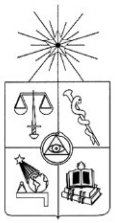

UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IMPLEMENTACIÓN DE UNA API DE INFORMACIÓN BANCARIA PARA LA SBIF

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO *CIVIL EN COMPUTACIÓN*

Rodrigo Alejandro González Orellana

PROFESOR GUÍA:
JOSÉ ALBERTO PINO URTUBIA

MIEMBROS DE LA COMISION:
ALEJANDRO HEVIA ANGULO
JOSÉ ANDRÉS BENGURIA DONOSO

SANTIAGO DE CHILE
ENERO 2010

Resumen Ejecutivo

Debido al avance en tecnologías de información y telecomunicaciones, la tendencia a nivel mundial en sistemas de gobierno es migrar desde una democracia representativa a una democracia participativa y colaborativa. Esto es lo que se llama un Gobierno Abierto. Países como EEUU, Australia, Reino Unido, Finlandia y Suecia han mostrado avances significativos en esta materia.

Para que el Gobierno Abierto sea posible son necesarias varias cosas: voluntad y decisión política, apertura de datos públicos en formatos abiertos y estrategias de colaboración y participación ciudadana. Suponiendo que existe en Chile voluntad y decisión política, el paso siguiente es la apertura de datos públicos en formatos abiertos.

En un afán de replicar la experiencia internacional y siguiendo la misión de alcanzar un Gobierno Abierto, nace el presente trabajo de memoria. Este trabajo consiste en la implementación y puesta en funcionamiento de una API de información financiera para la Superintendencia de Bancos e Instituciones Financieras (SBIF).

Una API es una interfaz de operación que permite la interacción entre diferentes partes de un sistema de software. En el caso especifico de la API implementada para la SBIF, es un canal difusor de información pública en formatos abiertos y estándar, para el uso por parte de agentes externos.

La información disponible por medio de la API corresponde a información pública que se encuentra disponible en el actual sitio web de la Superintendencia de Bancos e Instituciones Financieras. Esta información se compone de recursos como indicadores financieros, balances de instituciones financieras, etc.

Para la implementación de la API se usó la técnica de ingeniería de software REST, la cual hace uso sólo del protocolo HTTP para la entrega de recursos. Esto hace que operacionalmente la API sea eficiente en el uso de recursos y accesible a un amplio espectro de clientes.

Índice

Índice de figuras

1. Agradecimientos

Es mi intención en estas líneas agradecer el apoyo a todos quienes de forma directa o indirecta lograron que este trabajo llegue a buen puerto. En particular quiero destacar el apoyo de familiares y amigos, quienes con sus incondicional apoyo brindaron fuerza y entusiasmo constante.

Muy importante también es agradecer a mi profesor guía José Pino, quien en el momento más crítico de la memoria tuvo la sabiduría para dar el consejo más adecuado.

Finalmente parte importante de este logro está dedicado a la Unidad de Internet y Publicaciones de la SBIF, quienes siempre tuvieron la mejor disposición para apoyar el trabajo.

2. Introducción

La Superintendencia de Bancos e Instituciones Financieras (SBIF) fue creada en 1925 y es una institución pública y autónoma.

El jefe superior de la SBIF es el Superintendente, quien es nombrado por el Presidente de la República.

La misión de la SBIF es supervisar las empresas bancarias y otras instituciones financieras, en resguardo de los depositantes, otros acreedores y del interés público.

Una de las formas en las cuales se canaliza actualmente la misión de la SBIF es la difusión. El objetivo es crear usuarios del sistema financiero cada vez más capacitados que sean aptos para actuar a conciencia.

Dadas las tendencias tecnológicas actuales en arquitecturas de aplicaciones web y en respuesta a la labor difusora de la SBIF, el señor Juan Carlos Camus, Jefe de la Unidad de Internet y Publicaciones de la SBIF, sintió la inquietud de disponibilizar parte de la información contenida en las bases de datos de la SBIF. El camino elegido fué orientar el actual desarrollo informático de la SBIF hacia una arquitectura orientada a servicios (SOA)[1].

Sumado a lo anterior, el lunes 11 de agosto del 2008, la entonces Presidenta de la República, Michelle Bachelet, promulgó en Chile la Ley sobre Transparencia de la Función Pública y Acceso a la Información de los órganos de la Administración del Estado. Como resultado de la promulgación de esta Ley, un gran volumen de antecedentes de todos los organismos del Estado son transformados en públicos.

Bajo este contexto, se inició en la SBIF un proceso progresivo de cambio en los desarrollos existentes y futuros. En el año 2009 se lanza el proyecto de implementación de un API[1] en base a servicios web para información bancaria, el cual inicialmente contempla etapas de desarrollo hasta el año 2011. Actualmente, como parte del desarrollo del proyecto, la SBIF cuenta con un servicio web implementado que permite dentro de la red interna, obtener información de los balances de los bancos fiscalizados por la SBIF.

Etapas posteriores del proyecto, dentro de las cuales en enmarca el desarrollo de esta memoria, contemplan la implementación de una API en base servicios web. Esta API está destinada a proveer información diversa contenida en las bases de datos de la SBIF, usando una técnica de arquitectura software llamada REST, a instituciones o particulares externos que la requieran para sus desarrollos.

[1] Una API es una interfaz de operación que permite la interacción entre diferentes partes de un sistema de software.

3. Antecedentes

3.1. Conceptos básicos

- API: Una API es una abstracción que describe una interfaz para la interacción con un conjunto de funciones utilizadas por los componentes de un sistema de software . El software que proporciona las funciones descritas por un API se dice que es una implementación de la API.
- Servicio Web: Sistema de software diseado para permitir interoperatibidad máquina a máquina en una red.
- SOA: Arquitectura orientada a servicios. Se trata esencialmente de un set de servicios sueltos, donde cada uno es relativamente económico de construir o reemplazar si es necesario. Al ser independientes, el poder unirlos permite a SOA adaptar cambios, cuestión imposible para arquitecturas tradicionales. En la Arquitectura Orientada a Servicios, se puede reemplazar un servicio sin tener que preocuparse por la tecnología fundamental; la interfaz es lo que importa, y está definida en un estándar universal en servicios Web y XML.
- SOAP: Protocolo de comunicación basado en XML, para invocar procedimientos en forma remota (Remote Procedure Call). Utiliza cualquier protocolo que permita transportar mensajes de texto, siendo HTTP el más utilizado. Su especificación describe el contenido que debe tener un mensaje XML para ser usado en una invocación remota. Cualquier aplicación que cumpla la especificación puede invocar y proveer servicios sin importar en que lenguaje o plataforma esté implementado, lo único necesario es que la aplicación sea capaz de interpretar el mensaje SOAP que recibe, realizar el procesamiento que el mensaje requiera, y devolver otro mensaje SOAP con la respuesta.
- REST: (REpresentation State Transfer). Estilo de arquitectura de software para sistemas hipermediales distribuidos tales como la Web. REST se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen cómo se define y disecciona los recusos. El término es utilizado frecuentemente en el sentido de describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional, como hace SOAP.
- Open Data: Filosofía y práctica que persigue que determinados datos estén disponibles de forma libre a todo el mundo, sin restricciones de copyright, patentes u otros mecanismos de control.

4. Especificación del problema

La doctrina política de Gobierno abierto[2] sostiene que los temas de gobierno y administración pública deben ser abiertos a todos los niveles posibles en cuanto a transparencia para conseguir una mayor participación ciudadana y una mejor regulación.

El Gobierno abierto de basa en tres conceptos:

- La transparencia
- La rendición de cuentas
- La colaboración y participación ciudadana en asuntos públicos

Este concepto es lo que muchas personas denominan la transición natural desde una democracia representativa a una democracia participativa e inclusiva.

Sin embargo, para que el concepto de Gobierno abierto pueda ser puesto en práctica, es necesario que estén presentes tres factores muy importantes:

- Voluntad y decisión política
- La disponibilización de datos públicos en formatos estándar
- La estrategia de colaboración y participación ciudadana en asuntos públicos

La voluntad política de liberar datos públicos en formatos estándar y reutilizables por terceras personas reconoce la capacidad creativa e innovadora de la ciudadanía para generar nuevas colecciones de datos provenientes de diferentes fuentes, los cuales pueden dar visiones matizadas de la información, útiles al momento de generar nuevas actividades económicas y/o tomar mejores decisiones estratégicas. Para que esto sea posible, es necesario que exista como un objetivo intermedio, un determinado grado de interoperabilidad de la información que es liberada desde las diferentes fuentes.

Países como EEUU, Australia, Reino Unido, Finlandia o Suecia ya han realizado la apuesta. En España, Asturias, Cataluña y el País Vasco lideran el movimiento, junto a los Ayuntamientos de Gijón, Barcelona y Zaragoza.

Actualmente Chile se encuentra avanzando en el camino hacia un Gobierno Abierto. El 20 de abril de 2009 entró en vigencia la Ley de Transparencia, un hito histórico para el sistema administrativo nacional y un paso importante dado por Chile para profundizar su democracia y garantizar el ejercicio transparente de la acción gubernamental. Sin embargo, esto significa el cumplimiento de solo uno de los tres conceptos básicos de un Gobierno Abierto, por lo cual se torna urgente abordar los conceptos de rendición de cuentas y colaboración y participación ciudadana.

4.1. Antecedentes de la detección del problema

El sitio web de la SBIF tiene en sus publicaciones, datos actuales e históricos de diversa naturaleza. Algunos de estos datos son de propiedad de la SBIF, como los balances de las diferentes instituciones financieras, mientras otros no son propiedad de la SBIF pero se obtienen de sus fuentes propietarias, como por ejemplo, los valores de los indicadores financieros (IPC, UTM, etc.), los cuales

[2] http://es.wikipedia.org/wiki/Gobierno_abierto

son propiedad del Banco Central. La forma en la cual los datos son ofrecidos en el sitio web es como información contenida dentro del mismo código HTML, como archivos CSV, como archivos de extensión emph.xls (Formato de Microsoft Excel) o simplemente como archivos PDF.

Bajo este contexto, hubo síntomas determinantes que llevaron a la identificación del problema anteriormente descrito, los cuales se expresaron como necesidades de desarrolladores externos que no estaban cubiertas. Estos síntomas fueron:

- Muchas personas como analistas financieros, diseñadores web, etc. se comunicaban continuamente mediante el e-mail de contacto del sitio web, solicitando si era posible que la información les fuese provista de alguna forma estándar, como un archivo XML o algún CSV en caso que el sitio no provea esa opción.
- Muchos desarrolladores externos de visualización de información, han señalado continuamente en diferentes seminarios y jornadas de trabajo que en ausencia de las herramientas adecuadas, la forma en la cual obtienen datos públicos como el valor del Dolar, el Euro, etc. es mediante un script que parsea sitios web que publican dicha información.

Gracias a la manifestación de estos síntomas, fue posible llegar en una etapa temprana del proyecto a determinar que no solo es importante que los datos sean publicados (Transparencia), sino que para lograr un Gobierno abierto participativo y colaborativo, era también importante la forma en la cual la información es provista.

4.2. Primera propuesta para solucionar el problema

Previamente ya teníamos identificado que los 3 requisitos fundamentales para la existencia de un Gobierno abierto son:

- Voluntad y decisión política
- Disponibilización de datos públicos en formatos estándar
- Estrategia de colaboración y participación ciudadana en asuntos públicos

Suponiendo que existe la voluntad política de migrar hacia un Gobierno abierto, nuestra contribución a solucionar el problema va de la mano de disponibilizar los datos públicos existentes y crear una estrategia de integración ciudadana. En este escenario, fue natural pensar que la solución local para el caso de SBIF es diseñar un sistema que permita que usuarios o máquinas puedan acceder a la información pública contenida en las bases de datos de la SBIF.

En esta línea, actualmente la SBIF cuenta con un servicio web de prueba implementado usando SOA. Este servicio web entrega información referente a los balances de los bancos fiscalizados por la SBIF, usando XML como documentos de intercambio basados en el protocolo SOAP. El uso que se da a este servicio web está restringido al buscador interactivo existente actualmente en la página de la SBIF.

Hoy en día, SOA surge como una de las tendencias naturales para las arquitecturas web a nivel empresarial[2], lo cual hace del tema un tópico vigente e innovador en nuestro país. Uno de los factores que la hace treméndamente atractiva es que considera el uso de las aplicaciones ya existentes. Como resultado, etapas previas del proyecto contemplaron el desarrollo de servicios web usando SOA.

Dentro de las ventajas que presenta una implementación usando SOA tenemos:

- Interoperabilidad multi-proveedor.
- Reutilización de código y componentes.
- Agilidad en la aplicación de cambios.
- Aplicaciones compuestas.
- Arquitecturas altamente distribuidas.
- Control de la calidad de servicio.

Sin embargo SOA también presenta una gran desventaja. Dado que la comunicación entre el proveedor de datos y los consumidores está regida por contratos, cambios en los atributos de los contratos generan rupturas en la comunicación afectando la interoperabilidad. Esto genera gran rigidez en las aplicaciones, con lo cual se requiere un cambio profundo en las organizaciones con un alto esfuerzo, no siendo sencillo para la mayoría de las organizaciones de adoptar.

En este punto es donde se consideró posteriormente la implementación de una arquitectura orientada a recursos donde los datos se entregan usando REST, esto se debe a que el uso del contrato sólo es considerado al momento de iniciar la comunicación. Esto nos conduce a una aplicación mucho más flexible que genera menos resistencia al cambio dentro de la organización favoreciendo la interoperabilidad de la información.

Las ventajas que tiene para este caso el uso de REST sobre SOA son básicamente:

- REST ofrece recursos y representaciones variadas, por ejemplo, xml, html, text, pdf, binarios, etc. Además permite la utilización de links entre diversos recursos. Por el contrario en SOA los servicios web solo manejan XML como documentos de intercambio basados en el protocolo SOAP.
- Los servicios web en SOA no están focalizados en recursos, sino en operaciones.
- REST depende directamente del protocolo HTTP (Hypertext Transfer Protocol), este expone siempre la misma interface a todos los consumidores, mediante operaciones bien definidas: POST, GET, PUT y DELETE. Desde este punto de vista, REST es mucho más escalable que los servicios web SOA, donde tenemos diversas interfaces, es decir, diversas operaciones.
- REST, expone sus recursos a través de un URI (Uniform Resource Identifier); y permite cambiar el estado de un recurso a través de operaciones genéricas para todos los consumidores.

Considerando lo anterior, se contempló en primera instancia de solución el desarrollo de una API de nuevos servicios web capaces de entregar información bancaria usando REST.

4.3. Situación previa a la solución

En este apartado se hace un levantamiento del contexto en el cual se desarrolla software, haciendo uso de información pública de la SBIF, tanto dentro como fuera de la misma SBIF.

4.3.1. Desarrollos de software dentro de la SBIF

El desarrollo de aplicaciones web dentro de la SBIF es un desarrollo que se enmarca dentro de una arquitectura especial, la cual fué diseñada a medida, y comenzó a operar en abril del año 2004.

La materialización de está arquitectura esta desarrollada en Java usando servlets. Sigue un modelo MVC[3], donde los diferentes controladores son los encargados de consultar a la base datos previa solicitud. Cualquier nuevo desarrollo dentro de la SBIF, está contemplado como una extensión de este gran proyecto.

