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

Ingeniería de Software II

Trabajo Práctico 2

Developing in-the-large - Planificación

Grupo 5 - *El nene está bien*

Integrante	LU	Correo electrónico
Martín Alejandro Miguel	181/09	m2.march@gmail.com
Iván Postolski	216/09	ivan.postolski@gmail.com
Juan Manuel Martinez Caamaño	276/09	jmartinezcaamao@gmail.com
Matías Incem	396/09	matias.incem@gmail.com
Pablo Gauna	334/09	gaunapablo@gmail.com

Índice

1. Introducción

Para este trabajo práctico se nos pide planificar y diagramar la arquitectura de una aplicación para ahorradores. Esta aplicación, de forma similar a *Precio Justo* (presentada en el primer trabajo práctico), deberá servir para que los ahorradores puedan consultar ofertas de distintos productos, publicadas en varias redes sociales o medios de Internet. Sin embargo, a diferencia de *Precio Justo*, el alcance de esta aplicación resulta ser mucho mayor.

Para esta entrega se nos pide:

1. *Atributos de calidad* identificados en la situación planteada, presentado en la sección 3.

2. Diagrama de la *arquitectura*, en la sección 4. Además, se explica cómo la arquitectura satisface los atributos de calidad, planteados en esta entrega, y los casos de uso introducidos en la entrega anterior.

3. Comparación entre *Unified Process* y *Scrum*, y también una comparación entre *Programming in the small* y *Programming in the large*. Esto último se detalla en la sección 5.

Adicionalmente, en la sección 6, los casos de uso de la entrega anterior son recordados.

2. Diccionario de términos

- **Oferta**: Información sobre el precio de un producto (válido) determinado en un lugar determinado. Lleva un identificador de su autor y del tiempo en el que fue enviada.
- **Usuario autenticado**: Usuario que tiene un identificador (o "nombre") único dentro del sistema, y que confirmó su identidad ante el mismo para establecer una comunicación.
- **Admin/Administrador**: Usuario especial del sistema que tiene acceso a funciones no disponibles para usuarios normales, y que puede tomar decisiones que afectan el funcionamiento del sistema para todos los usuarios.
- **Regla de sustitución**: Regla del sistema que relaciona un producto con otro similar, indicando que es sustituible, es decir, que un usuario interesado en el primero podría cambiar de idea e interesarse por el segundo si se le proporciona información.
- **Regla de asociación**: Regla del sistema que relaciona un producto con otro indicando que están asociados, es decir, que un usuario interesado en el primero podría interesarse también por el segundo (y comprar ambos) si se le proporciona información.
- **Query**: Pedido que un usuario realiza al sistema, mediante la aplicación, con el objetivo de obtener un conjunto de ofertas como respuesta.

3. Atributos de calidad

En esta sección se presentan los distintos *atributos de calidad* que fueron obtenidos de analizar el documento original con las especificaciones de los interesados y los resultados del QAW (*Quality Assurance Workshop*).

3.0.1. Atributo de modificabilidad:

Se trabajará con todo tipo de productos, clasificados en rubros. Se espera que a futuro se pueda incrementar la cantidad de rubros existentes, y también controlar el conjunto de productos válidos y la pertenencia de cada uno a determinados rubros.

- Fuente: Administradores del sistema.
- Estimulo: Se desea agregar un nuevo producto/rubro.
- Artefacto: Subsistema de rubros y productos.
- Entorno: Funcionamiento normal.
- Respuesta: Utilizando la interfaz de administración, se agrega un nuevo producto/rubro al sistema, sin detener su funcionamiento.
- Medida: El administrador puede acceder mediante una interfas apropiada al sistema y realizar los cambios sin que sea necesario cambiar el código fuente del sistema.

3.0.2. Atributo de modificabilidad:

Las reglas de asociación y sustitución entre productos todavía no están bien definidas, por lo que se anticipa que deberán cambiarse con frecuencia.

- Fuente: Administradores del sistema
- Estimulo: Se desea agregar/modificar las reglas de asociación y sustitución.
- Artefacto: Subsistema de reglas.
- Entorno: Modo de mantenimiento.
- Respuesta: Se modifican las reglas de asociación y sustitución.
- Medida: En menos de 5 horas de trabajo se implementan y ponen en funcionamiento las modificaciones a estas reglas.

3.0.3. Atributo de usabilidad:

Se desarrollará un sistema de confianza configurable individualmente, el cual será fácil de operar por un usuario común.

- Fuente: Usuario autenticado.
- Estimulo: Modifica las reglas de confianza de ofertas.
- Artefacto: Módulo UI (tanto el cliente web como cliente móvil).
- Entorno: Funcionamiento normal.
- Respuesta: El sistema provee un asistente de configuración de confianza de ofertas, que ayuda a modificar dichas preferencias.
- Medida: Un usuario nuevo del sistema logra completar la tarea en menos de 15 minutos.

3.0.4. Atributo de detección y prevención de fallas:

Si la estructura de un sitio web monitoreado por un bot cambia de manera que la información no puede ser interpretada, se deberá ajustar el bot para que pueda continuar funcionando correctamente.

- Fuente: Las páginas monitoreadas.
- Estimulo: Se produce un cambio en la estructura de las páginas monitoreadas.
- Artefacto: Bot del sistema de obtención de datos.
- Entorno: En funcionamiento normal.
- Respuesta: Se detecta el cambio en la estructura de la página, se detiene el bot, y se informa a los administradores.
- Medida: Cuando en una serie de consultas se detecta un 70 % de cambios en su estructura, se considera como error, se informa de esta situación y el bot se detiene.

3.0.5. Atributo de usabilidad:

Se desea que la interfaz de la aplicación para realizar consultas sea intuitiva y amigable.

- Fuente: Un usuario nuevo.
- Estimulo: El usuario desea realizar una consulta.
- Artefacto: Interfaz Móvil / Interfaz Web.
- Entorno: Funcionamiento normal.
- Respuesta: Se realiza la consulta y se informan los resultados.
- Medida: En menos de 5 minutos, un usuario nuevo comprende la interfaz y puede comenzar a utilizarla.

3.0.6. Atributo de usabilidad:

Una característica que se espera de sistema será su inmediata respuesta y su capacidad de anticiparse a nuestras búsquedas. Los resultados deben aparecer a medida que ingresamos una búsqueda.

- Fuente: Un usuario.
- Estimulo: El usuario desea realizar una consulta.
- Artefacto: Interfaz Movil / Interfaz Web.
- Entorno: Funcionamiento normal.
- Respuesta: Se comienzan a presentar resultados, que son las ofertas populares / más buscadas / más recomendadas.
- Medida: En menos de 1 segundo luego de que se comenzo a escribir una consulta, el sistema comienza a sugerir posibles resultados.

3.0.7. Atributo de disponibilidad:

El servicio debe estar disponible en todo momento, y no debe caerse, en la medida de lo posible.

- Fuente: Usuario.

- Estimulo: Se desea realizar una consulta.
- Artefacto: Sistema Central TPA.
- Entorno: Funcionamiento normal.
- Respuesta: Se realiza la consulta al sistema, se obtienen resultados y son mostrados al usuario.
- Medida: Siempre que el usuario disponga de conexión, en el 99 % de los casos el envío de la consulta al sistema central y la recepción de la respuesta se realizan con éxito.

3.0.8. Atributo de disponibilidad:

Cuando el usuario no tenga conexión a Internet, debe poder consultar a la aplicación de todos modos, y recibir una respuesta satisfactoria.

- Fuente: Usuario.
- Estimulo: Se desea realizar una consulta.
- Artefacto: Interfaz móvil.
- Entorno: El dispositivo móvil no tiene conexión.
- Respuesta: Si la consulta se encuentra disponible sin conexión, se obtienen los resultados y son mostrados al usuario.
- Medida: Las últimas consultas y las consultas más populares al momento de la última conexión se encuentran disponibles.

3.0.9. Atributo de modificabilidad:

Se contratará temporariamente el oneroso servicio de la empresa SpamBust, mientras se pone a punto una implementación propia. La transición entre los servicios debe ser simple y transparente, y estos deben poder coexistir en caso de que sea necesario.

- Fuente: Equipo de desarrollo.
- Estimulo: Se desea modificar el funcionamiento del servicio de detección de Spam.
- Artefacto: Sistema de detección de Spam.
- Entorno: Funcionamiento normal.
- Respuesta: Se modifica el funcionamiento del servicio.
- Medida: El funcionamiento del servicio de detección de Spam debe poder modificarse (cambio de proveedor, nuevo proveedor, modificar un servicio existente, etc. . .) con el sistema en funcionamiento, y deberían poder coexistir varios sistemas de detección de Spam.

3.0.10. Atributo de auditabilidad:

Cada martes debe publicarse un informe con la cantidad de ofertas falsas detectadas, productos con precios dudosos, información contradictoria y alertas de engaños reportados por los usuarios. La información completa que lleve a la conclusión de que un precio es falso tendrá que quedar disponible para una evaluación posterior más minuciosa.

- Fuente: Administradores del sistema.

- Estimulo: Se desea ver el resumen de ofertas falsas, detectadas, etc...
- Artefacto: Sistema de detección de Spam.
- Entorno: Funcionamiento normal.
- Respuesta: Se obtiene un resumen con la información de ofertas falsas detectadas, precios dudosos, etc...
- Medida: Cada vez que se elimina / modifica una oferta se registra *qué oferta, por quién, cuándo, por qué motivo.*

3.0.11. Atributo de extensibilidad:

El sistema obtendrá datos de diversas fuentes, que incluyen redes sociales, SMS y sitios web. En el futuro se podrán agregar nuevas fuentes de datos, ya sean similares a las anteriores o diferentes.

- Fuente: El equipo de desarrolladores.
- Estimulo: Se desea implementar una nueva fuente de datos.
- Artefacto: Sistema de obtención de datos.
- Entorno: En funcionamiento normal.
- Respuesta: Se implementa la fuente de datos y se la integra al sistema.
- Medida: La fuente de datos se integra con el sistema en menos de 1 hora, sin detener al sistema.

3.0.12. Atributo de performance:

La actualización de precios debe realizarse en el instante mismo en que se publica la información.

