reemplaza el manejador de eventos por uno correspondiente a la plataforma indicada. Es claro entonces que la táctica utilizada es básicamente un Abstract Common Services.

9.1.4. Disponibilidad

Escenario 10:

- **Descripción:** Alertar a los administradores ante caídas de la caída de una base de datos
- **Fuente:** Sales Data Base
- **Estimulo:** Deja de responder consultas acerca de su estado
- **Artefacto:** Database Pinger
- **Entorno:** Sistema online pero con una base de datos caída
- **Respuesta:** Se notifica a los administradores sobre la caída de la base de datos
- **Medida:** Los administradores reciben la alerta en menos de 10 segundos de la caída

Para lograr este escenario utilizamos la táctica de detección de fallas implementando un componente encargado de verificar el estado de las base de datos y de notificar mediante alarmas a los administradores en caso de que alguna base se encuentre caída o degradada.

Escenario 11:

- **Descripción:** **Tolerancia a fallas** . Recuperar sistema ante caídas del Connection Dispatcher en menos de 30 segundos
- **Fuente:** Módulo servidor
- **Estimulo:** No obtiene respuesta del sistema
- **Artefacto:** Connection Dispatcher
- **Entorno:** Sistema con Connection Dispatcher caído
- **Respuesta:** El sistema detecta que el dispatcher está caído, notifica a las copias redundantes del mismo y recupera el servicio
- **Medida:** La recuperación del servicio se logra dentro de los 30 segundos

Para lograr esto utilizamos la táctica de Recovery, Preparation and Repair implementando redundancia en el Connection Dispatcher. Así cuando uno está fuera de funcionamiento se puede recuperar el sistema con la otra copia.

9.1.5. Auditabilidad

Escenario 12:

- **Descripción:** Almacenar información de ventas falsas del último mes
- **Fuente:** SpamBust
- **Estimulo:** Envía informe de información de venta falsa
- **Artefacto:** New data collector
- **Entorno:** Sistema online
- **Respuesta:** El sistema guarda una copia de la información de venta falsa previo a eliminarlas para un análisis posterior
- **Medida:** El log de información falsa debe tener toda la información eliminada del mes pasado

Para lograr la auditabilidad del sistema, mantenemos la información de las ventas reportadas como falsas que son eliminadas, para poder hacer luego un análisis sobre ellas.

Escenario 13:

- **Descripción:** Almacenar perfiles de fuentes de información
- **Fuente:** Recolector de red social
- **Estimulo:** Encuentra nueva información de venta de un producto
- **Artefacto:** Misinformation manager
- **Entorno:** Puesto en funcionamiento
- **Respuesta:** El sistema guarda perfiles de las fuentes de la información para evitar información de perfiles con alto nivel de información falsa
- **Medida:** El perfil de fuentes es actualizado semanalmente

En este caso mantenemos los perfiles de las fuentes de información para poder auditarlos y rechazar usuarios fraudulentos.

9.1.6. Seguridad

Escenario 14:

- **Descripción:** Detectar informes fraudulentos de promociones falsas
- **Fuente:** Atacante de identidad desconocida
- **Estimulo:** Envía informe de promociones falsas
- **Artefacto:** New data collector
- **Entorno:** Normal
- **Respuesta:** Sistema detecta que se trata de un informe no originado por SpamBust y lo descarta
- **Medida:** El 99,9 % de las veces se descarta el informe falso

Aquí seguimos una táctica de Detecting attacks e intrusion detection para descubrir cuando un informe de promociones falsas es originado por alguien ajeno al sistema y debe ser, por lo tanto, descartado.

Escenario 15:

- **Descripción:** **Confabilidad** . Evitar descriptación de los datos
- **Fuente:** Atacante no autenticado
- **Estimulo:** Captura mensaje con datos privados de usuario
- **Artefacto:** Authentication Encrypter
- **Entorno:** Sistema online
- **Respuesta:** Los datos del usuario se mantienen inaccesibles por al menos un año
- **Medida:** El 99,9 % de los casos los ataques no consiguen la información descriptada dentro de un periodo de un año

La táctica utilizada en este caso apunta a mantener la confidencialidad de los datos. Así, con un buen sistema de encriptación nos aseguramos que, aunque un atacante consiga capturar un mensaje con datos privados del usuario, estos sean inaccesibles por un largo período de tiempo.