Para las vistas de la aplicación se usa XSLT[4]. Esto implica que los controladores de la aplicación web entregan archivos XML como respuesta.

Una de las implicancias directas de una gran aplicación implementada de esta forma, es que la curva de aprendizaje para un nuevo desarrollador que se integre al equipo es bastante empinada, dado que la arquitectura en sí no es sencilla de comprender.

También es importante hacer notar que el uso de XSLT y XML para las vistas de la aplicación involucra necesariamente rigidez y restricción al momento de usar librerías javascript para visualización de información.

4.3.2. Desarrollos de software fuera de la SBIF

Si bien el desarrollo fuera de la SBIF tiene muchos matices, hay líneas generales que son recurrentes en muchas implementaciones. De forma bastante general, una aplicación que usa datos de la SBIF los obtiene directamente del sitio web como se muestra en la Figura 1.

[3]Modelo Vista Controlalor. http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

[4]Extensible Stylesheet Language Transformations. http://en.wikipedia.org/wiki/XSLT

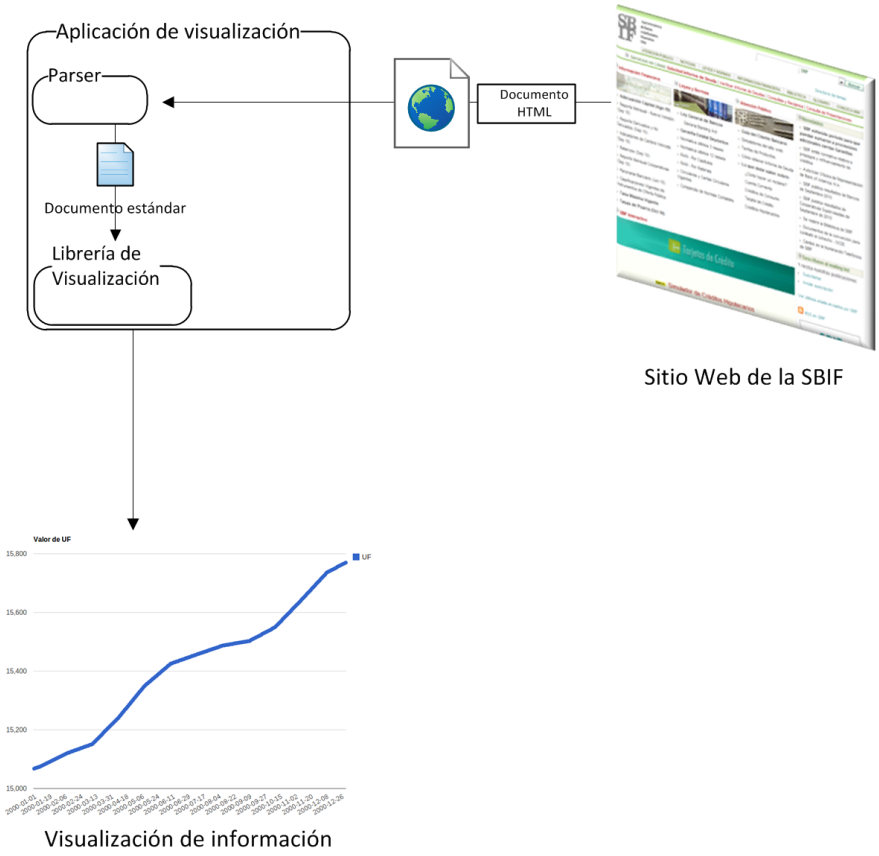

Figura 1: Diagrama de la arquitectura lógica de los desarrollos externos antes de implantada la solución

La estrategia más común es usar un parser directamente sobre el código HTML del sitio web. Una vez obtenidos los datos de esta forma, estos son transformados en un archivo estándar, como un XML o un JSON. Finalmente, este documento generado es entregado como un parámetro a alguna librería de visualización, por ejemplo, el API de visualización de Google.

El formato más usado es JSON, dado que puede ser directamente manejado por el código Javascript y pasado a alguna librería también Javascript.

El recurso más popular es el valor de la UF. Se estima que existe un número no menor de desarrolladores que obtiene este valor de la forma anteriormente descrita.

4.4. Justificación de la existencia de una solución al problema

Una de las tareas que la SBIF ha llevado a cabo durante los últimos años es ofrecer transparencia de la información que recibe y publica, entregando datos para apoyar la toma de decisiones en el mercado. En este sentido, las tecnologías de información han sido las aliadas naturales en el proceso de ofrecer dicha información.

El desarrollo de esta memoria buscó seguir avanzando por este camino a través de la incorporación de nuevas plataformas difusoras de información.

Cabe resaltar que en el país este tipo de proyectos aún no tiene desarrollos públicos conocidos en el ámbito financiero, ya que sólo se conocen sistemas que utilizan servicios web para visualización de mapas y otras aplicaciones similares.

En el mundo en tanto, hay cada vez más desarrollos de este tipo destacando ámpliamente los realizados por el Banco Mundial a través de su API pública .

Dado que actualmente existe una gran atención en los desarrollos de la SBIF y esperando que la tendencia se replique, la implementación de la memoria representa un paso importante en el actual desarrollo de aplicaciones web, no solo para la misma organización, sino también para otras organizaciones públicas y privadas a nivel nacional. De esta forma se busca seguir avanzando en el camino para llegar a un *Gobierno Abierto*, participativo y colaborativo.

4.5. Propuesta de valor al solucionar el problema

En este apartado se mencionan los beneficios o propuestas de valor que se generan luego de la puesta en marcha de una solución al problema. Para abarcar todos los posibles beneficios, se efectúa un análisis que involucra tres puntos de vista que van desde el interior de la de misma organización hasta la sociedad en general.

4.5.1. Propuesta de valor a la SBIF

Las propuestas de valor dentro de la SBIF abarcan diferentes ámbitos y todos muy variados.

Técnicamente la API se convertiría en una herramienta que facilita los desarrollos de software al interior de la SBIF, convirtiéndose en una interface estándar de comunicación con el motor de base de datos. Además, se espera que la API sea un nexo entre la información contenida en las bases de datos y la actuales opciones de visualización de información disponibles en la red. Este nexo permitiría dar dinamismo al actual sitio web de la SBIF, haciéndolo más atractivo y generando que más gente lo visite cada día.

Sumado a lo anterior y dado el carácter innovador de una aplicación así en el sector publico, llevar a cabo este proyecto sitúa a la SBIF como la institución pionera en este tipo de desarrollos, la cual en un futuro será la encargada de marcar la pauta en estas materias.

Finalmente el desarrollo del proyecto se enmarca dentro de las metas institucionales contempladas por la SBIF para el año 2010, por lo cual su ejecución trae consigo una evaluación positiva de la institución dentro de los demás servicios del Estado.

4.5.2. Propuesta de valor a la comunidad de desarrollo externa

Dado que los desarrolladores externos son el nicho donde apunta nuestra solución (más que la propia SBIF), son ellos quienes se verían tremenda y directamente beneficiados con la implementación del API.

Una primera propuesta de valor para desarrolladores externos pasa por la propiedad de datos que antes no tenían disponibles de forma fácil. Si bien todo dato provisto por la API existiría también en el sitio web, no todos esos datos son manipulables. Esto los transforma en información con un alto costo de procesamiento, resultando poco viable un desarrollo externo sobre los mismos.

También hay que considerar que no sólo agregaría valor la flexibilidad de la representación de los datos, sino también la facilidad de acceso y la robustez que el uso de la API agrega a las nuevas aplicaciones. Por ejemplo, una aplicación que obtiene datos parseando un archivo HTML falla inmediatamente en el momento que en la SBIF cambia la estructura del sitio web.

4.5.3. Propuesta de valor a la sociedad

Si bien la puesta en marcha de la API no marca una gran diferencia por sí sola cuando nos referimos a la sociedad, se espera que sea una pequeño aporte que marque la pauta para que otras organizaciones públicas repliquen la experiencia. Este proyecto es un paso más hacia la construcción de un *Gobierno Abierto* que hace participe a la sociedad y fomenta la actividad colaborativa.

La propuesta de valor asociada a un *Gobierno Abierto*, es la que se describe en la Figura 2.

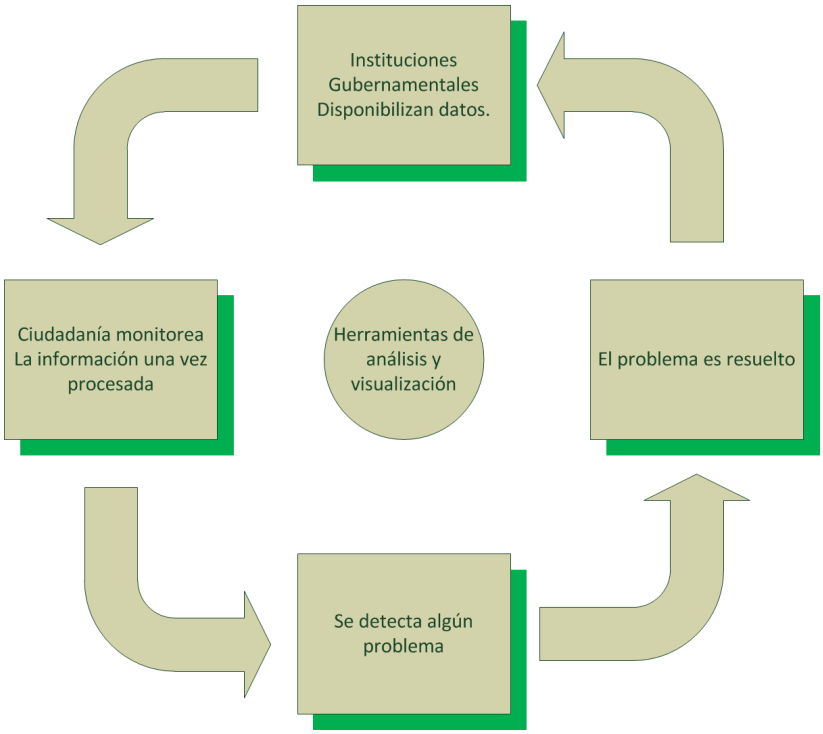

Figura 2: Diagrama de la propuesta de valor de la API de la SBIF para la sociedad.

Como primer paso tenemos a organizaciones publicando sus datos en formatos estándar. Esto genera que personas interesadas hagan uso de diferentes herramientas de análisis y/o visualización, reúnan datos de una o más fuentes y sean capaces de detectar los problemas que se hacen patentes. Luego esto hace que las organizaciones reciban estos reportes como retroalimentación y se encarguen de solucionar estos problemas. De esta forma el modelo de Gobierno abierto se vuelve autosustentable reportando variados beneficios a la sociedad en cada iteración.

4.6. Requisitos de una solución efectiva al problema

Para que luego de la puesta en marcha de la API el problema deje de estar presente, la solución debe cumplir con una serie de requisitos. Estos requisitos se acordaron previo inicio del desarrollo entre el memorista y la unidad de internet y publicaciones de la SBIF. Estos requisitos se detallan a continuación:

- Estandarización de la información: La API debe entregar la información en formatos estándar para lograr interoperabilidad de la información. Estos formatos pueden ser XML, XBRL, JSON, etc.
- Calidad de información: Se debe garantizar la calidad de la información entregada. Datos erróneos pueden conducir a conclusiones erróneas sobre los mismos.
- Disponibilidad: La disponibilidad de la aplicación debe ser alta, dado que si la API deja de funcionar muchas otras aplicaciones clientes también dejan de funcionar.
- Acceso: El acceso al API debe estar garantizado a cualquier miembro de la sociedad.
- Documentación: Se debe proveer documentación de calidad a desarrolladores externos respecto del uso del API como de sus modificaciones.
- Compatibilidad: Se debe asegurar la compatibilidad de aplicaciones anteriores al lanzamiento de una nueva versión del API.

4.7. Criterios de aceptación para una solución exitosa

Tanto o más importante como que la solución sea efectiva, es que la solución sea exitosa. La principal seal que refleja que la aplicación es exitosa es la aceptación de la solución por parte de los desarrolladores externos. Para llegar a un indicador de aceptación, al igual que los requisitos de efectividad, se fijó previo al desarrollo de la memoria, criterios que determinan la penetración de la solución. Estos criterios son los que se detallan a continuación:

- Relevancia de Datos: La API debe poner a disposición información que por su carácter revista interés público.
- Número de usuarios: La API debe ser una herramienta atractiva capaz de atraer nuevos usuarios.
- Actividad de usuarios: Los usuarios registrados en el API deben mantener una tasa de consultas mayor a cero.
- Retroalimentación: La aplicación debe evolucionar en base al retroalimentación de los usuarios recibida en la dirección de contacto.

4.8. Situación proyectada luego de solucionado el problema

Una vez solucionado el problema, se espera que el sitio web de la SBIF comience a tomar dinamismo en base a un nexo sólido con los datos. Esto quiere decir que el sitio web se transforme en una ventana de visualización de los datos. En ejemplo de lo anterior sería que los actuales gráficos estáticos (imágenes) sean reemplazados por gráficos dinámicos, al igual que muchas de las tablas y los simuladores de créditos.

También se espera que la puesta en marcha de este proyecto sea el puntapié inicial para futuras expansiones del sitio web. Estas expansiones están contempladas en la formulación del proyecto en el cual se enmarca la memoria, y consideran puntos como la inclusión de visualizaciones nuevas que incluyen datos que antes no estaban representados de ninguna forma en el sitio.

A nivel de desarrollos externos, se espera que mejore la calidad de las aplicaciones que previa puesta en marcha del proyecto de la API, usaban datos de la SBIF. Con mejoras de la calidad de aplicaciones se denotan avances en robustez, accesibilidad e interoperabilidad del software.

Transcurrido un tiempo del lanzamiento del API, se espera se forme una comunidad de desarrollo en torno al servicio, donde se este constantemente recibiendo e integrando retroalimentación. Esto permitirá mejorar progresivamente la calidad del servicio prestado y la experiencia de los usuarios.

5. Descripción de la solución

En esta sección se describe en detalle las partes de la solución definitiva. Más que al producto final, esta descripción está centrada en el proceso que se vivió y que desencadeó la adopción de determinadas características en la solución.

5.1. Contexto de la solución

El contexto en el cual opera la solución, es muy importante de mencionar al momento de describir la solución ya que fue determinante al momento de tomar decisiones estratégicas y de implementación que se detallan en este apartado.

5.1.1. Arquitectura física de la SBIF

El diagrama que describe la arquitectura física donde la aplicación esta operando es el que se muestra en la Figura 3. Esta arquitectura consta de cuatro partes:

- El Firewall: Encargado de dictaminar que requerimientos pueden acceder a las redes internas de la SBIF.
- El Balanceador de carga: Encargado de hacer que los servidores dentro del cluster tengan una carga de consultas similar para atender.
- El cluster de servidores: Cluster formado por 3 servidores Windows de 32 bits virtualizados que corren sobre una misma máquina real. Durante el desarrollo del proyecto se comenzó ademas un proceso de expansión donde se agregan otros 3 servidores Linux de 64 bits, pero aún no se encuentran operativos por lo cual no fueron considerados.
- La base de datos: Donde están contenidos los datos que se publican en el sitio web de la SBIF. Dentro del mismo proyecto de expansión anteriormente mencionado, se agregará otro motor de base de datos que formará cluster con el actual.