- Fuente: Usuario externo
- Estimulo: Se publica una oferta en un medio monitoreado por el sistema de obtención de datos.
- Artefacto: Sistema de obtención de datos.
- Entorno: Funcionamiento normal.
- Respuesta: El sistema de obtención de datos identifica esta oferta y la agrega a la base de datos.
- Medida: La oferta se comienza a tener en cuenta como máximo un minuto despues de que fue publicada por una fuente de datos. Es importante destacar que existe cierto tiempo desde que un usuario publica una oferta en un medio hasta que este medio la hace disponible; este tiempo no es controlable.

3.0.13. Atributo de modificabilidad:

El sistema será capaz de minar la web en busca de ofertas en las páginas de las grandes cadenas de supermercados, sitios de descuento y páginas de subastas. A futuro se podrá decidir que se incorporen nuevas páginas a esta búsqueda.

- Fuente: El equipo de desarrolladores.
- Estimulo: Se desea *implementar/modificar* un bot para obtener datos de una página web determinada.
- Artefacto: Sistema de obtención de datos.

- Entorno: En funcionamiento normal.
- Respuesta: Se implementa el bot y se integra con el sistema.
- Medida: Implementar o modificar el bot modifica un único componente (el respectivo al bot) y se integra con el sistema en menos de 1 hora, sin detener al sistema.

4. Arquitectura

4.1. Referencia de conectores

A continuación presentamos una referencia a los conectores utilizados en los diagramas que se presentarán en las siguientes secciones. Los conectores *no-standard* utilizados son especificados más abajo.

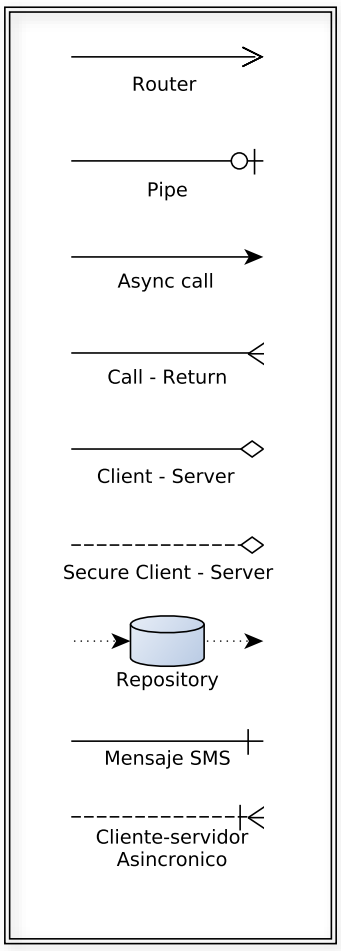

Figura 1: Referencia de conectores.

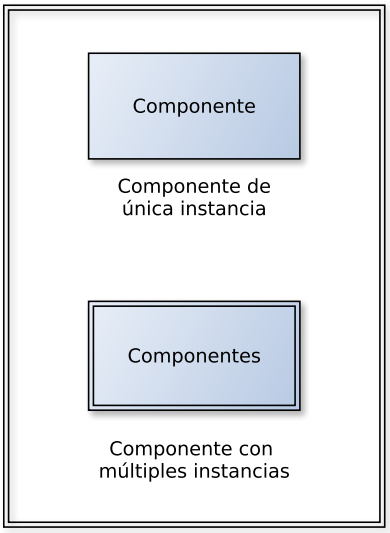

Figura 2: Referencia de componentes.

Conectores no standard

- **Client-Server Async** : Funciona igual que un client-server, pero el cliente no espera una respuesta. El server luego puede retornar su respuesta de manera asincrónica.
- **Secure Client-Server** : Este conector resulta muy similar al conector client-server. Se diferencia únicamente en que el cliente, antes de enviar una consulta al servidor, *encripta* el mensaje para luego enviarlo. Ya en

el servidor, este último desencripta el mensaje y lo interpreta. Para la respuesta, la comunicación sucede de forma similar; ahora es el servidor quien encripta el mensaje y el cliente quien lo desencripta.

4.2. Conectividad externa del sistema TPA

En el siguiente diagrama vemos la disposición global del sistema central **TPA** respecto a los agentes externos con los que interactúa. Además se presenta una primera subdivisión del sistema principal en sus componentes más importantes.

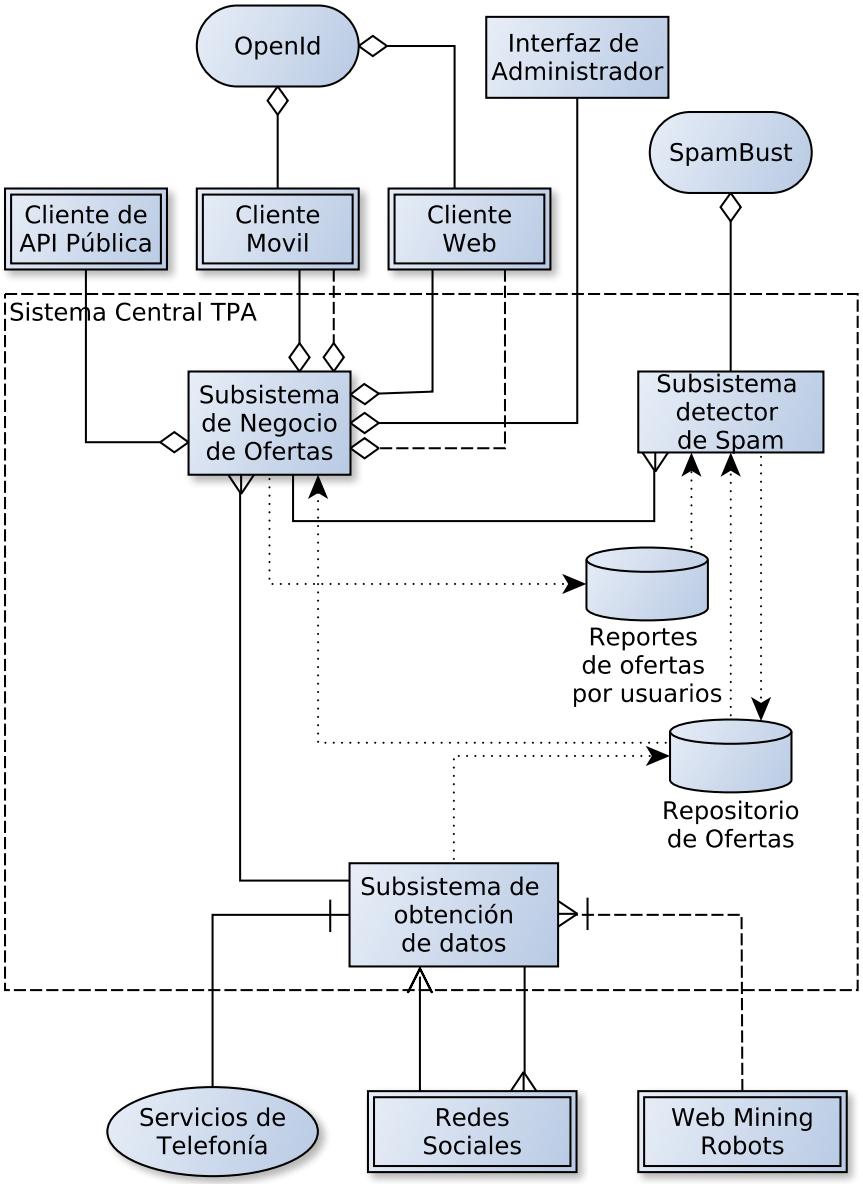

Figura 3: Diagrama arquitectónico global del sistema **TPA**.

Aquí vemos al Sistema central TPA dividido en 4 componentes. El Subsistema de Negocio de Ofertas es el encargado de responder a las consultas de los posibles

clientes. En este momento los clientes son nuestras aplicaciones web y móvil, así como aquellos que deseen consumir nuestra API pública. Dentro del Subsistema de Negocio de Ofertas tenemos los componentes encargados de comprender los pedidos, administrar las preferencias y reputación de usuarios, administrar las publicidades generadas por el sistema, mantener las reglas de asociación y sustitución y contener la información de los productos y rubros aceptados.

El Subsistema de Obtención de datos es el encargado de recurrir a las distintas fuentes de datos con el fin de obtener ofertas para productos. Por lo tanto, contiene los medios necesarios para poder extraer información de Redes sociales y para poder recibir información mediante SMS. Como además se decidió poder registrar sitios web con interfaces variadas (muy distintas a las Redes sociales), se cuenta con los componentes Web Mining Robots, que constituyen robots que saben recorrer ciertos sitios específicos en busca de ofertas. El Subsistema de Obtención de datos sabe cómo comunicarse con los robots, quienes le proveen información.

Las ofertas obtenidas por el Subsistema de Obtención de datos le son provistas al Subsistema de Negocio de Ofertas mediante el uso de un repositorio intermedio. El mismo es revisado continuamente por el Subsistema de Detección de Spam, que contiene mecanismos para detectar ofertas de naturaleza dudosa. En este momento, el principal mecanismo es el sistema de terceros SpamBust.