10. Análisis

10.1. Programming in the large *vs* Programming in the small

Se denomina *programming in the large* al desarrollo de un sistema de alta complejidad por un gran grupo de personas durante un lapso largo de tiempo. Debido al esfuerzo que se debe afrontar en dichos proyectos, son críticas las tareas de planificación y documentación como actividades de soporte durante ese periodo de producción. En dichos proyectos, los procesos de la organización influyen fuertemente en el producto a construir, por eso se vuelven relevantes que sean formalmente definidos, revisados y modificados. También se vuelven más relevantes que en pequeños proyectos las tareas de gestión para que se implementen los procesos definidos y se establezca una organización que sea beneficiosa para el desarrollo del sistema.

Cuando se desarrolla sistemas de gran complejidad, es necesario definir cuales son los atributos de calidad que el sistema deberá cumplir. La posible solución de un sistema debería ser analizado, para determinar si se cumplen dichos atributos establecidos lo antes posible, para aumentar la probabilidad de éxito de un proyecto de software. Para esto sirve la arquitectura de un sistema, como un conjunto de estructuras del sistema que me permiten ver las relaciones y propiedades entre sus elementos. La arquitectura funciona para permitir una comunicación de dicha solución previa, el razonamiento y el análisis del sistema a construir.

El diseño de esta arquitectura se vuelve punto clave en *programming in the large* pero no es lo único importante, también son necesarios pasos previos a la construcción como la elicitación de los requerimientos funcionales y no funcionales del sistema y la priorización de estos atributos para poder satisfacer posibles conflictos que pudieran surgir al construir la arquitectura.

Por el contrario, *programming in the small* requiere de un menor nivel de planificación y de gestionamiento debido a que tamaño del sistema a desarrollar es pequeño. El desarrollo se puede hacer incrementalmente realizando pequeñas iteraciones de tipo waterfall sin importar demasiado en procesos formalmente definidos. Cuando se tiene un proyecto de este tipo se pueden utilizar metodologías ágiles que permiten enfocarse en el desarrollo, donde es más importante un prototipado rápido que un producto estable y de calidad .

10.2. UP *vs* SCRUM

Las metodologías utilizadas en los trabajos prácticos pueden ser comparadas en varios niveles. A un nivel más amplio, se puede decir que ambas tratan de desarrollar software utilizando un modelo iterativo incremental. Mirando más detenidamente comienzan a surgir diferencias significativas que muestran distintos *approaches* a la forma de desarrollar software. La metodología de SCRUM está basada en una comunicación continua con el Project Owner durante los distintos ciclos donde cada uno produce un software funcionando para validar. Por otro lado, las iteraciones de UP se denominan fases, son de distintos tipos que difieren en sus entregables, como podrían ser documentos de requerimientos, de análisis de dominio o software.

SCRUM se basa en la continua comunicación entre los integrantes del grupo de desarrollo y el Project Owner para lograr el avance del proyecto. En esta metodología se toman en cuenta los stories que tienen mayor valor para el negocio para planificar el próximo ciclo. UP desarrolla la formalización de dicha comunicación mediante documentos formales ayudados por plantillas que especifiquen los riesgos, atributos de calidad y casos de uso. Son estos últimos artefactos en los que UP se basa para la construcción del sistema, donde se basa en los riesgos y los atributos de calidad para planificar una arquitectura base que los resuelva y que casos de uso desarrollar primero.

Los cambios de los requerimientos en SCRUM son bienvenidos mientras que los mismos no afecten la iteración actual. Estos cambios pueden ser expresados en el Backlog para luego ser considerado en próximas iteraciones. Esto y la falta de estimaciones precisas en todos los elementos del Backlog (debido a que no se descompuso las mismas en tareas estimables) lleva a que la fecha de finalización del proyecto no sea certera. La planificación del ciclo en SCRUM se realiza previo a su comienzo, teniendo en cuenta estimaciones de esfuerzo y descomposición de tareas.