Como se observa en la Figura 3, los requerimientos de internet (LAN externa) pasan por el firewall externo. Una vez atravesado en firewall, el requerimiento pasa al balanceador de carga. El balanceador equilibra los requerimientos a los servidores web del 1 al 3 según un algoritmo propio del balanceador. Posteriormente estos requerimientos son recibidos por el servidor web Apache dentro del servidor físico (ver Figura 4), quien sirve contenidos estáticos o envía la solicitud al servidor de aplicaciones Jboss a través del balanceador. Posterior recepción del requerimiento por parte del servidor JBoss, este hace requerimientos a la base de datos Sybase. Una vez resuelto por el servidor de aplicaciones el requerimiento toma el camino de regreso al cliente que realizó dicha solicitud pasando por Apache, balanceador F5 y cortafuego(sólo si es de internet).

5.1.2. Arquitectura lógica de los desarrollos de la SBIF

La arquitectura de la actual aplicación web de la SBIF (llamada SBIFWEB), y con la cual fue necesario trabajar al momento de implemntar la API, es la que se describe en el diagrama de la Figura 4.

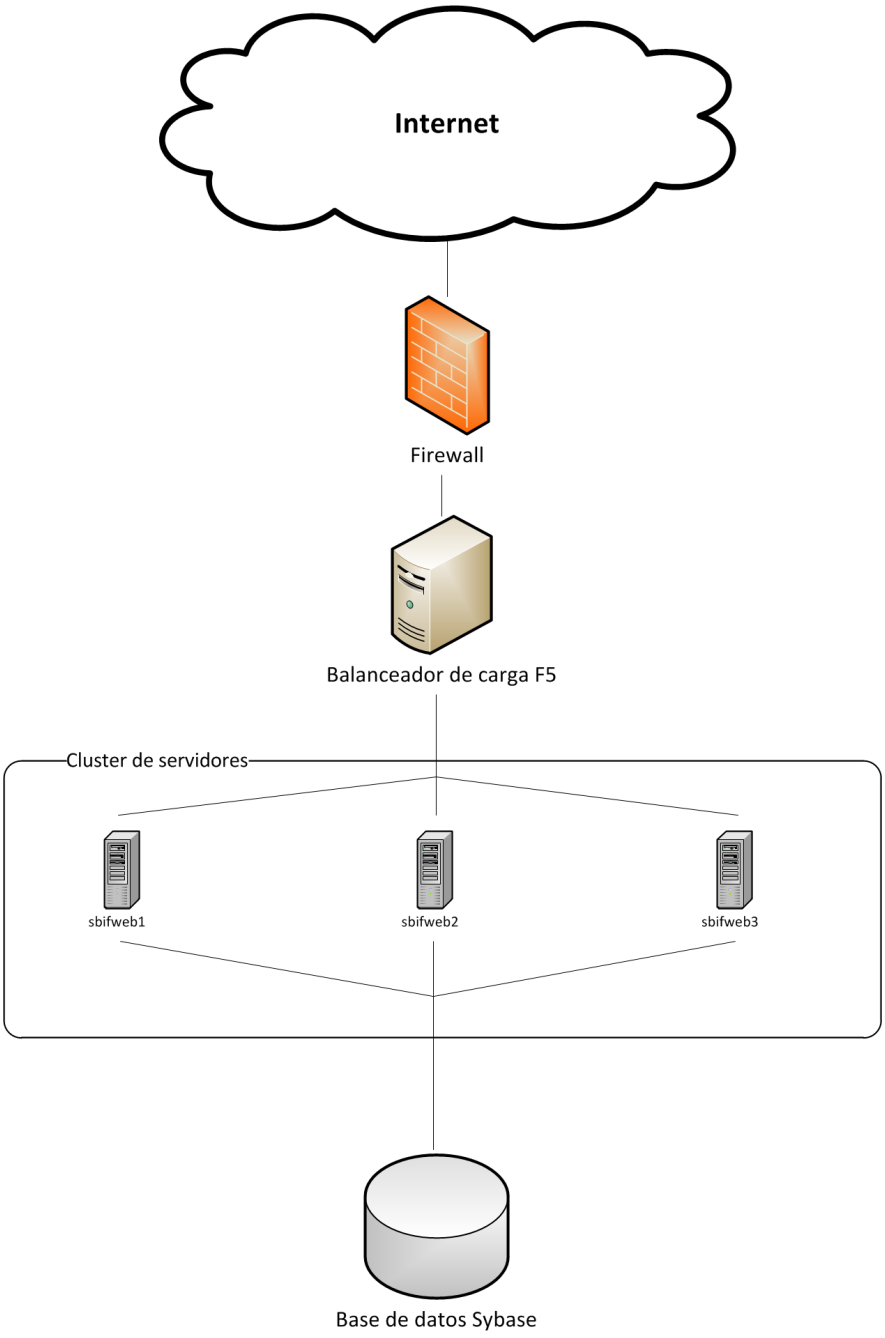

Figura 3: Diagrama de la arquitectura física donde habita la solución

Se trata de una arquitectura de tres capas que sigue el modelo MVC[5]. Está implementada usando servlets Java, sin la ayuda de un framework. Fue diseñada por el ingeniero informático Hugo

[5]Modelo Vista Controlalor. http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

Martínez, quien forma parte de la unidad de Internet y publicaciones de la SBIF.

Esta arquitectura consta de tres partes principales:

- Un servidor web Apache encargado de gestionar los requerimientos entrantes.
- Un servidor de aplicación JBoss encargado de manejar todos los proyectos Java.
- Un motor de base de datos Sybase donde está contenida la información de la SBIF.

La arquitectura con la cual se desarrolló la aplicación web SBIFWEB, es una arquitectura centralizada. Todos los requerimientos entrantes pasan por un único servlet de control llamado *Portada*, este servlet hereda de una clase abstracta llamada *Componente*, la cual define métodos estándar para el control de la aplicación. La labor de este servlet es ser quien decide dependiendo del requerimiento, a qué clase Java llamar para solicitar los datos correspondientes a la base de datos. Estas clases Java auxiliares son las llamadas *Agentes*.

Las vistas de la aplicación son generadas usando XSLT. Para ello, todos los agentes generan como respuesta un archivo XML con la información que obtienen desde la base de datos.

El flujo operacional descrito por la aplicación existente se describe detalladamente en la Figura 4. Paso a paso lo que ocurre es:

- Paso 0. Un usuario genera un requerimiento HTTP hacia el servidor de la SBIF ingresando una URL en su browser.
- Paso 1. El requerimiento es decepcionado por nuestro servidor web Apache, quien filtrará la URL.
- Paso 2. Apache filtra la URL. Si el recurso es dinámico el requerimiento es derivado al servidor de aplicación JBoss. En caso de ser un recurso estático el requerimiento es servido inmediatamente por Apache.
- Paso 3. El requerimiento es recibido por el servlet de control *Portada*. Este servlet recibe un parámetro GET llamado *indice*, el cual es consultado en base de datos para determinar la ruta al archivo XSLT que será el encargado de procesar la vista de respuesta al requerimiento.
- Paso 4. El servlet de control *Portada* llama al agente indicado para atender la consulta.
- Paso 5. El agente llamado por *Portada* hace una consulta a la base de datos en busca de los datos necesarios para satisfacer el requerimiento.
- Paso 6. Luego de obtener los datos necesarios, el agente genera un documento XML con todos los datos del despliegue de la respuesta, el cual entrega como respuesta a *Portada*.
- Paso 7. *Portada* teniendo en su poder el archivo XML con los datos de despliegue y la ruta al archivo XSLT con el cual se dará estilo a los datos, hace un llamado al módulo que convierte estos dos archivos en un documento HTML.
- Paso 8. El módulo de conversión teniendo la ruta al archivo XSLT, va al sistema de archivos y lo busca. Luego genera el HTML.

- Paso 9. El módulo entrega el documento HTML al servidor web Apache, el cual lo entrega como respuesta HTTP.
- Paso 10.El browser recibe la respuesta a su petición como un documento HTML, el cual se despliega ante el usuario.

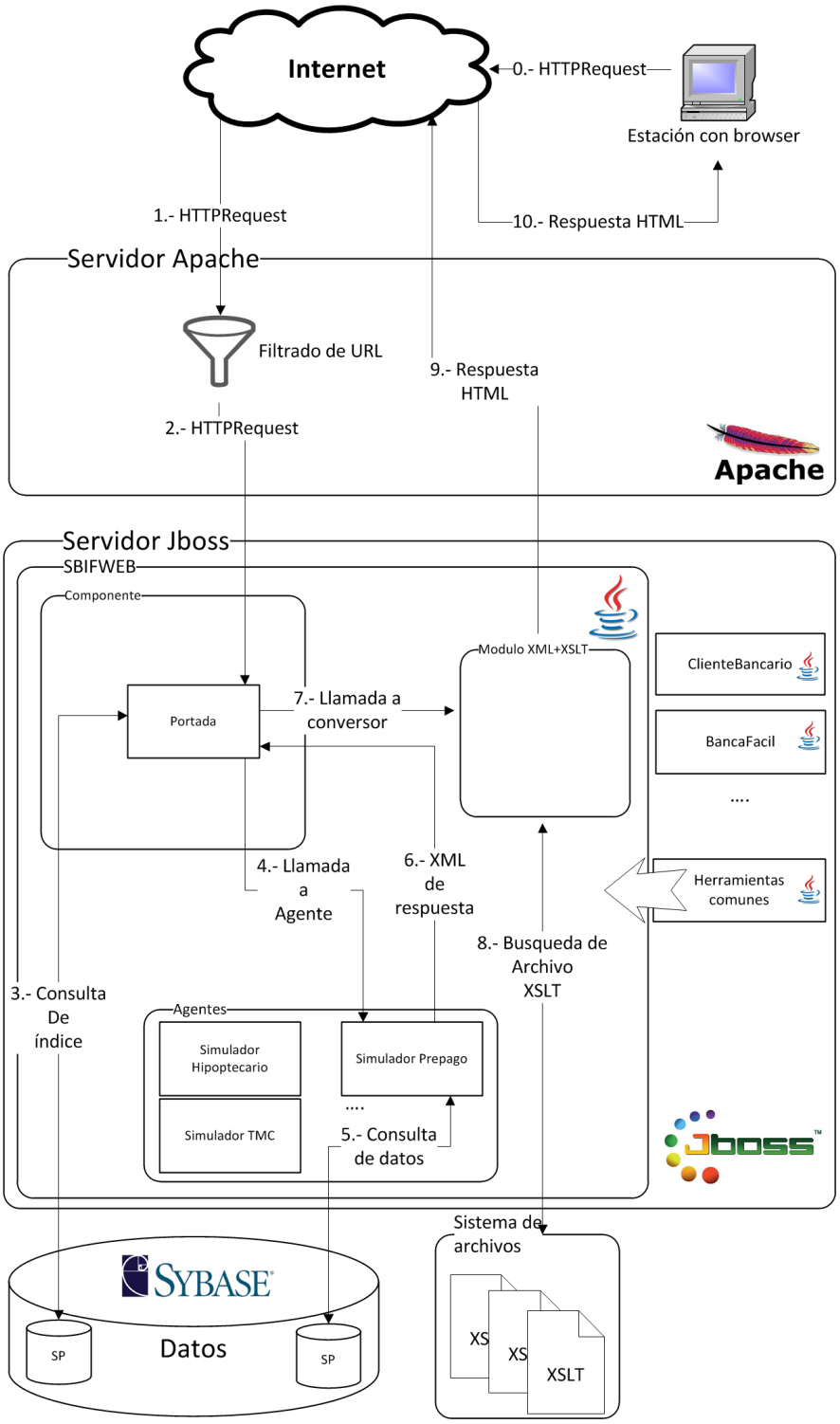

Figura 4: Diagrama de la arquitectura lógica de los desarrollos de la SBIF previa existencia de la solución

5.2. Primer acercamiento de solución

En esta sección se describe la solución contemplada en primera instancia, luego de identificado el problema y antes de comenzar la implementación. Es importante esta descripción preliminar, ya que da evidencia del proceso vivido durante el desarrollo, el cual determinó que el producto final fuese muy distinto al contemplado en un comienzo.

5.2.1. Estrategia de solución

Ya sabíamos que la solución al problema tenía que ser la implementación de una API de servicios web que operen usando REST (ver sección 4.2). La razón de postular esta solución estaba inspirada en la existencia de un servicio web de uso interno de la SBIF, el cual estaba implementado usando el protocolo SOAP. Este servicio web se encarga de alimentar con los balances de las instituciones financieras a diversas tablas contenidas en el sitio de la SBIF.

La forma en la cual este servicio web entrega los balances es recibiendo como parámetro de entrada la institución, el año y mes del balance. La respuesta del servicio web es un archivo XML que contiene la información solicitada.

Dado que este servicio web resultaba ser una suerte de API operando dentro de la red interna de la SBIF, la estrategia para implementar la solución fue replicar el código de este servicio web, quitarle el envoltorio provisto por el protocolo SOAP y dejarlo operando como un componente REST. Una vez lograda esta transformación de ese servicio web SOAP, hacer lo mismo para otros recursos que se disponibilizarían con la API (recursos aún no definidos en esta etapa). Posterior implementación de estos servicios web, se permitiría acceso a los mismos desde internet, haciendo uso de algún mecanismo de autentificación/autorización.

5.2.2. Tecnología considerada

Dado que todos los desarrollos de la SBIF están hechos en Java, existe una infraestructura de servidores optimizada para el uso de este lenguaje en los desarrollos. Esto implicó que optar por otra tecnología no fuese viable en primera instancia, dado que acarrearía consigo mayor carga a las máquinas al tener que instalar nuevos servidores. Este punto presentó bastante falta de flexibilidad y nos obligaba a trabajar con un lenguaje que si bien es muy potente y posee innumerables ventajas, es lento en tiempos de desarrollo y ejecución.

5.2.3. Metodología de trabajo

La metodología con la cual se acordó abordar el proyecto en primera instancia contempló un desarrollo por etapas, las cuales fueron concebidas desde un punto de vista de alto nivel. La idea tras la disposición de dichas etapas fue tener tempranamente ubicadas las etapa de acercamiento y experimentación, las cuales se contemplaron con el fin de introducir al memorista de forma gradual al tema. Finalmente se propuso acabar con etapas prácticas donde se generaría e implantaría la solución definitiva.

Las etapas contempladas fueron las siguientes:

1. Documentación relacionada a servicios web y APIs en base a los mismos. La idea en esta etapa, ademas de obtener la información necesaria para el comienzo del proyecto, fue crear y ejecutar un servicio web SOA sencillo para efectos de prueba.

2. Investigar sobre diversos formatos de representación de datos para la extracción de la información desde las bases de datos. Tales formatos contemplan CSV[6], XML, XBRL[7] [3] entre otros. Al finalizar esta etapa, se espera transformar el servicio web SOA usado para el buscador interactivo en un servicio web REST, donde la información se entrega representada según los datos aportados durante la investigación previamente realizada.