4.3. Subsistema de Negocio de Ofertas

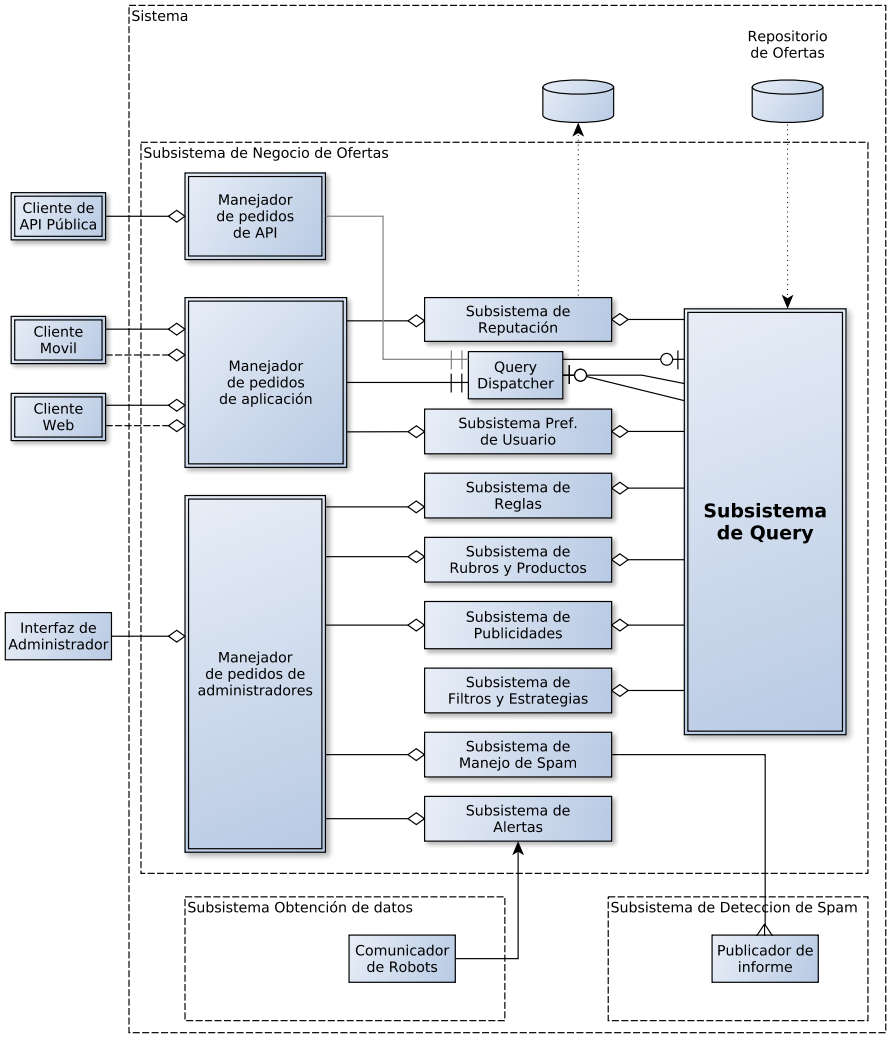

Figura 4: Diagrama arquitectónico con el detalle del Subsistema de Negocio de Ofertas.

El Subsistema de Negocio de Ofertas es un componente que contiene los distintos *subistemas de negocio* y los componentes de interacción con el cliente. Los *subsistemas de negocio* son componentes que se encargan de administrar datos y mecanismos relacionados con las distintas funcionalidades del sistema. Los *componentes de interacción con el cliente* son los que saben comunicarse con los distintos clientes y administrar sus pedidos.

En las siguiente secciones hacemos principal enfoque en el *manejo de pedidos de ofertas*, que es la principal funcionalidad del sistema.

4.3.1. Manejo de pedidos de ofertas

El manejo de pedidos de ofertas provenientes de clientes se resuelve de la siguiente manera: el pedido es aceptado por un Manejador de pedidos de API o Manejador de pedidos de aplicación; este recibe el pedido y le asigna un identificador de pedido. Luego arma un mensaje con su identificador, el identificador del pedido y el contenido del pedido y envía dicho mensaje como cliente (asincrónico) de algún Query Dispatcher. Cabe destacar que el manejador tiene memoria de aquellos pedidos que todavía le debe responder a sus clientes. El Query Dispatcher toma el mensaje y lo pone en la cola de trabajo del Subsistema de Query. Al terminar el procesamiento del mensaje el resultado del mismo es devuelto en la cola de resultados del Query Dispatcher, que según el identificador del mensaje original sabe enviarlo asincrónicamente hacia el manejador que lo creó. El manejador recibe el resultado y se lo devuelve al cliente que originó el pedido.

Esta forma de manejar los pedidos de los clientes sobre ofertas satisface el atributo de calidad 7, que refiere a la *disponibilidad* del sistema para responder consultas. Con los componentes descriptos anteriormente, ningún pedido debería ser rechazado por no haber nadie disponible para atenderlo; si se asume una coherente asignación de instancias de manejadores, dispatchers y de elementos procesadores de queries. Lo que se logra es desacoplar la consulta y el procesamiento, de forma de aprovechar el pipeline de procesamiento contenido en el Subsistema de query, y así aumentar el throughput. Se espera así que el sistema pueda manejar grandes volúmenes de consultas por productos.

4.3.2. Manejo de mensajes de otros subsistemas de negocio

El manejo de mensajes relacionado con las otras funcionalidades (como ser cambios de reputación y preferencias de usuario) se hace de manera diferente al de pedido de ofertas, porque el volumen esperado para los mismos es menor y además su resolución es mucho más rápida. Usamos entonces un conector client-server convencional entre el manejador y el subsistema de negocio correspondiente.

4.4. Subsistema de Query

En el siguiente diagrama se presenta el Subsistema de Query donde, dada una consulta por productos (**query**), se arma la respuesta para el usuario que realizó la consulta. Este es el subsistema de negocio más importante, ya que es encargado de la principal funcionalidad del sistema. Sobre el mismo se aplican varios atributos de calidad relacionados con *performance* y *modificabilidad*. La funcionalidad de este sistema responde a los *casos de uso* 4 (consultar por un producto), 9 y 10 (relacionados con productos asociados y sustituibles), 11 (mostrar publicidades), y 16 (consultar estando autenticado y aprovechar las preferencias).

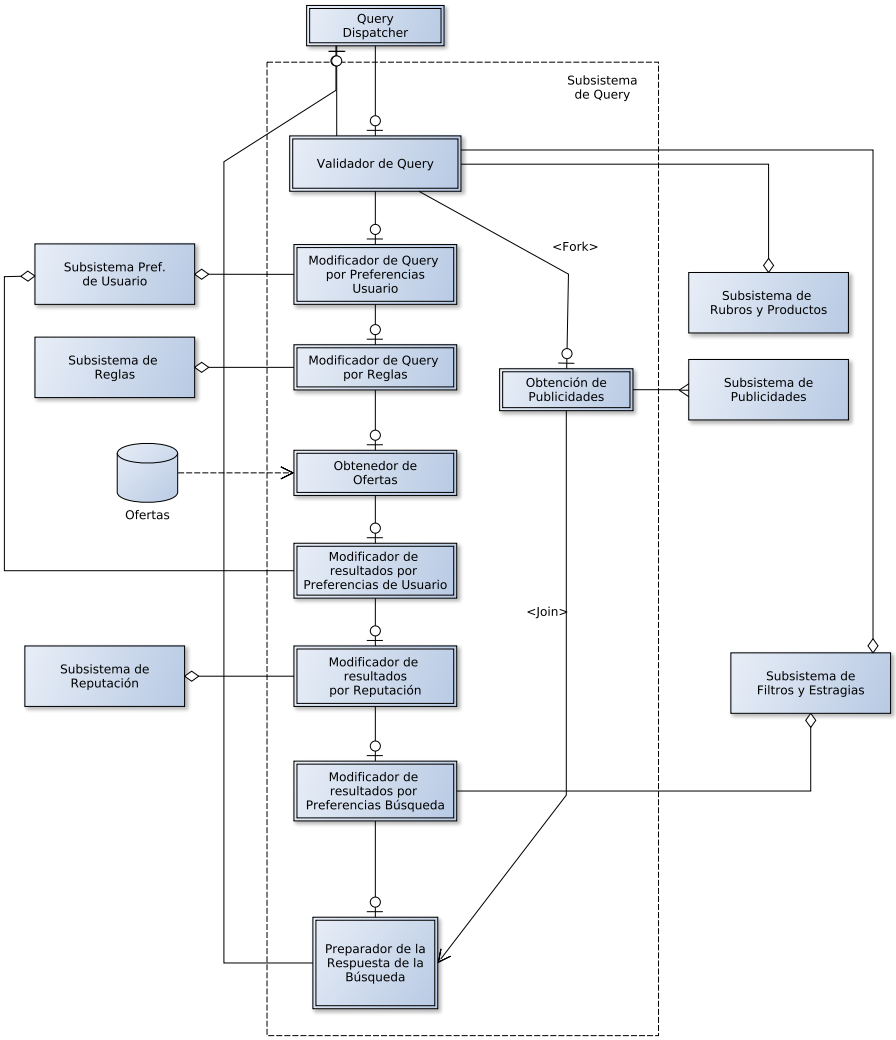

Figura 5: Diagrama arquitectónico con el detalle del Subsistema de Query.

La funcionalidad de este componente es la de procesar un pedido por productos para armar todo el contenido que le será devuelto al usuario. Este procesa-

miento incluye varias funcionalidades del sistema. Todas estas pueden realizarse de forma encadenada, lo cual lleva a que el componente trabaje con un estilo arquitectónico de tipo *pipes and filters*, en el cual cada etapa de procesamiento se realiza secuencialmente pero de forma independiente. La elección de este estilo tiene como fin lograr *modificabilidad* y *performance*, ya que las etapas pueden reubicarse sin tener que modificar fuertemente el sistema; además, para aquellas etapas que requieren mayor poder de cómputo pueden agregarse más recursos, de forma que no generen un cuello de botella.

En primer lugar se valida que el pedido sea realizable (i.e.: es de un producto soportado y los filtros y estrategias son válidos en el sistema) por el Validador de Query. Si la consulta no es válida el componente ya genera una respuesta y la encola para ser devuelta por el Query Dispatcher. Sino continúa el proceso.

Dentro de las funcionalidades que forman parte de una *consulta*, se definieron distintas modificaciones que podrían sufrir las mismas. Los siguientes dos componentes (Modificador de Query por Preferencias Usuario y Modificador de Query por Reglas) se encargan de comunicarse con los subsistemas correspondientes que definen como debe realizarse estas modificaciones. El validador encola la consulta para que sea procesada por estos componentes y a su vez la encola para que sea trabajada por el Generador de publicidades, que se encarga de generar las publicidades que formarán parte de la respuesta. Esto se realiza con un mecanismo de *fork and join*, donde el Generador de publicidades toma la *query*, la trabaja y luego espera a que sea consultado por el resultado. De esta forma, la generación de propagandas, que se considera independiente del resto del procesamiento a realizar, puede hacerse en paralelo, mejorando la *performance del proceso*.

Las modificaciones que se realizan a la consulta están relacionadas con las preferencias del usuario (que puede decidir no ver las consultas provenientes de algunos usuarios) y las reglas de asociación y sustitución, (que establecen que distintas extensiones a las búsquedas). La naturaleza de estas modificaciones no está completamente definida pero estará contenida exclusivamente en los Subsistema de Preferencias de Usuario y Subsistema de Reglas, para lograr fácil *modificabilidad* de las misma.