En cambio, en UP, todo el desarrollo del software está representado en el cronograma del proyecto y el mismo se planifica completamente en sus primeras iteraciones, por lo que el proyecto está dirigido para cumplir con ese plan. Los cambios en este tipo de metodologías más tradicionales deben ser inspeccionados más cuidadosamente, analizando su impacto tanto en termino de costos, fechas y cambios a producir en el sistema. Todo cambio debe ser reflejado en el plan del proyecto.

Por lo mencionado anteriormente, SCRUM es una metodología más adaptable a proyectos más cortos en tiempo, orientado a usuario, pequeños en grupo, de poca criticidad y sin fechas de finalización establecidas.

Por lo contrario, UP lo consideramos recomendable para proyectos con fechas establecidas, críticos o de gran escala tanto en tamaño como en complejidad.

10.3. Conclusiones del trabajo

En este trabajo práctico, se nos planteó un escenario diferente que en el anterior. Debíamos basarnos en una metodología tradicional como es UP dejando de lado la forma de trabajar que veníamos llevando para el primer tp, con metodologías ágiles, y adoptando una nueva forma de enfrentar el desarrollo de un sistema de gran escala.

Partiendo de esta base, nuestro avance ya no se basaría en construir un pequeño análisis preliminar para luego ir produciendo código de a pequeñas partes e ir mejorando el producto a partir de estos resultados parciales, sino que debíamos planear cuidadosamente cada paso a seguir utilizando modelos formales, resultando en un desarrollo considerablemente resistente a cambios y basado fuertemente en los riesgos y en los atributos de calidad.

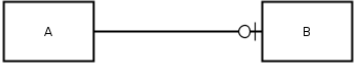
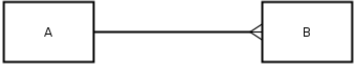
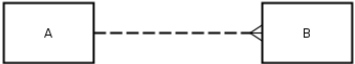
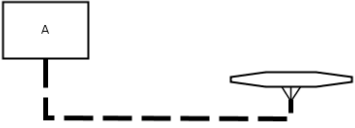

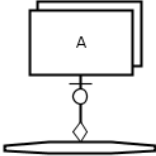
El desafío principal consistió en entender con claridad que era lo que se buscaba de nuestro proyecto, y a partir de eso, generar una estructura adecuada para lograr los objetivos planteados. Esto resultó en una tarea para nada sencilla, ya que a pesar de la claridad del enunciado muchos aspectos no funcionales del mismo quedaban librados a la interpretación nuestra y merecían un estudio más profundo en pos de generar el mejor resultado posible.

El problema que encontramos en este sentido fue que, en el trabajo práctico anterior, el hecho de ir programando distintas funcionalidades a medida que avanzábamos con el planeamiento nos iba esclareciendo dudas y dándonos pistas sobre como diseñar aquello que todavía estaba en proceso, así como planteando cambios a hacer en el resto del desarrollo. Con este nuevo enfoque, esto ya no era posible y debíamos si o si pensar todo desde “el aire”, o bien buscar ayuda con nuestro corrector.

Junto con esto, el cliente (o en nuestro caso nuestro corrector asignado) ya no estaría involucrado directamente dentro del avance de nuestro proyecto, sino que debíamos pactar reuniones con él en momentos de consultas. De todas formas esto lo vinimos haciendo también en la primera etapa, por lo que no implicó un cambio significativo.

11. Apéndice I: Referencia de conectores

A continuación damos una tabla de referencia para podernos referir más comodamente a los conectores usados (los que no detallamos fueron usados exactamente como aparecen en la cartilla dada por la cátedra):

Conector	Nombre
	Pipe (Representa el dado por la cátedra)
	Call Sincrónico (Representa el dado por la cátedra)
	Client Server (Representa el dado por la cátedra)
	Canal de Conexión Segura (Conector custom)
	Publish/Subscribe con respuesta (Conector custom)
	Publish/Pipe (Conector custom)