3. Informarse sobre sobre el uso de API keys para la autentificación y autorización de usuarios. En base a esta información, crear un sistema de accesos y administración de cuotas de información por usuario.

4. Desarrollar un nuevo servicio web REST, como una réplica del anteriormente realizado, el cual entregara indicadores financieros.

5. Realizar pruebas de carga para medir cual será la repercusión en el sistema existente de esta nueva implementación. Lo que se persigue en esta etapa es poder saber previamente y documentar cómo se comportará el sistema ante el acceso concurrente de volúmenes de usuarios.

6. Como etapa transversal a toda la memoria, se realiza una documentación sobre la forma en la cual un usuario debe operar el API. Esta documentación es entregada directamente al señor Juan Carlos Camus, jefe de unidad de internet y publicaciones de la SBIF, quien se encargará de publicarla de una forma adecuada para que los desarrolladores puedan hacer uso de la API.

Los problemas que se esperó abordar inicialmente fueron:

- Identificación de usuarios en base a API keys y control de cuotas de acceso a la información.
- La forma como escala la arquitectura en relación al número de usuarios.

5.2.4. Plan de actividades proyectado

Temporalmente se proyectó que estas actividades se lleven a cabo según la carta Gantt mostrada en la Figura 5:

[6] Comma-Separated Values.
[7] eXtensible Business Reporting Language.

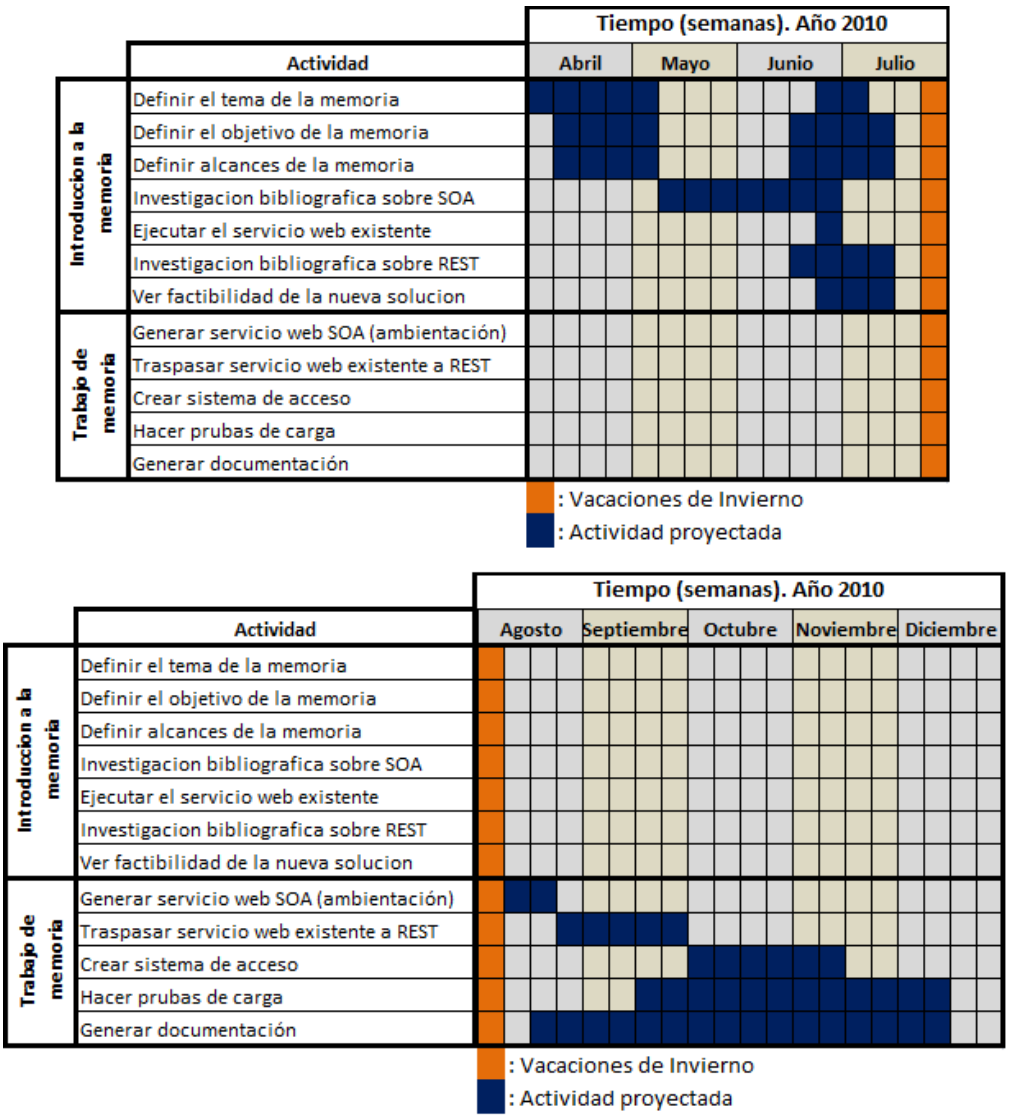

Figura 5: Carta Gantt de las actividades contempladas durante la memoria.

5.3. Objetivos del proyecto

Luego de tener un primer y muy general acercamiento a la solución que se adoptará para el problema, los objetivos definidos para el proyecto son los que se detallan a continuación.

5.3.1. Objetivo general

Desarrollar una API usando una arquitectura de servicios web REST, la cual permita acceder a la información bancaria provista por SBIF a través de su sitio web, representada en formatos estándar.

5.3.2. Objetivos específicos

- Permitir que la información incorporada en la base de datos, por medio de la API, pueda ser obtenida para ser utilizada y presentada. Dicha información debe estar estructurada como parte de un repositorio de Open Data.[8]
- Transformar el servicio web existente para el buscador interactivo del sitio web en un servicio web que entregue la información usando REST.
- Generar un sistema de acceso basado en API keys[9] para la autentificación y autorización de usuarios, además de la administración de la cuota de uso de la información financiera de la Base de Datos SBIF [6].
- Hacer pruebas de carga sobre la base de datos, servidor de aplicación y red, con el fin de verificar cómo escala la aplicación en relación al número de consultas.
- Desarrollar un nuevo servicio web que entregue Indicadores Financieros (UF, Dólar,. Euro, IPC, UTM y Tasas de Interés) usando REST.
- Proveer documentación necesaria sobre el uso de la API al Jefe de la unidad de internet y publicaciones, la cual él se encargará de publicar.

5.4. Descripción general de solución definitiva

En esta sección se describe de forma general, la solución definitiva que resultó luego del proceso de desarrollo e implantación. Es muy interesante contrastar lo que se proyectó en un comienzo para la solución (ver sección 5.2) y cómo ésta fue cambiando a medida que surgían inconvenientes, hasta llegar a convertirse en lo que hoy se encuentra operativo.

5.4.1. Estrategia de solución

La estrategia usada es la que se encuentra ilustrada en la Figura 6.

Primero que todo, la API de la SBIF está definida como un conjunto de servicios web diseñados bajo la técnica de arquitectura de software llamada REST, los cuales están bajo otro servicio web de control y administración. Cada servicio web, es el encargado de satisfacer las peticiones asociadas

[8] Ver en http://en.wikipedia.org/wiki/Open_Data

[9] Ver en http://en.wikipedia.org/wiki/Application_programming_interface_key

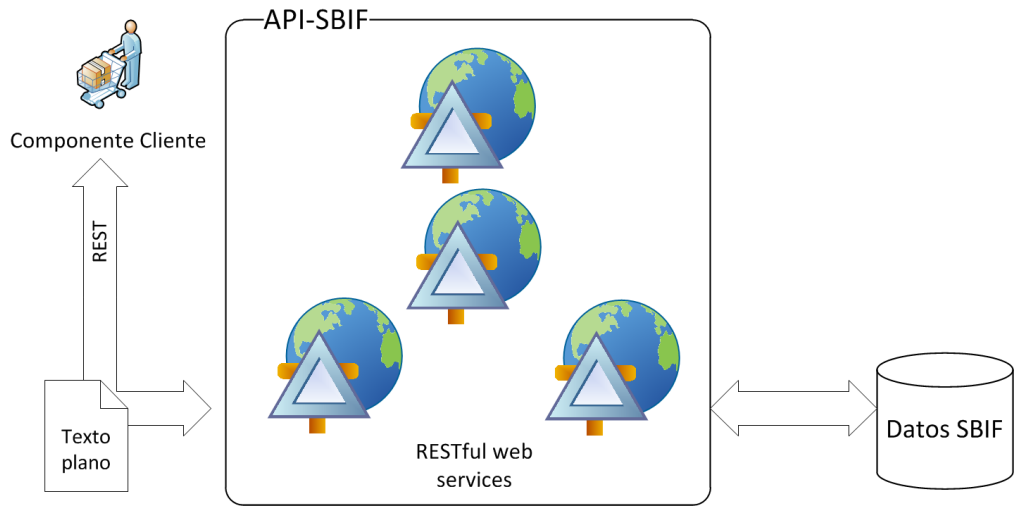

Figura 6: Descripción general de la solución implementada

a un tipo particular de recurso definido dentro de la API. Ante una petición, este servicio web ha de responder mediante texto plano, una representación estándar de dicho recurso al componente de software que lo solicita. Esta representación es negociada entre la componente de software que la solicita y la API.

El lugar donde se encuentra la información es la base de datos de SBIF. Sin embargo, la interpretación de esta información como recursos existe dentro de la API.

Una de las grandes diferencias existentes con la estrategia considerada en primera instancia, es que finalmente no fue posible rescatar el código de el servicio web SOAP existente. Dado que aquel servicio presentaba demasiada rigidez para sus consultas, el costo de adaptar dicho código a nuestras necesidades era mucho más alto que reescribir el servicio web completo.

5.4.2. Tecnología considerada

Al momento de comenzar el desarrollo de la API, el panorama en cuanto a la tecnología a usar en desarrollo había cambiado por completo. Se vivía en aquel tiempo un proceso de renovación tecnológica en el área de internet y publicaciones, dado que habían caído en la cuenta de que Java(usado sin ayuda de ningún framework) como lenguaje era muy lento a nivel de desarrollos. En este marco se comenzó a experimentar con diferentes lenguajes y diferentes plataformas, para ver cuál sería la tecnología con la que se implementaría la API y los desarrollos futuros de la SBIF.

Dado que la decisión tecnológica no solo involucraba el desarrollo de la API, sino también los desarrollos a futuros de la SBIF, era necesario llegar a a un consenso entre los intereses del memorista como implementador de la API y los intereses de la unidad de internet y publicaciones como desarrolladores estables.

Las pruebas fueron muchas, entre los lenguajes y plataformas probadas podemos destacar:

- Java haciendo uso del framework Spring.
- Java haciendo uso del framework Struts.
- Ruby on Rails.
- Python haciendo uso del framework Django.
- Groovy on Grails.
- ASP .NET usando el framework MVC.

Luego de aproximadamente una semana investigando y haciendo pruebas, se tomó la decisión de usar Groovy on Grail (desde este momento llamado solo Grails) como tecnología considerada para la API y los futuros desarrollos de la SBIF.

Las características que llevaron a la elección de Grail fueron las siguientes:

- Corre sobre la plataforma Java, por lo cual no es necesario hacer cambios en los servidores de aplicación.
- El lenguaje en el cual se codifica es Groovy, el cual presenta muchas similitudes con Java pero es de más alto nivel.
- Permite integrar clases Java de forma natural a un proyecto Grails, lo cual asegura la reutilización de código con proyectos antiguos.
- El desarrollo de aplicaciones es muy ágil, muy similar a Ruby on Rails.
- Grails hace uso de la interface Hibernate para agilizar trabajo con el motor de la base de datos.

5.4.3. Metodología de trabajo

La metodología de trabajo usada fue una mezcla entre un desarrollo por etapas y un desarrollo ágil. Debido a que en comiendo había grandes incertidumbres en el proyecto, una carta Gantt por si sola resultaba ser demasiado estricta. Había muchos requerimientos de la aplicación que aparecían en medio del desarrollo, lo cual hizo necesario re priorizar tareas y re estimar tareas muchas veces. Una vez descubierta esta limitante, se optó por generar una carta Gantt con las etapas más marcadas, las cuales sabíamos a ciencia cierta que tenían que ser cumplidas. Cada etapa poseía dentro todas las tareas propias de la etapa, luego se estimaron tiempos reales para cada una de esas tareas y un tiempo de holgura para la etapa completa (el doble de la suma de los tiempos reales de las tareas), y se creó un contador de adelanto. La idea es que cualquier actividad extra, no prevista o atrasada que apareciera dentro de una etapa, hiciera uso del tiempo de holgura de la etapa. En caso de agotado el tiempo de holgura, se podía hacer uso del tiempo contenido en el contador de adelanto. Si este tiempo de holgura de una etapa no era usado, nuestro contador de adelanto aumentaba para una etapa posterior. En caso contrario, si una nueva actividad consumía todo el tiempo de holgura de una etapa, se hacía necesario re-priorizar actividades de la etapa dejando cosas fuera del proyecto. Afortunadamente no fue necesario dejar nada fuera.

La revisión de tiempos de las actividades fue realizada semanalmente. La notificación de actividades terminadas era inmediata via e-mail al Jefe de la unidad de internet y publicaciones. Los términos de etapas eran subidos al ambiente de test para que el resto de la unidad hiciera pruebas y entregara retroalimentación.

5.4.4. Plan de actividades proyectado

Temporalmente se proyectó que estas actividades se lleven a cabo según la carta Gantt mostrada en la Figura 7:

Afortunadamente el desarrollo fue exitoso, y se pudo cumplir la carta Gantt proyectada con un contador de adelanto de aproximadamente 2 semanas.