Con la consulta (*query*) ya definida y ajustada, el componente Obtenedor de Ofertas accede al repositorio de ofertas y adquiere todas aquellas que responden a la consulta. Una vez realizado esto pasamos a la etapa donde se prepara el resultado. Por un lado el componente Modificador de resultados por Preferencias de Usuario se encarga de tareas como jerarquizar las ofertas según las preferencias del usuario. Las ofertas luego son decoradas, eliminadas y reorganizadas según la reputación de los usuarios que las originaron (componente Modificador de resultados por Reputación). A continuación se realiza un nuevo trabajo sobre los resultados aplicando los filtros y estrategias seleccionados por el usuario en la consulta original (componente Modificador de resultados por filtros y estrategias). Finalmente, el componente Preparador de la Respuesta de la Búsqueda prepara todo el paquete de respuesta que será enviado de nuevo al usuario.

4.5. Subsistemas de Información de Negocio

En el diagrama del Subsistema de Negocio de Ofertas (figura 4) se encuentran varios subsistemas que manejan información de negocio. Estos subsistemas son: el Subsistema de Publicidades, el Subsistema de Rubros y Productos, el Subsistema de Reglas, el Subsistema Pref. de Usuario, el Subsistema de Reputación, el Subsistema de Filtros y Estrategias, el Subsistema de Manejo de Spam y el Subsistema de Alertas. El Subsistema de Query es también considerado un *subsistema de negocio*, pero por su relevancia es tratado individualmente en las secciones anteriores. En todos los otros casos los *subsistemas de negocio* son accedidos mediante una conexión de tipo *cliente-servidor* ya que proveen un servicio respecto a la información de negocio contenida en cada uno. Además, se busca permitir concurrencia en el acceso a cada uno de los servicios.

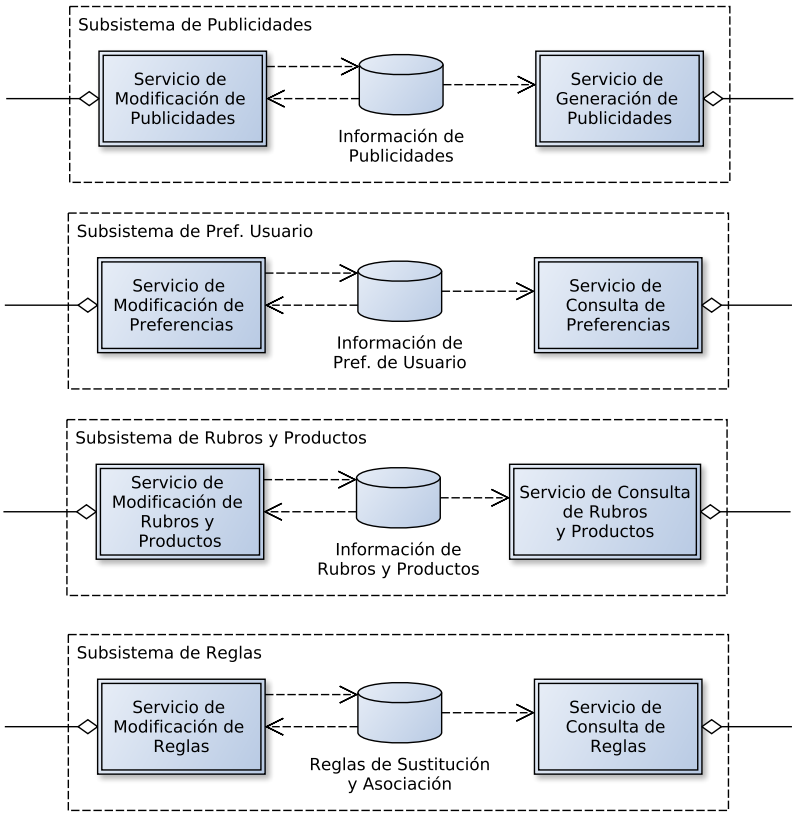

Figura 6: Diagramas arquitectónicos con el detalle de algunos Subsistemas de Información de Negocio.

Cada uno de estos subsistemas permiten modificar la información de negocio por usuarios y administradores según corresponda. Tanto los datos como la inteligencia de negocio contenida en cada uno de los subsistemas es además publicada mediante otro componente para uso interno del sistema, en particular, para ser consultada por el Subsistema de Query. En la medida de lo posible, el acceso podrá ser eficiente y concurrente por distintos clientes. La *modificabilidad*

de cada una de los mecanismos de negocio representado en cada subsistema estará contenido en cada uno de ellos.

Los atributos de calidad 1 y 2 refieren a la modificabilidad de dos de estas secciones del negocio. Mediante la interfaz de administrador un *administrador* puede modificar productos y rubros. Los componentes que comunican al *administrador* con el Subsistema de Rubros y Productos están preparados para permitir modificar el repositorio que contiene los productos, rubros y las asociaciones entre ellos. Luego solo se modifica el repositorio. Lo mismo sucede para agregar reglas de asociación en el sistema Subsistema de Reglas. Además, si llega a cambiar la naturaleza de las reglas (que es sabido no están del todo definidas), toda la inteligencia de negocio asociada estará contenida en el subsistema por lo que solo se deberán modificar componentes del mismo. b

El Subsistema de Reputación difiere del resto de los subsistemas de negocio en que el repositorio que contiene los datos no es interno, sino que es externo, ya que es compartido con otros subsistemas. Su diagrama se muestra a continuación.

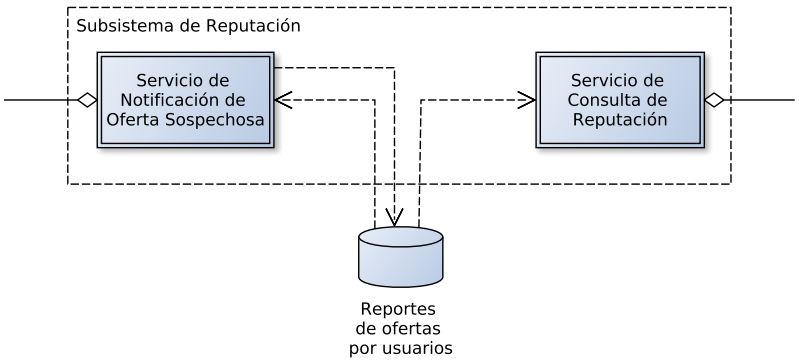

Figura 7: Diagrama arquitectónico con el detalle del Subsistema de Reputación.

4.6. Interfaz móvil

La interfaz móvil es uno de los componentes encargados de facilitar una interfaz cómoda al usuario, y comunicar los pedidos de este con el servidor central de *Twitteando para ahorrar*.

A través de este subsistema, el usuario debería ser capaz de:

1. Realizar consultas de ofertas.
2. Denunciar ofertas como falsas.
3. Conectarse con una cuenta en un proveedor de *OpenId*.
4. Modificar la configuración del usuario.

Al mismo tiempo, ciertos atributos de calidad guían el diseño de este subsistema. Algunos de ellos son la necesidad de *responder rápidamente al usuario*, inclusive antes de que este comience a tipear algo (atributo de usabilidad 6), y la necesidad de *garantizar disponibilidad*, a pesar de que no exista conexión (atributo 8).

Al módulo de interfaz de usuario se le delega la tarea, especificada en el atributo de usabilidad 5, de presentar los datos de una forma 'bonita y comprensible' al usuario. Además, este módulo se comunica con el componente encargado de facilitar la posibilidad de autenticarse con *OpenId*. Vale destacar que el componente de comunicación con OpenId se encuentra en el cliente y no del lado del servidor, ya que es el *proveedor de OpenId* (Facebook, LinkedIn, etc.) el cual ofrece un formulario para autenticarse. De esta forma, durante la autenticación, el control es cedido al *proveedor de OpenId*.

Por otro lado, la interfaz de usuario se comunica con el módulo que intercambia mensajes con el sistema central de *twitteando para ahorrar*. Se decidió que, tanto los mensajes enviados desde la interfaz al módulo de comunicación, como los del módulo de comunicación a la interfaz de usuario, serán enviados de forma asíncrona. Esta ultima decisión tiene como objetivo evitar 'congelar' la interfaz esperando por una respuesta del servidor.

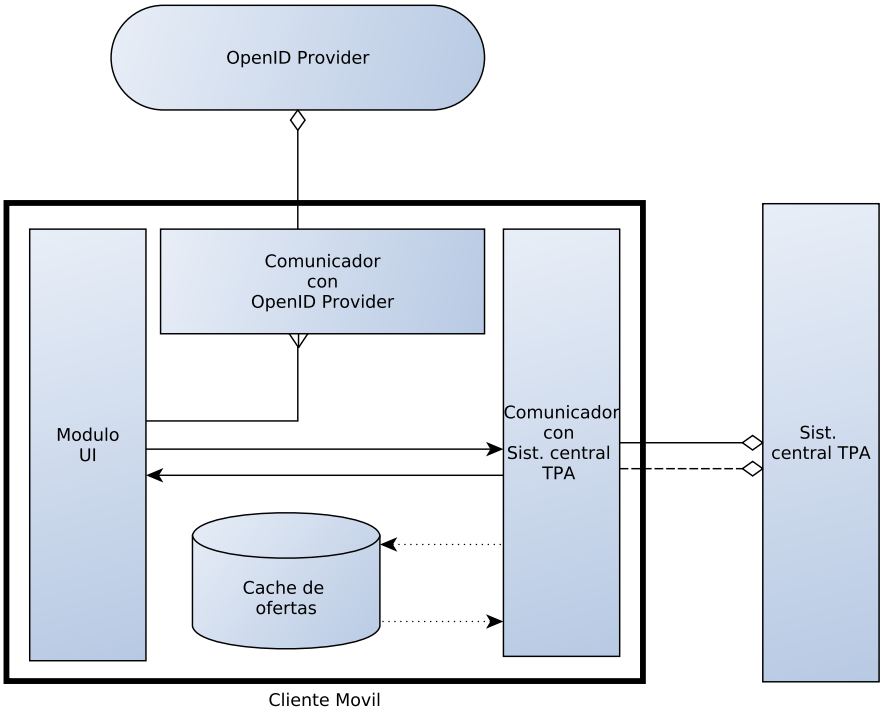

Figura 8: Diagrama arquitectónico con el detalle del Cliente móvil.