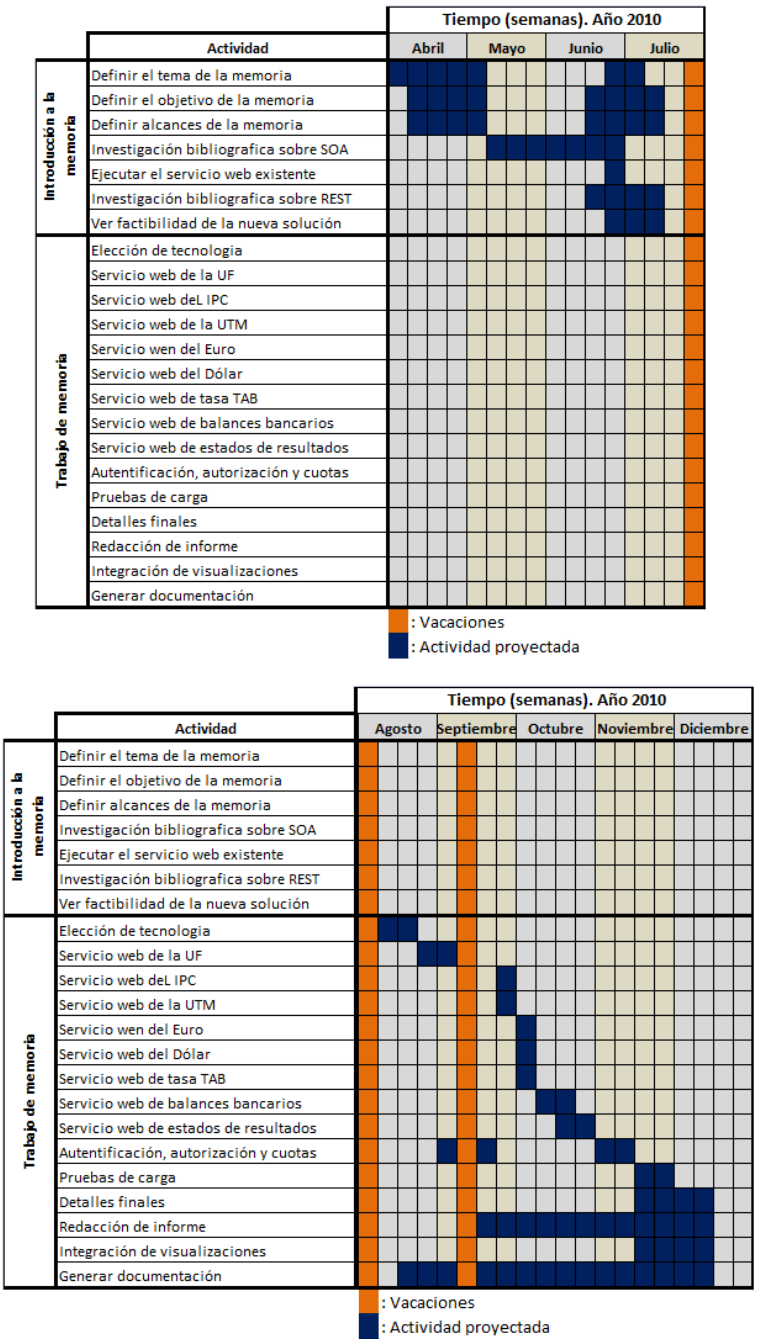

Figura 7: Carta Gantt refinada de las actividades contempladas durante la memoria.

5.5. Descripción detallada de solución definitiva

En esta sección se describe en detalle la solución resultante implementada para la SBIF, la cual actualmente se encuentra en producción[10].

5.5.1. Arquitectura física de la solución

En un comienzo y dado que quedó abierta la posibilidad de un cambio de plataforma para los nuevos desarrollos, se contempló la idea de la creación de nuevos servidores virtualizados con Linux para la operación de la API (y de los futuros proyectos desarrollados por la SBIF). Sin embargo, dada la elección del lenguaje y framework de trabajo descrita en la sección 5.4.2, donde los proyectos nuevos funcionan también usando la plataforma Java, no fue necesario hacer extensión alguna. Finalmente esto llevó a que la arquitectura física de la aplicación siguiera como siempre. Esta arquitectura está descrita en detalle en la Sección 5.1.1.

5.5.2. Arquitectura lógica de la solución

La arquitectura lógica de la API de la SBIF es la que se encuentra descrita en la Figura 8. Nació en abril del año 2004 con el equipo que desarrolló del actual sitio de la SBIF.

Esta arquitectura consta de tres partes principales:

- Un servidor web Apache encargado de gestionar los requerimientos entrantes.
- Un servidor de aplicación JBoss encargado de manejar todos los proyectos Java.
- Un motor de base de datos Sybase donde está contenida la información de la SBIF.

La arquitectura lógica del API de SBIF es la que se describe en la Figura 9. Esta arquitectura asigna a cada recurso(ver recursos en la Sección 5.5.3) un servicio web encargado de atender todas las peticiones asociadas al mismo.

Sin embargo, esta arquitectura está centralizada. Esto quiere decir que todos los requerimiento pasan necesariamente por un servicio web de control, independiente del recurso que estén solicitando. Este servicio web de control es el encargado de la autentificación y autorización de los usuarios mediante un *API Key* (Ver Sección 5.5.7 para mayor detalle).

Para clarificar como funciona la arquitectura, tomaremos los diagramas de las Figuras 8 y 9 y veremos como es el flujo de ejecución dado un requerimiento HTTP.

- Paso 0. Un componente de software genera un requerimiento HTTP solicitando un recurso al API de la SBIF.
- Paso 1. El requerimiento HTTP, luego de viajar por internet, llega al servidor Apache de la SBIF.
- Paso 2. El servidor Apache revisa la URL de entrada y verifica si es una URL permitida de ser consultada desde internet (esto es porque los servicios web de cada recurso son sólo accesibles desde dentro de la SBIF). Si es un requerimiento admitido, este es derivado directamente al API de la SBIF dentro del servidor JBoss, donde lo recibe el controlador de acceso.

[10] Aplicación y documentación necesaria disponible en el sitio web http://api.sbif.cl/API

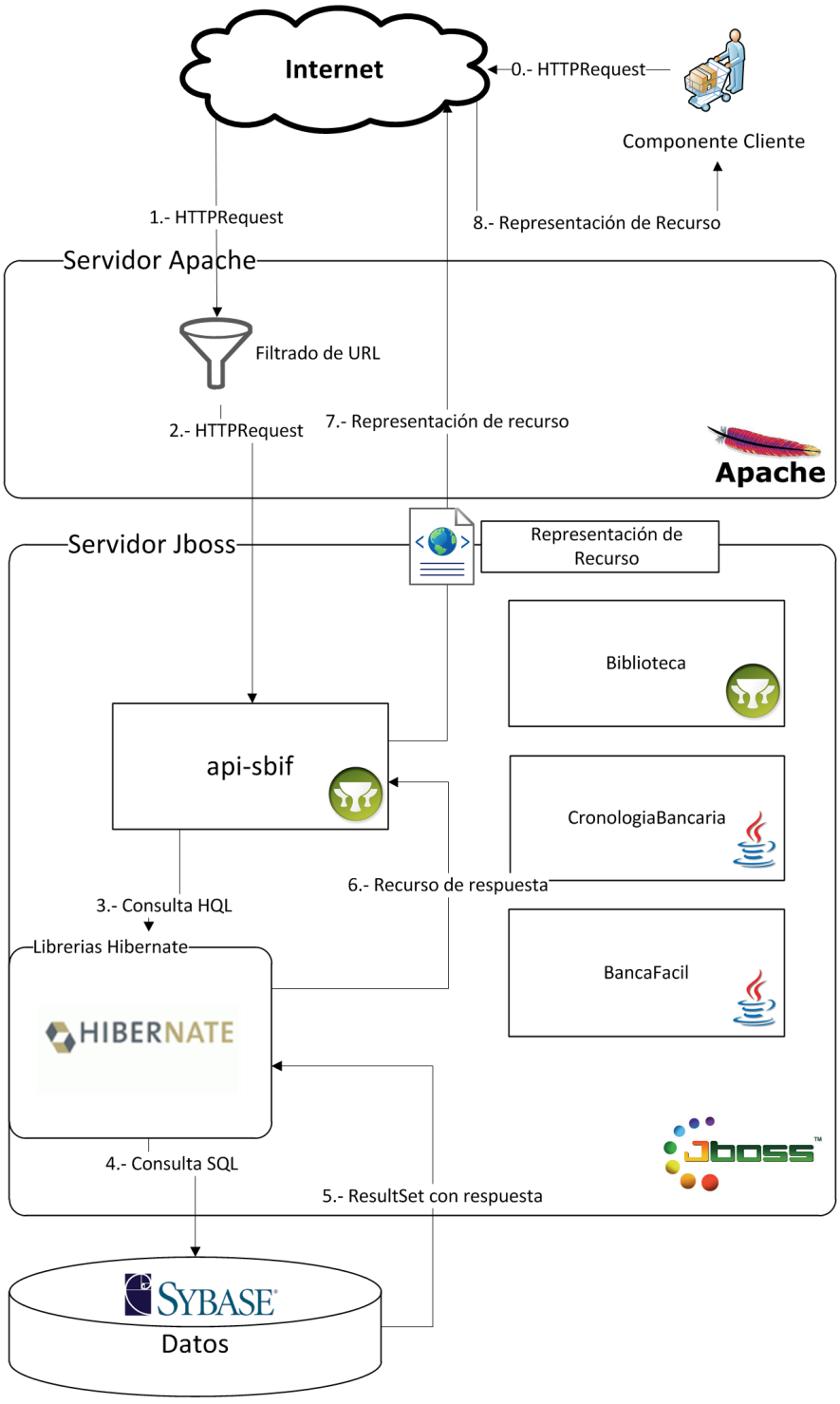

Figura 8: Diagrama de la arquitectura lógica donde habita la solución

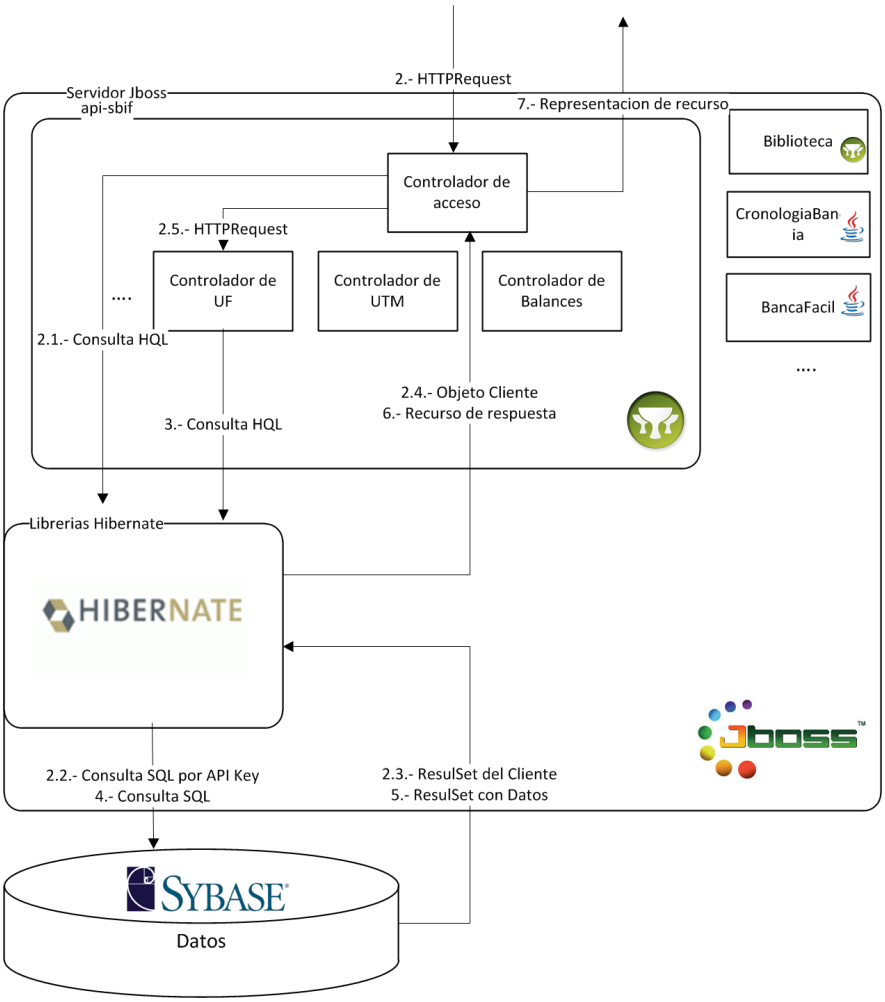

Figura 9: Diagrama de arquitectura lógica de la aplicación

- Paso 2.1. El controlador de acceso envía una consulta en HQL[11] al las librerías de Hibernate del servidor, consultando por el usuario al cual pertenece el API Key.
- Paso 2.2. Las librerías de Hibernate transforman esta consulta en una consulta SQL y la envían al motor de base de datos Sybase.
- Paso 2.3. El motor de base de datos retorna un resultSet con la respuesta a la consulta SQL por el usuario al que pertenece el API Key.
- Paso 2.4. Las librerías de Hibernate retornan un objeto Java que representa al propietario del API Key.
- Paso 2.5. En base al recurso consultado, el controlador de acceso deriva el requerimiento HTTP al servicio web correspondiente.

- Paso 3. La API realiza una consulta HQL por el recurso solicitado a las librerías de Hibernate.

[11] Hibernate Query Languaje

- Paso 4. Las librerías de Hibernate transforman esta consulta en una consulta SQL y la envian al motor de base de datos Sybase.
- Paso 5. El motor de base de datos retorna un ResultSet con la información solicitada.
- Paso 6. Las librerías de Hibernate retornan un objeto Java que representa al recurso solicitado.
- Paso 7. La API de la SBIF retorna una representación del recurso solicitado al cliente, según se solicitó en el requerimiento.
- Paso 8. La representación del recurso es retornada al componente de software solicitante.

5.5.3. Recursos

Si bien los recursos disponibles dentro del API de la SBIF no son estáticos y dependen de la URI con la cual son llamados(para mayor detalle ver Sección 5.5.5 y 8.3) estos se encuentran agrupados en 8 macro-grupos, los cuales en sí son entidades dentro del diagrama entidad relación la Base de Datos de la SBIF. Estos grupos son los que muestran a continuación.

Indicadores financieros:

- Unidad de Fomento (UF).
- Unidad tributaria mensual (UTM).
- Índice de precios al consumidor (IPC).
- Valor del Euro.
- Valor del Dolar.
- Tasa TAB UF 360.

Reportes Bancarios:

- Balances de las instituciones financieras.
- Estados de resultados de las instituciones financieras.

5.5.4. Estructura de la base de datos

Con el fin de aislar el API de las demás aplicaciones de SBIF, se creó una base de datos especial para la operación del API, la cual contiene sólo las tablas necesarias para alimentar los recursos que provee. También se generaron los mecanismos adecuados en las antiguas aplicaciones, para que cada inserción de datos en las tablas antiguas también generara la misma inserción en las nuevas tablas.

La estructura de la base de datos es bastante simple, dado que se trata de tablas aisladas las cuales no tienen relaciones entre ellas. El diagrama de la estructura es el que se puede ver en la Figura 10.

La razón de este aislamiento de tablas fue poder monitorear al API con relación al motor de Base de Datos de forma directa, dado que se estima que el API tendá una gran carga de consultas en un futuro cercano.

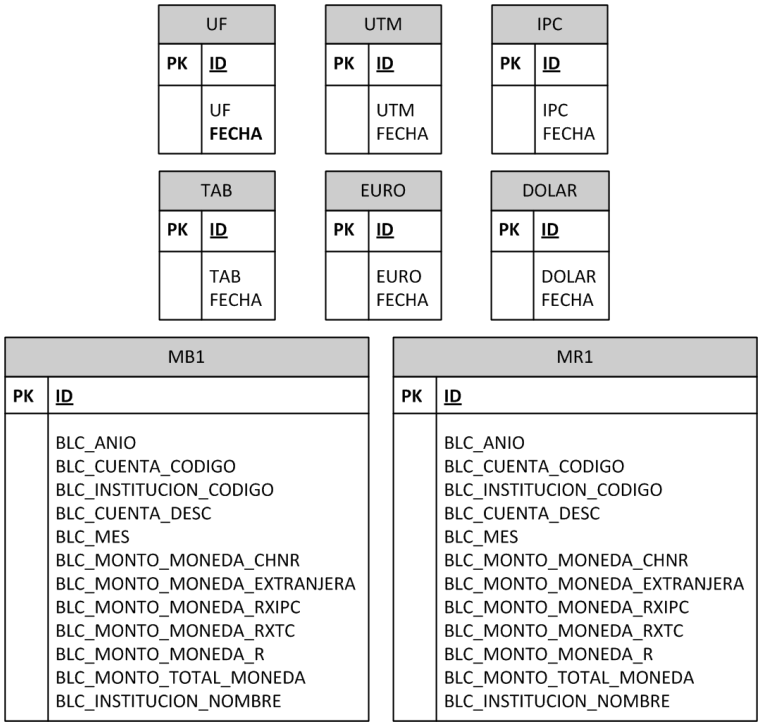

Figura 10: Estructura de la base de datos

5.5.5. Identificadores de recursos (URIs)

En este punto fue necesario poner especial cuidado, dado que en este proyecto hablamos de una aplicación que no tiene interfaze de usuario, por lo cual la cara visible de la aplicación es la estructura de la URIs con la cual se llama a los recursos.

En un comienzo no se tenía mucha referencia respecto a que estrategia usar, en este marco nació la estructura bajo la cual se rigen los recursos de todos los indicadores financieros (UF,UTM, IPC, Dólar, Euro, tasa TAB). Este esquema está orientado a la temporalidad y usa como referencia las fechas. Las modalidades de consultar en este esquema son:

- Actual: Entrega el indicador de la fecha actual.
- Anterior a: Entrega los indicadores de fechas anteriores a la consultada.
- Posterior a: Entrega los indicadores a una fecha posterior a la consultada.
- Período: Entrega los indicadores de un periodo comprendido entre dos fechas consultadas.

Para mayor detalle ver en Anexos la Sección 8.3.

Sin embargo, posterior a esta implementación de los proveedores de información de los indicadores financieros, se escuchó hablar que usualmente en este tipo de APIs, se usa una estructura orientada a recursos. En este punto es donde nace la estructura que se dio a los recursos relacionados con los reportes bancarios (balances y estados de resultados de instituciones financieras). La idea para definir la estructura con orientación a recursos fue usar una estrategia *llave/identificador*.

La primera versión de esta estructura es la que se muestra en la figura 11. Esta es una representación usando una estructura de sistema de archivos para representar la URI. La consulta que se representa es:

```
http://api.sbif.cl/api-sbif
/acceso/43ffef8d0SBIFb4b175b85d564538151539f3954
/xml/resultados/2010/01/instituciones/001
```

Esta consulta entrega como resultado el estado de resultados de Enero del 2010 del Banco de Chile (código 001).

Luego de implementada esta primera versión de estructura de URIs, se recibió retroalimentación de parte de nuestros testers (ver Sección 6.2), la cual hacía referencia a separar dentro de la misma URI lo referente al recurso y lo referente a otros aspectos secundarios, como la autentificación o la representación de los recursos. En este marco nace la representación definitiva de recursos, la cual se ve en la Figura 12. Esta estructura representa a la siguiente consulta:

```
http://api.sbif.cl/api-sbif/recursos
/resultados/2010/01/instituciones
?apikey=43ffef8d0SBIFb4b175b85d564538151539f3954f
&formato=xml
```

Esta consulta entrega la lista de instituciones financieras (y sus códigos) que poseen un estado de resultados publicado en el mes de Enero del año 2010.

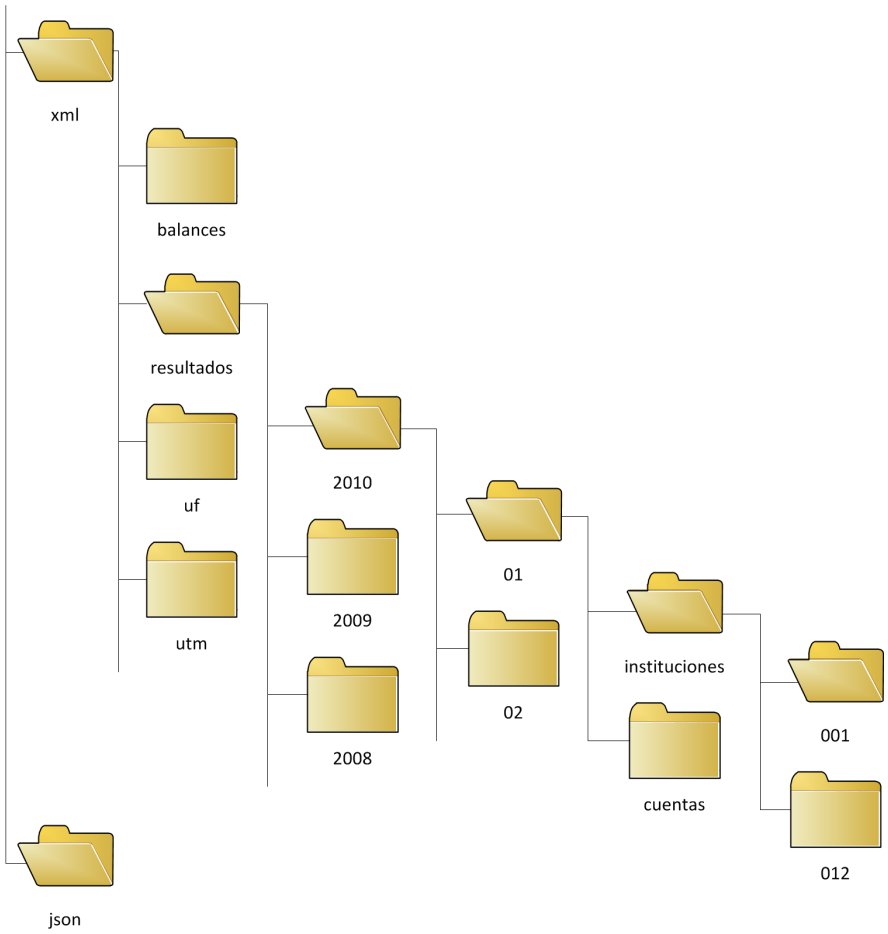

Figura 11: Estructura de URIs. Primera versión

Como se puede observar, para la obtención del recursos en sí se hace uso de la URI, mientras que para representar elementos de la negociación de la representación o la autentificación, se hace uso de parámetros GET del protocolo HTTP.

En este esquema, si queremos ir más profundo en la consulta anterior, podemos bajar un nivel y hacer la consulta representada en la Figura 13. Esta consulta, al igual que la consulta representada en la Figura 11, entrega como resultado el estado de resultados de Enero del 2010 del Banco de Chile (código 001). Sin embargo, la URI que la representa es:

```
http://api.sbif.cl/api-sbif/recursos
/resultados/2010/01/instituciones/001
?apikey=43ffef8d0SBIFb4b175b85d564538151539f3954f
&formato=xml
```

Esta URI en sí resulta ser mucho más intuitiva que la anteriormente ilustrada y permite que implementadores que usen nuestra API requieran menos visitas a la documentación para poder usarla. Esto transforma nuestra API en un producto mucho más usable.

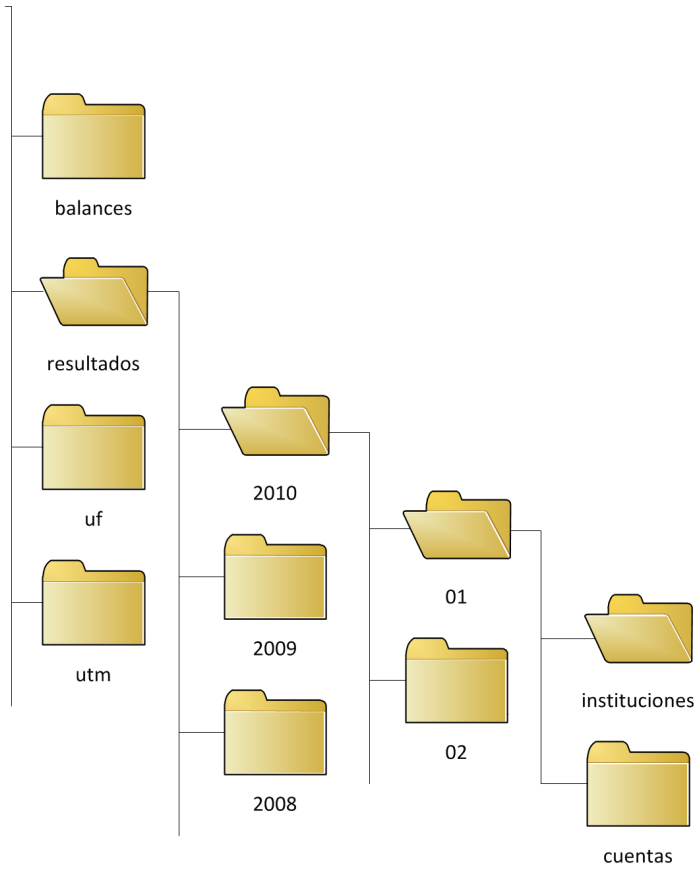

Figura 12: Estructura de URIs. Primer nivel

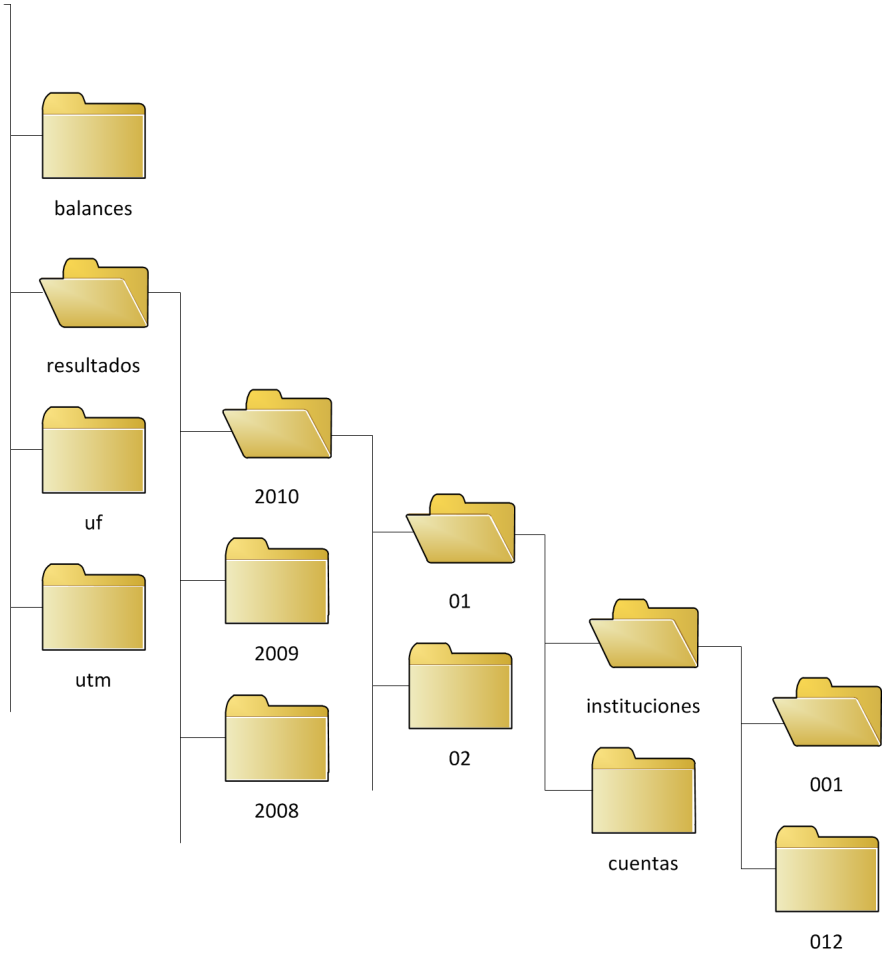

Figura 13: Estructura de URIs. Segundo nivel

5.5.6. Representación de los recursos

Para la representación de los recursos se consideró desde el inicio del proyecto, los formatos XML y JSON de forma prioritaria, mientras que de forma sujeta a evalución el formato CSV, el cual finalmente fue descartado por una razón costo/beneficio. La elección de estos formatos se debe a que la información debía ser entregada de la forma más estándar posible y estos dos formatos son muy populares. Sin embargo, a medida que lanzábamos nuevas versiones de prueba, se presentó un problema detectado por uno de nuestros testers. Este problema es conocido como el problema de las llamadas AJAX entre dominios. Se origina en los navegadores, los cuales por razones de seguridad no permiten que javascript ejecute llamadas usando AJAX a dominios diferentes del cual proviene el código. Este hecho genera problemas al momento que la carga de un sitio externo haga un llamado a la API de la SBIF usando AJAX. Es a causa de esto que nos vimos en la obligación de implementar la entrega de datos usando JSONP además de XML y JSON[12].

JSONP o JSON con Padding [13] en sí no es un formato, sino es una forma de obtención de datos basada en el formato JSON. La técnica consiste en la inserción de un archivo obtenido desde el API dentro del código HTML del cliente como un script javascript (donde el código fuente está en la URI de consulta al API), el cual en su interior contiene una llamada a función javascript que dentro tiene como parámetro el código JSON solicitado (objeto javascript). Con esto se soluciona el problema de seguridad originado al solicitar archivos al API usando AJAX.

5.5.7. Autentificación y autorización

De todos los puntos a abordar en el desarrollo del API, en un comienzo el problema de auntentificación y autorización fue el más inmaduro, dado que no se tenía noción de que forma poder controlar quien accedía a los recursos ni cuantas veces lo hacía, dado que el API REST por definición es stateless. Lo único que se tenia claro es que era necesario proteger el acceso a la información, no por la información en sí, sino porque el número de consultas podía hacer caer nuestro servidor.

Los requisitos que se tenía para la implementación del sistema de acceso fueron:

1. Que se pueda identificar quién esta accediendo al API.

2. Que se maneje una cuota de acceso por usuario.

En este marco se barajó como primera opción el uso del protocolo abierto OAuth [14]. Este protocolo es el usado en muchas APIs populares, como por ejemplo, en el API de Twitter. Sin embargo, esta implementación poseía características adicionales que en nuestro caso no eran necesarias, las cuales hacían de esta alternativa demasiado costosa en tiempo de desarrollo, ademas de poner dificultad en el acceso a los clientes del API. Es por esto que la implementación de Oauth fue desechada.

Luego de esto y por consejos de Hector Vergara, un implementador con experiencia implementando APIs REST, se exploró otra posibilidad mucho más liviana, la cual estaba basada en identificar a los usuarios en base a una clave de acceso llamada *API Key*. Esta clave debía ser mantenida en secreto por los usuarios propietarios. La API Key se genera mediante un algoritmo de hashing y

[12] http://web.ontuts.com/tutoriales/jsonp-llamadas-ajax-entre-dominios/
[13] http://blog.ikhuerta.com/jsonp-json-con-padding
[14] http://oauth.net/

debía ser entregada junto con la consulta cada vez que un recurso es solicitado. Luego como esta API Key me permitía identificar a los usuarios de lado del servidor, se podía hacer un sistema de administración de cuotas. Finalmente se optó por tomar esta segunda alternativa, la cual es mucho mas liviana en ejecución y tomó mucho menos tiempo de desarrollo.