4.6.1. Módulo de comunicación con el sistema de Twitteando para ahorrar

Este módulo convierte las órdenes que el usuario envía desde la interfaz gráfica, en consultas efectivas por ofertas, entre otras funcionalidades.

Para comprender el funcionamiento de este subsistema, vamos a plantear el siguiente ejemplo:

Tito Codito, conocido ahorrador, abre la interfaz móvil de Twitteando para ahorrar, desde su smarthphone, y busca por ofertas de Pepas Tocoñato[1].

En el momento en el que Tito inicia la aplicación desde su móvil, el Modulo de consulta desde UI realiza una consulta al Modulo de consulta sin conexión, con el objetivo de obtener, de la Cache de ofertas, las ofertas registradas como *Más populares*. De esta forma, si bien Tito no comenzó a escribir una consulta, resultados interesantes se muestran en la interfaz de usuario. Estas ofertas se mantienen actualizadas en la cache por el Actualizador de consultas populares.

Luego, Tito comienza a escribir la consulta, *Pep*, sin embargo, la interfaz gráfica registra este evento antes de que la consulta se termine de tipear, y lo informa al módulo de consulta desde UI. Este módulo consulta al Módulo de consulta sin conexión por consultas sobre productos que empiecen con este patrón. Los resultados son comunicados inmediatamente a la interfaz. Así, resultados son mostrados antes de que la consulta se termine de tipear. La desventaja de esta solución es que solo se pueden 'predecir' resultados de consultas realizadas previamente, o consultas populares.

Una alternativa que no posee este problema es la de realizar esta consulta con el sistema central, sin verificar antes la cache; sin embargo, los tiempos necesarios para la transferencia de datos y resultados con el servidor hacen que esta opción sea inviable, y que una vez que se tienen resultados, Tito ya llevaría un tiempo considerable desde haber terminado de escribir la consulta.

Una vez que la interfaz detecta que la escritura de la consulta se ha terminado, se informa al Módulo de consulta desde UI, este resolverá:

- Si hay conexión con el servidor central, enviar esta consulta a este servidor y esperar por los resultados.
- En caso de no haber conexión, la consulta se intenta resolver utilizando la cache local.

En ambos casos, una vez obtenidos los resultados, estos son enviados a la interfaz gráfica para su visualización y, de esta forma, Tito ya puede escoger cuál es el lugar mas conveniente para comprar *pepas*.

De forma ortogonal a esto, existen otros dos subcomponentes de este módulo. Estos son, el módulo de cambio de preferencias y el módulo de denuncia de ofertas. El primero tiene la tarea de, ante un cambio de las preferencias del usuario, comunicárselas al servidor central. El segundo componente, comunica al servidor central la denuncia de una oferta como posible falso. La comunicación de ambos módulos con el servidor se realiza de forma *segura*, debido a que los datos que circulan en este momento son sensibles, y se busca evitar posibles usos malintencionados. Un ejemplo de lo anterior, seria atacar al sistema, denunciando ofertas, haciéndose pasar por otro usuario.

En la cache de ofertas, se almacenan las consultas recientes y sus resultados, y consultas populares y sus resultados. Si un usuario, sin conexión, consulta por ofertas de productos, que no son validos, no es necesario verificar la validez de estos, ya que resulta imposible que existan resultados de ofertas de este producto en la cache.

Es imposible modificar la configuración del usuario estando sin conexión. Consideramos que esta es la mejor opción, para evitar problemas de sincronización si el usuario posee varios dispositivos móviles o accede desde la interfaz

web. Además, por este motivo, la información sobre las fuentes de datos de confianza, se obtienen únicamente cuando hay conexión con el servidor central.

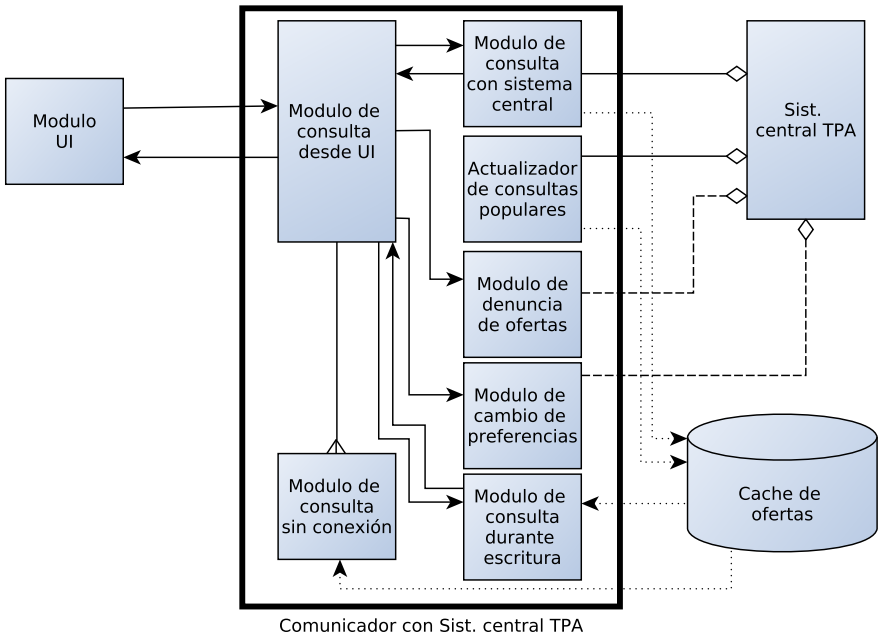

Figura 9: Diagrama arquitectónico con el detalle del Comunicador con Sist. central TPA del Cliente móvil.

Los atributos de calidad relacionados con la usabilidad del cliente móvil (atributos 3 y 5) son tenidos en cuenta en la arquitectura al tener desacoplado el Módulo UI, lo que hace que sea fácilmente modificable para lograr buena *usabilidad*.

4.7. Interfaz web

De forma similar a la Interfaz móvil, para permitir el acceso desde un conjunto diverso de dispositivos, optamos por dar lugar a una interfaz web de nuestra aplicación.

Esta interfaz se diferencia de la móvil en un punto importante, **no** se puede *persistir la información* obtenida de consultas anteriores. Esto imposibilita utilizar las estrategias, utilizadas en la interfaz móvil, para responder rápidamente a consultas del usuario, mientras estas se escriben. Por el mismo motivo, resulta imposible acceder sin conexión al servicio, sin embargo, es esperable que la misma interfaz sea inaccesible, en ausencia de conexión.

La única funcionalidad recortada en la interfaz web en comparación con la interfaz móvil es la posibilidad de anticiparse al usuario mientras este escribe la consulta.

Cuando un usuario se conecta a este servicio, al mismo tiempo que se carga la interfaz de usuario, comienzan a obtenerse las ofertas más populares, para

sugerirlas por la interfaz, antes de que se comiencen a realizar consultas por otros productos.

Luego, si se realiza una consulta por un producto en particular, esta es delegada al Modulo de consultas desde UI, y de forma sincrónica se realiza la comunicación con el servidor central.

La comunicación entre la interfaz gráfica y el módulo de consultas desde UI se realiza de forma asincrónica, para dar al usuario la sensación de agilidad en la respuesta, y no 'congelar' la interfaz gráfica.

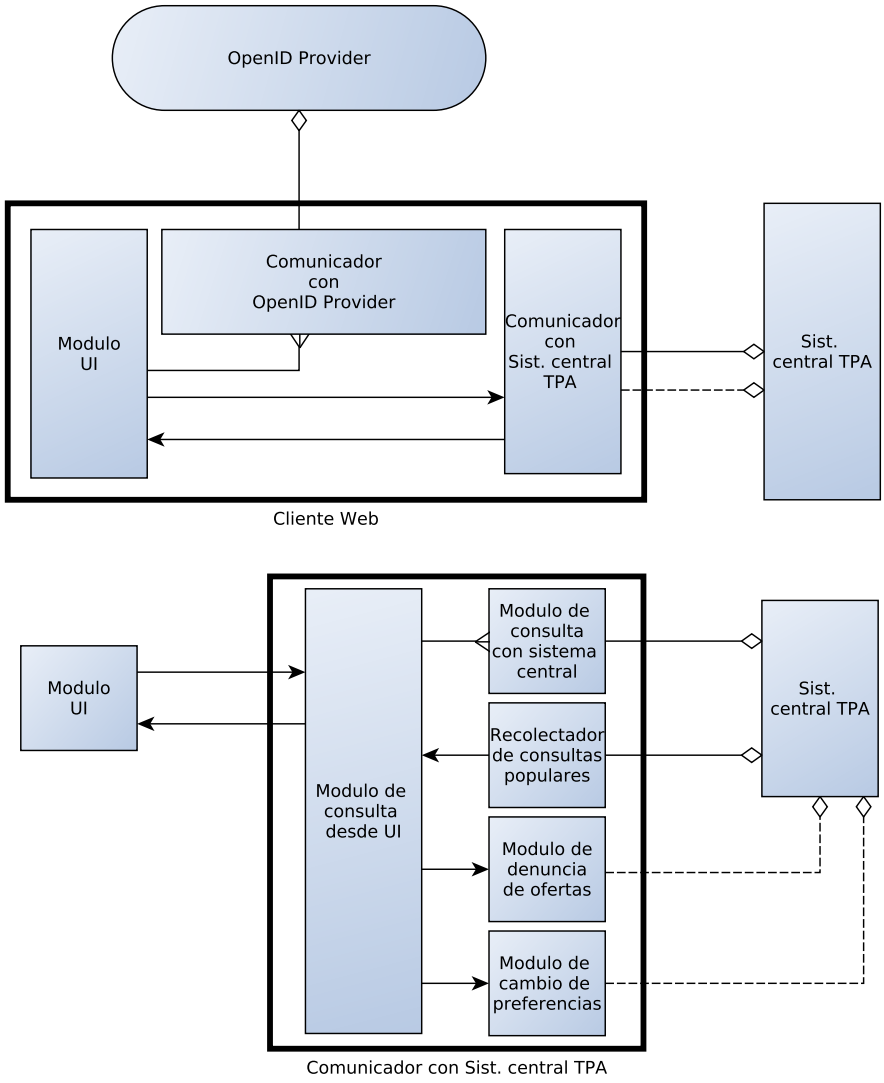

Figura 10: Diagrama arquitectónico con el detalle del Cliente web.