La temporalidad con la cual se acordó la renovación de la cuota de acceso fue mensual. Sin embargo, definir el número de accesos que se permitiría fue muy difícil y actualmente no se encuentra completamente definido. La razón es que en vez de fijar un número de accesos se está usando una estrategia dinámica. Esta estrategia consiste en entregar al primer mes un número casi ilimitado de accesos, aprovechando que el número de usuarios no sería mucho. Luego a medida que transcurran los meses de operación del API y a medida que se vallan sumando nuevos usuarios, ir achicando la cuota progresivamente a un valor entre el promedio de consultas por usuario y el máximo permitido por usuario. Se espera según el comportamiento observado en el poco tiempo que la aplicación lleva operando, que este valor debería converger a un número cercano a las 5000 consultas por usuario al mes.

También fue una opción en algún momento combinar el acceso basado en API Keys con el uso de SSL. Dado que cualquier persona capas de interceptar el requerimiento de un usuario podría obtener el API Key de dicho usuario y usar su cuota de acceso. Sin embargo, esta alternativa fue desechada por las siguientes razones:

1. SSL introduce carga de procesamiento adicional al servidor al encriptar la información enviada.
2. SSL requiere la compra e instalación de un certificado en los servidores.
3. El peor caso es que a un usuario le roben la clave de acceso agotando su cuota mensual, lo cual no es perjudicial para el API.

5.5.8. Manejo de errores operacionales

Dado que nuestra aplicación da mucha libertad al momento de efectuar consultas, también es muy propensa a que ocurran errores operacionales. Por ejemplo, un componente de software mal programado puede realizar consultas al API por fechas inexistentes o futuras. Es por esto que es de vital importancia que el manejo de errores operacionales sea preciso y lo más provechoso posible para el usuario.

Este tópico fue uno de los que dedicó más investigación y apoyo de implementadores experimentados. La razón es que en este tipo de aplicaciones no hay ningún estándar establecido, lo cual a fin de cuentas nos obligó a tomar muchas decisiones arbitrarias.

Luego de una investigación que se prolongo por cerca de una semana, se pudo observar que las principales tendencias para el manejo de errores en APIs famosas como la Twitter eran dos:

1. Manejar los errores operacionales haciendo uso solamente de los códigos de retorno provistos por el protocolo HTTP.
2. Entregar siempre un código de retorno HTTP 200 (OK) y los errores operacionales y mensaje entregarlos en la representación negociada en el requerimiento.

La primera opción parece ser la más correcta, dado que va a acorde a REST desde que no pone capas adicionales sobre la comunicación más que el mismo protocolo HTTP. Sin embargo, tenemos

como gran problema el hecho de que los códigos HTTP no fueron pensados para ser usados en una aplicación como una API. Luego se torna muy complejo decidir que código retornar cuando por ejemplo, se acabó la cuota de acceso del usuario dueño de la API Key.

La segunda opción se ve mucho más flexible, dado que los códigos de retorno son totalmente arbitrarios al igual que los mensajes asociados a dicho código. Sin embargo, optar por esta opción involucraba estar fuera de todo estándar además de romper con REST. Esto a fin de cuentas era sacrificar gran parte de la interoperabilidad de la información. Sumado a los argumentos anteriormente expuestos, se recibió como retroalimentación de los mismos implementadores que usan APIs, que muchas veces para llamar información se usan scripts que sólo son capaces de detectar que hubo errores en una consulta mediante el código de retorno HTTP.

Luego de un poco de investigación, se descubrió que los códigos de retorno HTTP no son rígidos y pueden ser extendidos sin problemas a otras necesidades.

Finalmente, y dado que ambas posiciones no son excluyentes, se optó por una posición híbrida entre ambas. Esto significó:

- Cada vez que una consulta es realizada exitósamente, se entrega el recurso solicitado en la representación negociada en el requerimiento. En la cabecera de la respuesta HTTP se entrega un código de retorno HTTP 200 (OK).

- Cada vez que ocurre un error operacional en la consulta, se entrega el código HTTP de error correspondiente al error ocurrido en la cabecera de la respuesta HTTP. Si el error HTTP no existe para el caso dado, se entrega un código de error extendido, el cual fue creado por nosotros. Adicional al código HTTP, se entrega una estructura de error en la representación negociada en el requerimiento, con el código propietario de error del API de la SBIF junto con un mensaje explicativo. La idea tras esta dualidad es que muchos errores del API caen dentro de un mismo error HTTP (un 404 por ejemplo), luego el código de error del API junto con el mensaje son una especificación del mismo error.

El detalle de todos los códigos de error entregados por el API de la SBIF se encuentra disponible en Anexos en la Sección 8.2.

5.5.9. Interoperabilidad de la solución

Luego de la primera presentación de la aplicación en público, donde solo se encontraba operativo el indicador financiero de la UF en la API (ver Sección 6.1), uno de los participantes de la presentación nos señaló que debíamos inscribir nuestros documentos de salida XML en el administrador de esquemas y meta datos del Gobierno[15]. El administrador de esquemas y meta datos es el Organismo Público encargado de administrar y mantener operativo, el procedimiento de inscripción de esquemas basales y documentales por parte de los órganos de la Administración del Estado, su evaluación técnica y posterior publicación en el Repositorio de Acceso Público de Esquemas de Gobierno. Esta organización es la encargada de velar por la inter operabilidad de los órganos de la Administración del Estado por medio del documento electrónico.

Actualmente el proyecto tiene en proceso las solicitudes desde la 519 a la 531 en el administrador de esquemas y meta datos. Estas solicitudes ya han aprobado la etapa de evaluación técnica, la cual

[15] http://www.aem.gob.cl/

consiste en verificar la correctitud de los esquemas y que los documentos XML del API sean validados por los mismos.

Todos los esquemas se encuentran disponibles y documentados en el sitio web de la API[16].

[16] http://api.sbif.cl/API/documentacion

6. Validación de la solución

Para este desarrollo en particular, la validación de la solución es sumamente importante. Esto se debe a que el proyecto tuvo alto grado de incertidumbre desde un comienzo debido al desconocimiento técnico tanto del memorista como del cliente sobre la aplicación a implementar. Esto llevó a tomar muchas decisiones arbitrarias en el camino que con el tiempo necesitaron ser validadas. En este apartado se revisan diferentes aspectos con el fin de mostrar que el desarrollo de esta memoria efectivamente fue una solución al problema planteado. Las formas que se idearon para validar nuestro trabajo fueron dos:

- Participar en diferentes jornadas donde se pudiese exponer el desarrollo y obtener retroalimentación.
- Publicitar el API en cada jornada con el fin de conseguir nuevos desarrolladores interesados en participar como testers en el proyecto.

6.1. Exposición del desarrollo

Durante los meses de desarrollo se expuso el proyecto del API en cuatro oportunidades, las cuales se detallan a continuación:

- Primera exposición en dependencias de la SBIF el día 9 de Septiembre del 2010. La lista de participantes está disponible en Anexos Sección 8.4.
- Exposición como conferencista en Barcamp 2010[17], el cual fue celebrado en dependencias del DUOC UC sede Alonso Ovalle el día 6 de Noviembre del 2010.
- Exposición en el Segundo seminario de publicación abierta de datos[18], el cual fue celebrado en el auditorio del DCC de la Universidad de Chile el día 10 de Noviembre del 2010.
- Exposición como conferencista en el evento de emprendimiento digital Webprendedor 2010[19], el cual fue celebrado el día 20 de Noviembre del 2010 en el aula magna del campus San Joaquín de la Pontificia Universidad Católica de Chile.

6.2. Testeo por parte de usuarios reales

A medida que el proyecto era expuesto en las instancias anteriormente mencionadas, se captó clientes interesados en usar el API como se esperó en un comienzo. Esto fue muy valioso dado que nos permitió contar con un equipo externo de personas que testearon el API previo lanzamiento de la aplicación al publico (1 de Diciembre del 2010). El retroalimentación que se nos entregó fue muy valioso para la implementación del proyecto, dado que fue el gestor de muchos cambios importantes no previstos por el memorista, como por ejemplo, la entrega de información usando JSONP, la cual fue sugerida por Héctor Vergara.

[17] http://barcamp.cl/
[18] http://chile-datos.cl/?q=node/34
[19] http://eventos.webprendedor.com/2010

Destacan dentro de las personas que entregaron retroalimentación la participación de Hector Vergara y Alvaro Graves, quienes con sus valiosos aportes y amplios conocimientos del tema hicieron que el API de la SBIF tome el rumbo correcto.

La lista completa del equipo de testers de la aplicación se encuentra en Anexos Sección 8.5.

7. Conclusiones

7.1. Resultados del proceso de desarrollo

Actualmente el API de la SBIF se encuentra en un ambiente de producción con un número creciente de usuarios con un API Key asignada. Esta API está entregando información financiera en formatos estándar XML y JSON, con la posibilidad de usar la técnica de obtención de información llamada JSONP.

Antes del lanzamiento de la aplicación al público, se hizo una revisión de la calidad de la información almacenada en Base de Datos por medio de implementaciones de visualización de datos, con lo cual fue posible encontrar registros redundantes o mal ingresados. Luego está garantizada la calidad de la información disponibilizada por la aplicación.

La disponibilidad de la aplicación lamentablemente no se puede asegurar en estos momentos dada la existencia de problemas en la red física de las dependencias de la SBIF, los cuales desencadenan problemas de comunicación entre el servidor de aplicación y el motor de base de datos. Esta es la razón por la cual hoy existe intermitencia en el servicio. Se espera que estos problemas se solucionen antes del 31 de Diciembre del 2010 con lo cual la disponibilidad del API debería quedar asegurada.

Para apoyar y fomentar los desarrollos ligados al API, se ha dispuesto de una nutrida documentación en el sitio web *http://api.sbif.cl/API*. Dentro del sitio se encuentra todo el material necesario para construir consultas y aplicaciones de visualización por medio de las librerías de la API de visualización de Google.

7.1.1. Recuento de objetivos logrados

Revisando los objetivos expuestos en la Sección 5.3, es posible hacer una retrospectiva sobre el éxito del proyecto en base a los objetivos.

El objetivo general está cumplido y la aplicación se encuentra actualmente en funcionamiento sirviendo a un gran número de usuarios reales.

Si se revisan los objetivos específicos, nos daremos cuenta que casi todos se han cumplido o han cambiado de forma debido a decisiones estratégicas tomadas en medio del desarrollo. Por ejemplo, el segundo objetivo específico que habla de modificar el servicio web existente cambió, dado que no fue factible su modificación y se re-implemento completamente por las razones señaladas en la Sección 5.4.1.

El objetivo específico que habla sobre las pruebas de carga de la aplicación no pudo ser llevado a puerto dado los problemas de la red física de la SBIF señalados en la Sección anterior. Luego el cumplimiento de este objetivo está pendiente con prioridad alta para cuando esta situación esté normalizada.

Finalmente se puede concluir que los objetivo fueron logrados en un alto grado, lo cual permite hablar de un proyecto exitoso.

7.1.2. Decisiones estratégicas que alteraron el plan de actividades original

Si bien existía una carta Gantt durante el desarrollo, la cual contenía una lista de actividades bastante detallada, no fue posible determinar a priori con exactitud cuales eran las actividades que se desarrollarían. Luego en medio del desarrollo comenzaron a surgir una serie de actividades

no previstas, las cuales significaron hacer uso de tiempo destinado a otras actividades, según la metodología expuesta en la Sección 5.4.3.

Las actividades no contempladas en la planificación original y el tiempo que tomó su desarrollo son las que enumeran a continuación:

1. Acomodar el servidor JBoss para ejecutar proyectos Grails. Tiempo aproximado: 2 semanas.
2. Elaboración e inscripción de esquemas para las respuestas en XML en el administrador de esquemas y meta datos del Gobierno. Tiempo aproximado: 2 semanas.
3. Hacer que la aplicación pueda entregar respuestas usando JSONP. Tiempo aproximado: 2 días.

7.2. Impacto de la implantación de la solución

7.2.1. Impacto dentro de la SBIF

El impacto generado dentro de los desarrollos de la SBIF es el que se ilustra en la Figura 14. Aquí se ven 2 flujos operacionales que son los que se han comenzado a generar. Primero tenemos el flujo operacional representado por los pasos desde el 1 al 5 en la figura, donde los nuevos módulos pertenecientes al proyecto SBIFWEB comenzaran a hacer uso del API para obtener información desde la base de datos en vez de hacerlo de forma directa. Esto genera dinamismo en la visualización de la información al facilitar la integración con APIs de visualización, las cuales reciben datos en formatos estándar como XML o JSON.

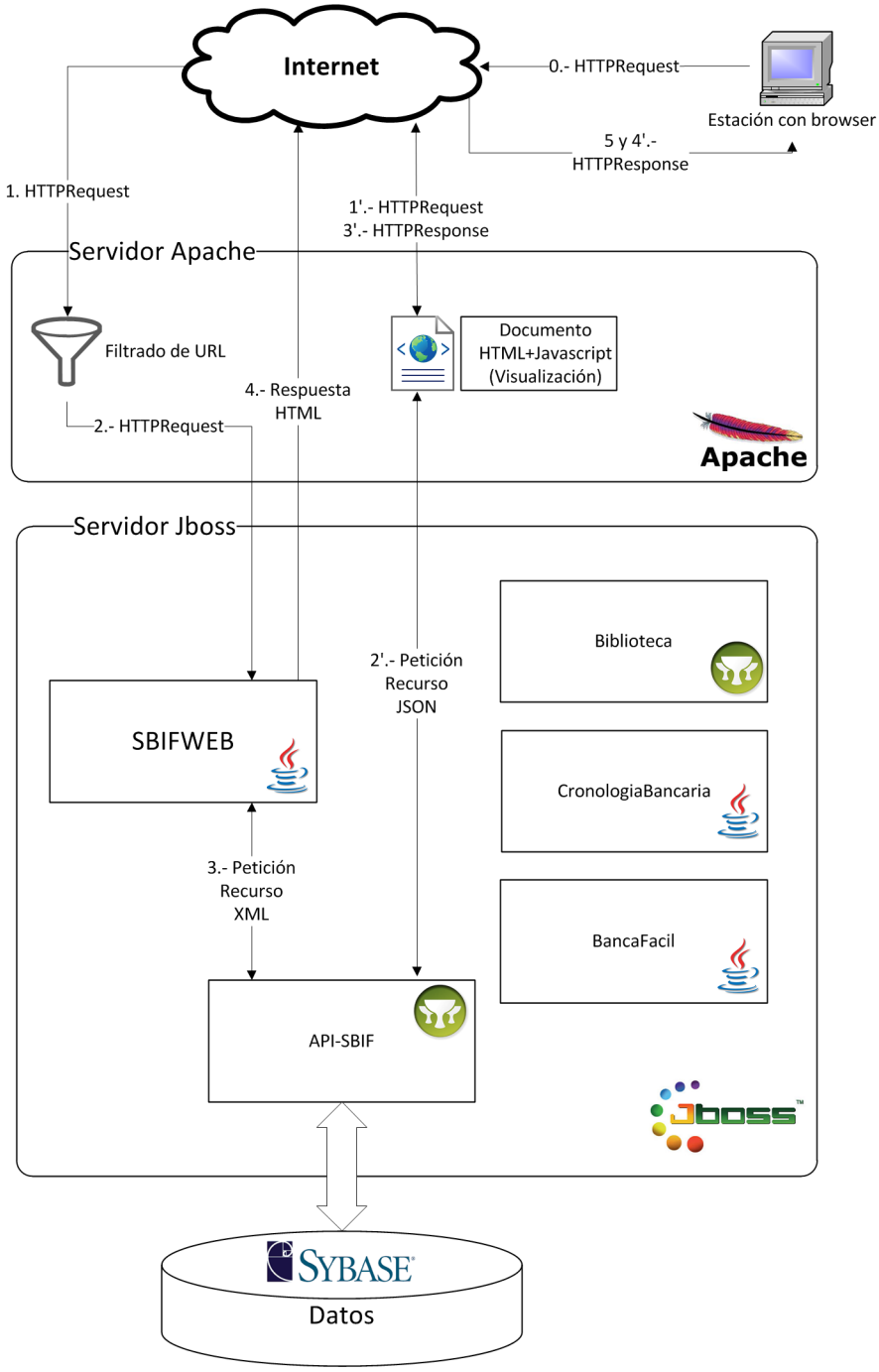

Figura 14: Diagrama de la arquitectura lógica de los desarrollos de la SBIF luego de la implantación del API

Adicional a este flujo está el flujo determinado por los pasos desde el 1'al 4'en la figura. Este es el flujo operacional que se da con los usuarios externos haciendo consultas al API de forma directa a través de internet. Las repercusiones que genera este flujo operacional son tráfico adicional tanto en las redes como en los servidores, dado que el nicho al cual apunta el API es diferente al cual apunta el sitio web.