De la misma forma que en la Interfaz móvil, los atributos de usabilidad 3 y 5 están contemplados en un Módulo UI de fácil modificación.

4.8. Subsistema de Detección de Spam

El siguiente diagrama representa el Subsistema de Detección de Spam (**SDS**) donde se discriminan las ofertas deseadas en el sistema de las que no lo son (**Spam**). Una oferta puede ser considerada no deseada por ser falsa, tener un producto con precio dudoso, contener información contradictoria y/o haber sido reportada por los usuarios del sistema como engañosa.

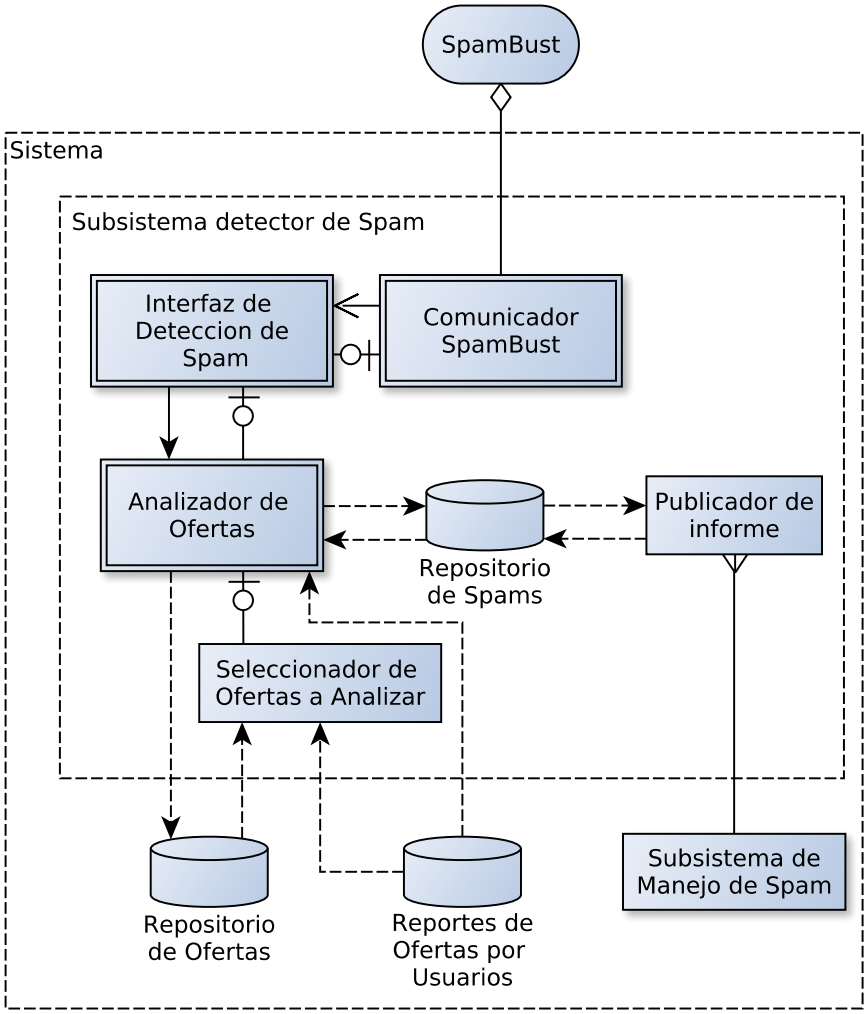

Figura 11: Diagrama arquitectónico con el detalle del Subsistema de Detección de Spam.

El SDS lee constantemente el Repositorio de Ofertas priorizando las ofertas recientemente agregadas que no han sido analizadas nunca como spam y las que fueron reportadas como engañosas por los usuarios; de esto se encarga la componente Selector de Ofertas a Analizar. Esta componente le pasa en forma de *pipe* las ofertas seleccionadas a la componente Analizador de Ofertas que se

encarga de discernir si la oferta es spam o no lo es. Por temas de *Performance* se decidió que haya múltiples instancias de esta componente, permitiendo que diversos recursos se ocupen de cada una de estas (*Concurrencia*).

El Analizador de Ofertas toma en consideración la respuesta del servicio de SpamBust, como también los reportes de usuarios o viejas ofertas detectadas como spam para tomar la decisión final de marcar una oferta como spam. Para la comunicación con los servicios de SpamBust interactúa con un componente *intermediario* llamada Interfaz de Detección de Spam, la cual se encarga de transferir los pedidos a la componente Comunicador SpamBust, que es el responsable de los detalles específicos de la comunicación con el servicio de SpamBust. Esto concede a la arquitectura una gran *Modificabilidad* ya que el Analizador de Ofertas se comunica solo con un *intermediario* que es fácilmente sustituido/modificado para apuntar a otra componente que no sea el Comunicador SpamBust, como por ejemplo una implementación propia que sustituya los servicios de SpamBust. Esto responde a lo especificado en el atributo de calidad 9.

El Analizador de Ofertas también se encarga de borrar del Repositorio de Ofertas aquellas ofertas encontradas como Spam y guardarlas en el Repositorio de Spam para su posterior revisión/publicación. En este último también se almacena toda la evidencia encontrada por el analizador. La componente Publicador de Informe obtiene la información del Repositorio de spam para generar informes sobre las ofertas descartadas y el motivo de las mismas. Se nos pide que el mismo se publique todos los martes (Ver atributo de auditabilidad 10). El Subsistema de Manejo de Spam es el punto de acceso a estos datos para la interfaz de administrador.

4.9. Subsistema de Obtención de Datos

En el siguiente diagrama representa el Subsistema de Obtención de Datos (SOD) el cual se encarga de la interacción con los diversos proveedores de ofertas para alimentar el sistema con estas.

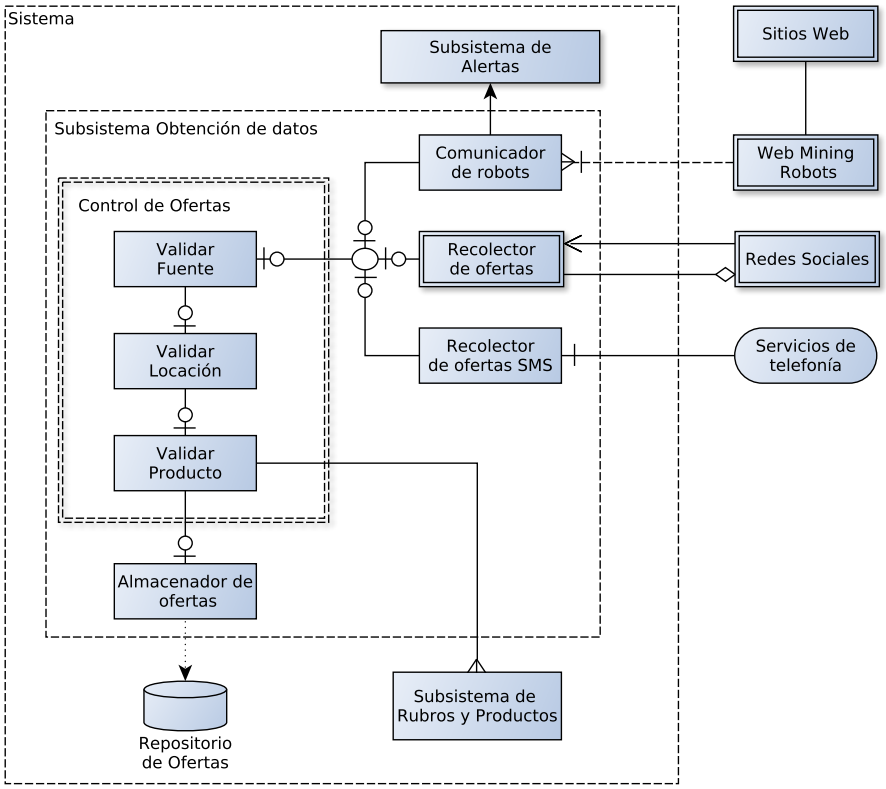

Figura 12: Diagrama arquitectónico con el detalle del Subsistema de Obtención de Datos.

El SOD se encarga de recolectar datos vía SMS, redes sociales y Web Mining. Cada una de estas fuentes de datos tiene un componente responsable de interactuar con sus respectivas fuentes de información y colocar las posibles ofertas en una manera genérica para que puedan ser validadas por la componente Control de Ofertas. Esta última componente recibe en forma de *Pipe* la información de las tres componentes encargadas de la interacción con el exterior(Comunicador de robots, Recolector de ofertas y Recolector de ofertas SMS).

La componente Control de Ofertas se encarga de verificar si la oferta es aceptada como válida por el sistema. Como esto puede llegar a ser mucho más costoso computacionalmente que la recolección de datos, se tiene múltiples instancias de esta componente para poder soportar *concurrencia*. Para los chequeos usa el estilo *pipes and filters* para que sea fácilmente *Modificable* en el futuro, de acuerdo a lo especificado en el atributo de extensibilidad 11. Una vez que se

hacen todos las validaciones pertinentes esta componente le pasa la oferta al Almacenador de Ofertas para que la guarde en el Repositorio de Ofertas.

Los Web Mining Robots son un conjunto de componentes, cada uno asignado a un sitio web específico, que leen periódicamente su contenido con el fin de encontrar nuevas ofertas, para luego enviarlas al Comunicador de Robots. Esto otorga una gran flexibilidad para poder eliminar/modificar/agregar fácilmente nuevos robots, debido a que son completamente independientes entre ellos (esto posibilita cumplir con el atributo de *Modificabilidad* número 13). En caso de que un robot incapaz de entender la estructura del sitio que releva, debido a que el mismo cambió recientemente (o si detectan grandes cambios y podría haber peligro de estar entendiendo mal las ofertas), el robot se detiene y manda una alerta avisando lo sucedido al Comunicador de Robots; este último le avisa al Subsistema de Alertas para darles aviso a los Admins (esto corresponde al atributo número 14 de *detección y prevención de fallas*).

4.10. Deployment

En la figura 13 presentamos un diagrama de alocación de recursos a distintos procesos de nuestro sistema. En particular, en este diagrama nos centramos en la alocación de recursos para el módulo que obtiene datos de ofertas.

Se espera que los componentes de control de ofertas, encargados de determinar si la información obtenida es realmente una oferta, se ejecuten en varias máquinas. Notemos que las instancias para determinar si cierta información es realmente una oferta válida son completamente independientes entre sí. Se pensó de esta manera con la idea de que el sistema pueda soportar la gran carga de trabajo que supone analizar la información obtenida. Además, de aumentar esta carga con el tiempo, resulta fácil escalar horizontalmente la performance del sistema, agregando más equipos. Sería deseable en un futuro poder manejar de forma elástica las instancias asignadas a la tarea de analizar esta información.

Con el objetivo de *alimentar al sistema con la mas fresca información de Internet*, se pensaron las fuentes de datos de forma que cada una se controle de forma independiente de las demás. Estas no necesariamente deben correr en máquinas separadas. No se implementó ningún tipo de redundancia sobre estos procesos, debido a que no consideramos como riesgosa la situación en la que uno o varios de estos equipos fallen durante un periodo corto de tiempo. Debido a la independencia entre estos procesos, resulta simple agregar nuevas instancias encargadas de obtener datos de otros medios, sin afectar a los demás componentes obtenedores de datos.

Una vez que los datos fueron identificados como ofertas válidas, estas se serializan en la base de datos de ofertas. Para poder soportar las grandes cargas de consultas, esta base de datos se encuentra implementada con un motor NOSQL. La replicación de los datos es manejada directamente por el software de la base de datos. De esta forma, podemos garantizar alta disponibilidad de los datos, y performance a la hora de responder las consultas.

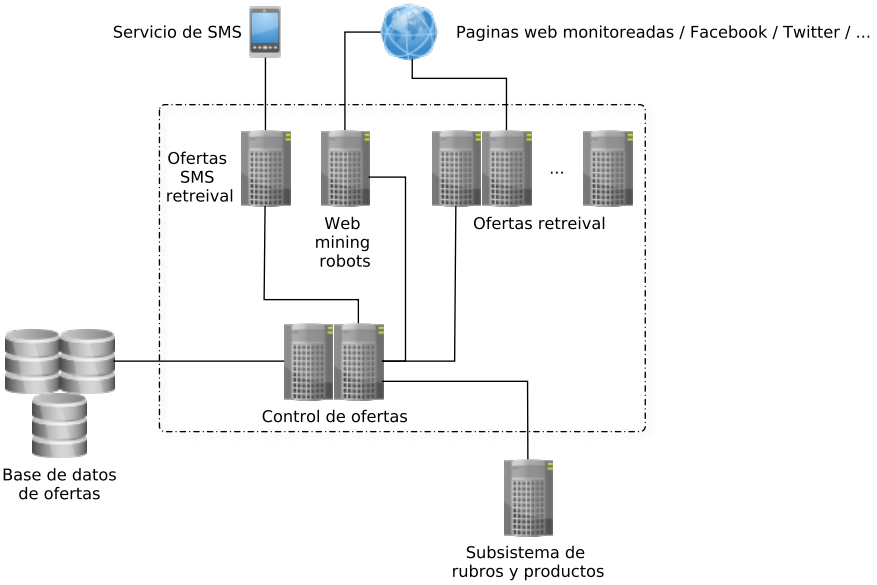

Figura 13: Diagrama de deployment del Subsistema de obtención de datos.

En la figura 14 se detalla el viewtype de alocación del subconjunto del sistema que se encarga de responder a las queries de los distintos usuarios.

Para poder manejar la cantidad de consultas provenientes de los distintos usuarios, distintos tipos de manejadores de consultas son instanciados en varias máquinas.

En particular, dado el volumen de consultas esperado, proveniente de la interfaz web y de la interfaz móvil, el manejador de pedidos de aplicación se encuentra instanciado en varias máquinas. Esto permite manejar una gran cantidad de consultas, y provee redundancia en caso de falla de uno de los servidores, para garantizar la disponibilidad del servicio. De forma similar se encuentran desplegadas varias instancias del manejador de pedidos de la API pública. Estos servicios fueron diseñados con el objetivo de poder escalar horizontalmente sin mayores complicaciones.

El subsistema de queries es el componente del sistema que recibe una mayor cantidad de consultas. Con el objetivo de poder lidiar con la carga de consultas, y dado que cada consulta es independiente de las demás, resulta natural distribuir esta carga de trabajo en un gran número de servidores. Es deseable que, en un futuro, estos puedan ser desplegados de forma elástica, para poder adaptarse a la carga del sistema y reducir costos. Replicar este servicio en varios servidores otorga una segunda ventaja: ayuda a garantizar una mayor disponibilidad del servicio. Esto último es particularmente importante, ya que este servicio forma parte del núcleo del sistema. Este sistema se encuentra conectado directamente a la base de datos de ofertas. Es importante que esta base de datos pueda antender a la cantidad de consultas de lectura que el procesador de consultas realiza. Vale aclarar que, dado que las ofertas se representan como un conjunto desestructurado de datos, resulta natural almacenarlos en una base de datos NOSQL.

De los distintos subsistemas de rubros y productos, de publicidades, de preferencias de usuario y de reputación, podemos destacar una característica común: todos estos subsistemas poseen una base de datos propia. Dado que en estos casos los datos son estructurados, optamos por utilizar bases de datos SQL. No se espera que estos sistemas sufran una gran carga de trabajo.

Una característica a destacar de la interfaz web es su capacidad para poder ser utilizada desde los más diversos dispositivos.

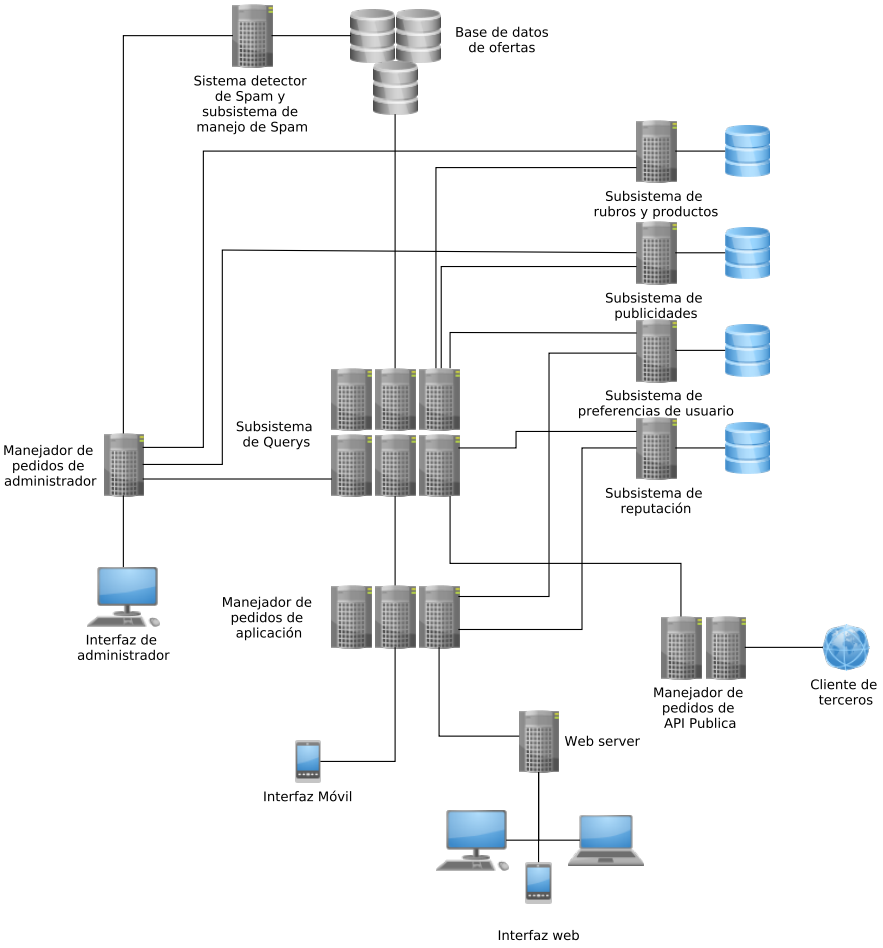

Figura 14: Diagrama de deployment del sistema central.

5. Comparación

5.1. Comparación Metodologías ágiles y Unified Process

Tanto *Unified Process* como *Scrum* son metodologías para el desarrollo de software, que proponen un marco de trabajo para planificar y controlar distintas etapas en el desarrollo de software.

Una característica común en ambas metodologías es que siguen un proceso *iterativo incremental*. Esto quiere decir que en ambas metodologías el desarrollo del software se realiza en forma de iteraciones bien definidas. Una de las principales diferencias encontradas durante la realización de los trabajos prácticos de la materia es que Unified Process organiza el desarrollo de software en distintas *etapas* (*inception, elaboration, construction* y *transition*); cada etapa se diferencia de las otras en la actividad sobre la cual se pone énfasis. Sin embargo, en cada etapa se podría realizar cada una de las posibles actividades (desarrollo, requerimientos, testing, etc...). Por otro lado, metodologías ágiles como Scrum no establecen ninguna distinción de etapas en el desarrollo.

Esto se refleja en que, para desarrollar la aplicación *Precio Justo*, se partió de una escasa planificación sobre la construcción del software a desarrollar; en cambio, para *Twitteando para ahorrar* se parte de un *plan de iteraciones* y de una *elaboración de la arquitectura* bien definida y documentada (si bien este plan y la arquitectura pueden estar sujetos a cambios).

El desarrollo en Scrum se centra en priorizar las funcionalidades que otorguen mas 'valor' al producto; esto se refleja en la idea de organizar las tareas a realizar en cada iteración en *user stories*, una serie de historias, similares a casos de uso, contadas desde el punto de vista del usuario/product owner de la aplicación. En cambio, en Unified Process, las tareas a realizar en cada iteración están son elegidas con el objetivo de reducir los riesgos. Para capturar las funcionalidades a implementar en cada iteración, Unified Process utiliza *casos de uso*.

De tener que escoger una metodología para el desarrollo de software, nosotros creemos que **no** tiene sentido elegir de forma *estricta* una única metodología. Nosotros proponemos utilizar ambas metodologías como un marco de ideas para construir una metodología que se adapte de la mejor forma a nuestro equipo y al proyecto con el cual nos enfrentemos.

Una de las diferencias que encontramos entre ambas metodologías es que Scrum pide al final de cada iteración presentar un entregable, potencialmente listo a ser usado. De esta forma, se obtiene feedback temprano de los interesados en el proyecto. En cambio en UP, no necesariamente hace falta presentar un entregable con capacidad de funcionar.

De ambas metodologías, las características que mas nos gustaron son:

- El desarrollo **iterativo incremental**: El método iterativo incremental nos permite tener 'checkpoints' bien definidos, distribuidos mas o menos uniformemente en la línea de tiempo. Esto facilita el establecer objetivos chicos, alcanzables en el corto plazo, que ayudan a mantener al equipo focalizado. Además ayudan a obtener feedback de los interesados tempranamente. Esto ultimo es especialmente importante para descubrir si las expectativas de los interesados eran distintas a las del equipo de desarrollo, y permite tomar acciones rápidamente, para contrarrestar esta situación.

- De Scrum, la idea de **al final de cada iteración presentar un entregable**, para obtener feedback temprano de los usuarios. Sin embargo, en algunos casos es necesario ser flexible con esta condición, en especial para proyectos grandes o que tienen poca interacción con el usuario, en las primeras iteraciones del proyecto.
- De Unified Process, el separar el desarrollo del sistema en **etapas**. Debido a que es esperable que en las primeras iteraciones, los esfuerzos se concentren en obtener requerimientos y planificar el proyecto, y que una vez que el proyecto se encuentra mas maduro, el foco se centre en programación, testing e integración.
- Las ideas de organizar el desarrollo en funcionalidades, representadas con *user stories*, y decidir cuando implementar cada funcionalidad de acuerdo a una combinación de **business values** y minimización de **riesgos**. De esta forma, se intenta satisfacer cuanto antes los intereses de los stakeholders, y se intenta obtener feedback de ellos cuanto antes y al mismo tiempo, se tienen en cuenta los riesgos, que tienden a afectar fuertemente la arquitectura del sistema y de no tenerse en cuenta de forma temprana, podrían poner en riesgo el proyecto, o los tiempos planeados para completarlo.
- La elaboración detallada de la **arquitectura** como uno de los puntos centrales de la metodología. Para, desde el inicio del proyecto, planificar el software para poder contrarrestar *riesgos* y satisfacer los *atributos de calidad* que restringen al proyecto. Además, el desarrollo de la arquitectura suele dividir al sistema en subsistemas mas pequeños, de los cuales es esperable que uno o dos desarrolladores puedan completar. Esto ayuda a la hora de asignar trabajo a los distintos miembros del equipo de desarrollo.
- De las metodologías ágiles, una de sus pautas claves, el **aceptar cambios** en los requerimientos e intentar adaptarse rápidamente a las circunstancias cambiantes que van surgiendo a lo largo del desarrollo de un proyecto.
- De Scrum, las **stand up meetings**, que ayudan a los distintos miembros del equipo a conocer en qué componentes trabajan sus pares y permite comprender el estado de avance de la iteración en detalle. Esto favorece la sincronización de los distintos miembros del equipo.

5.2. Comparación 'programming in the small' y 'programming in the large'

Programming in the large y *Programming in the small* se utilizan para describir distintos enfoques al desarrollo de software.

Típicamente el término *Programming in the small* se utiliza para describir el desarrollo de un sistema compuesto por una cantidad relativamente pequeña de subsistemas que interactúan entre si. Estos sistemas suelen desarrollarse por equipos pequeños en una cantidad corta de tiempo.

Por otra parte, *Programming in the large* hace referencia al desarrollo de sistemas, compuestos por varios subsistemas. De forma que el proyecto es realizado por un gran numero de equipos, o por un equipo en una cantidad grande de tiempo.

De esta forma, podríamos pensar a *Programming in the large* como el desarrollar grandes proyectos de software y a *Programming in the small* como el desarrollo de cada uno de sus subcomponentes mas pequeños, con tareas simples y bien definidas.

La principal diferencia que pudimos encontrar entre ambas es la rigurosidad con la cual se planifica cada una. En *Programming in the small* , la planificación resulta mínima; esto se debe en particular a que se desarrollan módulos pequeños en un período corto de tiempo. En cambio, en *Programming in the large* , el diseño del sistema surge de un análisis mas profundo de la problemática. *Programming in the large* no solo maneja proyectos mas grandes, sino que un error de diseño en un la arquitectura del sistema puede requerir cambios en varios módulos, que requieran una gran cantidad de horas/hombre. Esto ultimo se contrasta con un error de diseño introducido en un único módulo, donde los cambios están aislados.

6. Casos de uso

6.0.1. CU1: Obteniendo informacion de Internet

Descripción: El sistema colecta información sobre ofertas de distintos medios (redes sociales, sitios de información en Internet, SMS), la procesa para extraer los datos relevantes de la oferta, y la almacena para luego ser provista a los usuarios.

6.0.2. CU2: Almacenando datos obtenidos de distintos sitios de Internet

Descripción: El sistema recorre distintos sitios de Internet y consulta distintos servicios obteniendo ofertas, y las almacena internamente en el sistema para poder ser accedidos después por los mecanismos que presentan la información a los usuarios de la aplicación.

6.0.3. CU3: Se consulta información a través de una API pública

Descripción: Clientes externos pueden consultar a nuestro sistema por ofertas almacenadas, a través de un servicio público (API) ofrecido por nosotros.

6.0.4. CU4: El usuario consulta por un producto determinado

Descripción: El usuario consulta por un producto válido A de algún rubro, y recibe como respuesta ofertas que le indican dónde comprarlo y a qué precio.

Ruta alternativa: El producto A no es válido; se le informa al usuario que no existe información para dicho producto.

6.0.5. CU5: El usuario consulta precios a través de una interfaz amigable en su computadora

Descripción: El usuario accede a un sitio web propio de *Twitteando para ahorrar*, a través del cual puede consultar por precios para distintos productos.

6.0.6. CU6: El usuario consulta precios a través de una interfaz amigable en su celular

Descripción: El usuario accede a una aplicación de celular propia de *Twitteando para ahorrar*, a través de la cual puede consultar por precios para distintos productos.

6.0.7. CU7: Se realiza una consulta por un producto desde un dispositivo sin conexión

Descripción: El usuario accede a la aplicación de celular *Twitteando para ahorrar* sin tener conectividad a Internet y realiza una consulta. Como respuesta recibe las ofertas relevantes al producto buscado. La información provista al usuario podría estar limitada respecto de lo que vería si tuviera conectividad, pero debe ser satisfactoria dentro de lo posible. Se le notifica al usuario que está trabajando en un modo "sin conexión".

6.0.8. CU8: Se anticipa a los deseos del usuario y se muestra informacion preliminar

Descripción: Al mismo tiempo que el usuario comienza a ingresar una consulta, la aplicación se anticipa a los deseos del usuario para mostrarle rápidamente precios de productos que podrían responder a la consulta que se está formulando.

6.0.9. CU9: Se realiza una consulta por un producto sustituible

Descripción: El usuario consulta por un producto A dentro de los habilitados en algún rubro y existe una regla de sustitución que indica que A es sustituible por el producto B. Luego el usuario recibe información de ofertas de A y también ofertas de B.

6.0.10. CU10: Se realiza una consulta por un producto que tiene productos asociados

Descripción: El usuario consulta por un producto A dentro de los habilitados en algún rubro y existe una regla de asociación que indica que A se relaciona con el producto B. Luego el usuario recibe información de ofertas de A y también ofertas de B.

6.0.11. CU11: Mostrando publicidades

Descripción: Cuando el usuario utiliza la aplicación, visualiza, además de los resultados de su consulta, propaganda de los sponsors de *Twitteando para ahorrar*.

6.0.12. CU12: ABM de rubros habilitados

Descripción: Los administradores del sistema pueden acceder al mismo para agregar nuevos rubros, o modificar o borrar existentes.

6.0.13. CU13: ABM de productos en un rubro

Descripción: Los administradores del sistema pueden acceder al mismo para agregar nuevos productos, o modificar o borrar existentes. También pueden redefinir la pertenencia de un producto a uno o más rubros.

6.0.14. CU14: El usuario se autentica con el sistema

Descripción: El usuario de la aplicación puede utilizar alguna cuenta de un servicio asociado con OpenID (Google, Yahoo, Facebook y otros) para autenticarse en la aplicación. A partir de ese momento la aplicación sabe quién es el usuario y puede utilizar la información que tiene del mismo para proveerle funcionalidades más avanzadas.

6.0.15. CU15: Un usuario autenticado modifica sus preferencias de confianza

Descripción: El usuario autenticado accede a una sección de preferencias de usuario, y determina sus niveles de confianza para cada tipo de fuente de las ofertas. Los tipos de fuente pueden depender de la información de redes sociales asociada a la cuenta.

6.0.16. CU16: Un usuario realiza una consulta por un producto determinado estando autenticado

Descripción: Las ofertas que devuelve el sistema ante la consulta estarán ordenadas según las prefeencias establecidas por el usuario. Se mostrarán primero aquellas cuya fuente haya declarado como más confiables, luego las de fuentes con menor confianza. No se mostrará ninguna oferta cuya fuente esté declarada como no confiable.

6.0.17. CU17: Detectando ofertas falsas

Descripción: Al recopilar datos de precios en Internet, el sistema es capaz de detectar si la información es sospechosa y marcarla como tal, para futura revisión. Además el sistema recopila todas las evidencias encontradas para sospechar de los datos.

6.0.18. CU18: Siendo martes se publica un informe de ofertas falsas

Descripción: Cada martes el sistema arma y publica un informe con los productos sobre los cuales se encontraron precios dudosos junto con la evidencia que genera la sospecha. Este informe debe estar disponible para revisión por usuarios externos selectos.

6.0.19. CU19: Un usuario autenticado puede votar por la validez de una oferta

Descripción: Un usuario ya autenticado en el sistema elige una oferta y la marca como válida o inválida. Esto afecta la reputación del usuario o fuente que dio origen a la oferta para facilitar la detección de ofertas sospechosas.