7.2.2. Impacto en desarrollos externos

El impacto generado en los desarrollos fuera de la SBIF son los que se muestran en la Figura15. Los datos ahora se obtienen desde fuera mediante el uso de la API en vez de parseando directamente el sitio web de la SBIF(o alguna otra institución que los posea, como el Banco Central). Esto hace que el modelo de desarrollo externo haya evolucionado haciendo que las aplicaciones externas ganen robustez y disminuyan la dificultad de implementación, portabilidad y mantención.

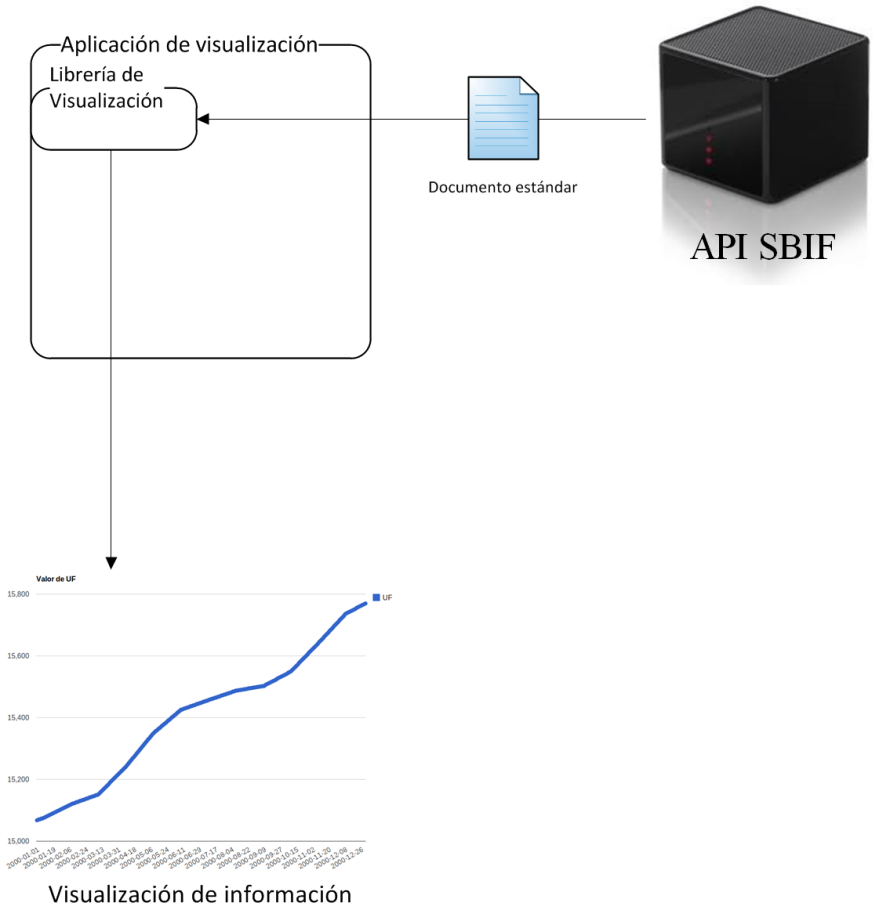

Figura 15: Diagrama de la arquitectura lógica de los desarrollos externos luego de implantada la solución

7.3. Pasos siguientes en el desarrollo del proyecto

Luego de acabada esta etapa del desarrollo, se vislumbran los pasos a seguir en el desarrollo. Estas etapas posteriores son las que se describen a continuación:

- Implementación de OAUTH: Re-considerar la implementación de OAUTH como sistema de acceso al API, dado que presenta numerosas características que en un futuro pueden enriquecer la calidad de servicio de la aplicación.
- Implementación de RDF: Instalar un motor de RDF alternativo a los métodos tradicionales del API, con el fin de proveer mayor libertad de consultas a usuarios expertos así como garantizar inter operabilidad con otros proveedores de datos.
- Optimización operacional: Crear un nuevo ambiente de producción separado del ambiente actual de la SBIF, para la ejecución del API. Esto nos permite disminuir el costo de una caída del API, la cual se espera recibirá una alta demanda en un poco tiempo más. Actualmente una caída del servidor de aplicación debido a las consultas del API hace caer todos los sitios web de la SBIF, lo cual es un tema muy sensible.

Referencias

[1] Papazoglou, M.P. and D. Georgakopoulos (2003). Service-Oriented Computing. In: Communications of the ACM, 46(10):25-28, October 2003.

[2] Faouzi Kamoun. The Convergence of Business Process Management and Service Oriented Architecture. College of Information Technology. University of Dubai. June 2007.

[3] Hanyang Luo, Jinling Gao, Hanyang Luo. Web Data Extraction Based on XBRL-GL Taxonomy. Vol. 1, pp.358-361, 2009 Asia-Pacific Conference on Information Processing, 2009.

[4] Walid Gaaloul, Sami Bhiri, and Mohsen Rouached. Event-Based Design and Runtime Verification of Composite Service Transactional Behavior. IEEE transactions on services computing , Vol. 3, NO. 1, pp. 32-45. January-March 2010.

[5] Ju Long , Michael J. Yuan , Andrew B. Whinston. Securing a New Era of Financial Services. pp. 15-21. July 2003.

[6] Anoop Singhal. Web Services Security: Challenges and Techniques. Computer Security Division, IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07). 2007.

8. Anexos

8.1. Consultas y respuestas

8.1.1. Ejemplo 1

Consulta

```
http://api.sbif.cl/api-sbif/recursos
  /balances/2009/12/instituciones
  ?apikey=SBIF9990SBIF44b7SBIF7f4c5a537d02358e1099
  &formato=xml
```

Resultado

Entrega un archivo en formato xml con el listado de instituciones vigentes a diciembre de 2009.

Código de Resultado

```
<ReportesBancarios
  xsi:schemaLocation="http://api.sbif.cl/XMLSchema
  ReportesBancarios-v1.0.xsd"
  SchemaVersion="1.0">
  <DescripcionescodigosDeInstituciones>
    <DescripcionIFI>
      <codigoInstitucion>
        001
      </codigoInstitucion>
      <NombreInstitucion>
        BANCO DE CHILE
      </NombreInstitucion>
    </DescripcionIFI>
                .
                .
                .
  </DescripcionescodigosDeInstituciones>
</ReportesBancarios>
```

8.1.2. Ejemplo 2

Consulta

```
http://api.sbif.cl/api-sbif/recursos
  /balances/2009/instituciones/012
  ?apikey=SBIF9990SBIF44b7SBIF7f4c5a537d02358e1099
  &formato=xml
```

Resultado

Entrega un archivo en formato xml con el balance del año 2009 para el Banco del Estado (Código 012).

Código de Resultado

```
<ReportesBancarios
  xsi:schemaLocation="http://api.sbif.cl/XMLSchema
  ReportesBancarios-v1.0.xsd"
  SchemaVersion="1.0">
  <codigosBalances>
    <codigoBalanceIFI>
      <codigoCuenta>
        1000000
      </codigoCuenta>
      <DescripcionCuenta>
        ACTIVOS
      </DescripcionCuenta>
      <codigoInstitucion>
        012
      </codigoInstitucion>
      <NombreInstitucion>
        BANCO DEL ESTADO DE CHILE
      </NombreInstitucion>
      <Anho>
        2009
      </Anho>
      <Mes>
        1
      </Mes>
      <MonedaChilenaNoReajustable>
        6410679,00
      </MonedaChilenaNoReajustable>
      <MonedaReajustablePorIPC>
        6763382,00
```

```
      </MonedaReajustablePorIPC>
      <MonedaReajustablePorTipoDeCambio>
        26579,00
      </MonedaReajustablePorTipoDeCambio>
      <MonedaExtranjera>
        1865378,00
      </MonedaExtranjera>
      <MonedaReajustable/>
      <MonedaTotal>
        15066018,00
      </MonedaTotal>
    </codigoBalanceIFI>
                .
               .
                .
  </codigosBalances>
</ReportesBancarios>
```

8.1.3. Ejemplo 3

Consulta

```
http://api.sbif.cl/api-sbif/recursos
  /balances/2009/01/instituciones/012
  ?apikey=SBIF9990SBIF44b7SBIF7f4c5a537d02358e1099
  &formato=xml
```

Resultado

Entrega un archivo en formato xml con el balance del mes de enero del año 2009 para el Banco del Estado (Código 012).

Código de Resultado

```
<ReportesBancarios
  xsi:schemaLocation="http://api.sbif.cl/XMLSchema
  ReportesBancarios-v1.0.xsd"
  SchemaVersion="1.0">
  <codigosBalances>
    <codigoBalanceIFI>
      <codigoCuenta>
        1000000
      </codigoCuenta>
      <DescripcionCuenta>
        ACTIVOS
      </DescripcionCuenta>
      <codigoInstitucion>
        012
      </codigoInstitucion>
      <NombreInstitucion>
        BANCO DEL ESTADO DE CHILE
      </NombreInstitucion>
      <Anho>
        2009
      </Anho>
      <Mes>
        1
      </Mes>
      <MonedaChilenaNoReajustable>
        6410679,00
      </MonedaChilenaNoReajustable>
      <MonedaReajustablePorIPC>
        6763382,00
```

```
      </MonedaReajustablePorIPC>
      <MonedaReajustablePorTipoDeCambio>
        26579,00
      </MonedaReajustablePorTipoDeCambio>
      <MonedaExtranjera>
        1865378,00
      </MonedaExtranjera>
      <MonedaReajustable/>
      <MonedaTotal>
        15066018,00
      </MonedaTotal>
    </codigoBalanceIFI>
                .
                .
                .
  </codigosBalances>
</ReportesBancarios>
```

8.1.4. Ejemplo 4

Consulta

```
http://api.sbif.cl/api-sbif/recursos
  /balances/periodo3/06/instituciones/012
  ?apikey=SBIF9990SBIF44b7SBIF7f4c5a537d02358e1099
  &formato=xml
```

Resultado

Entrega un archivo en formato xml con el balance del mes de junio de 2009 y 2010 (ambos comprendidos dentro del periodo3) para el Banco del Estado (Código 012).

Código de Resultado

```
<ReportesBancarios
  xsi:schemaLocation="http://api.sbif.cl/XMLSchema
  ReportesBancarios-v1.0.xsd"
  SchemaVersion="1.0">
  <codigosBalances>
    <codigoBalanceIFI>
      <codigoCuenta>
        1000000
      </codigoCuenta>
      <DescripcionCuenta>
        ACTIVOS
      </DescripcionCuenta>
      <codigoInstitucion>
        012
      </codigoInstitucion>
      <NombreInstitucion>
        BANCODELESTADODECHILE
      </NombreInstitucion>
      <Anho>
        2009
      </Anho>
      <Mes>
        6
      </Mes>
      <MonedaChilenaNoReajustable>
        6944071,00
      </MonedaChilenaNoReajustable>
      <MonedaReajustablePorIPC>
        6640698,00
```

```
      </MonedaReajustablePorIPC>
      <MonedaReajustablePorTipoDeCambio>
        21870,00
      </MonedaReajustablePorTipoDeCambio>
      <MonedaExtranjera>
        1806646,00
      </MonedaExtranjera>
      <MonedaReajustable/>
      <MonedaTotal>
        15413285,00
      </MonedaTotal>
    </codigoBalanceIFI>
                .
                .
                .
  </codigosBalances>
</ReportesBancarios>
```

8.1.5. Ejemplo 5

Consulta

```
http://api.sbif.cl/api-sbif/recursos
  /balances/2009/12/cuentas
  ?apikey=SBIF9990SBIF44b7SBIF7f4c5a537d02358e1099
  &formato=xml
```

Resultado

Entrega un archivo en formato xml con el listado de las Cuentas con su Código y Descripción para el balance del mes de diciembre de 2009.

Código de Resultado:

```
<ReportesBancarios
  xsi:schemaLocation="http://api.sbif.cl/XMLSchema
  ReportesBancarios-v1.0.xsd"
  SchemaVersion="1.0">
  <DescripcionescodigosContables>
    <descripcioncodigoIFI>
      <codigoCuenta>
        1000000
      </codigoCuenta>
      <DescripcionCuenta>
        ACTIVOS
      </DescripcionCuenta>
    </descripcioncodigoIFI>
          .
          .
          .
  </DescripcionescodigosContables>
</ReportesBancarios>
```

8.1.6. Ejemplo 6

Consulta

```
http://api.sbif.cl/api-sbif/recursos
  /balances/2009/cuentas/1000000/instituciones/012
  ?apikey=SBIF9990SBIF44b7SBIF7f4c5a537d02358e1099
  &formato=xml
```

Resultado

Entrega un archivo en formato xml con los valores correspondientes para la Cuenta ACTIVOS (Código 1000000) para el Banco del Estado (Código 012) para cada mes del año 2009.

Código de Resultado

```
<ReportesBancarios
  xsi:schemaLocation="http://api.sbif.cl/XMLSchema
  ReportesBancarios-v1.0.xsd"
  SchemaVersion="1.0">
  <codigosBalances>
    <codigoBalanceIFI>
      <codigoCuenta>
        1000000
      </codigoCuenta>
      <DescripcionCuenta>
        ACTIVOS
      </DescripcionCuenta>
      <codigoInstitucion>
        012
      </codigoInstitucion>
      <NombreInstitucion>
        BANCODELESTADODECHILE
      </NombreInstitucion>
      <Anho>
        2009
      </Anho>
      <Mes>
        1
      </Mes>
      <MonedaChilenaNoReajustable>
        6410679,00
      </MonedaChilenaNoReajustable>
      <MonedaReajustablePorIPC>
        6763382,00
```

```
      </MonedaReajustablePorIPC>
      <MonedaReajustablePorTipoDeCambio>
        26579,00
      </MonedaReajustablePorTipoDeCambio>
      <MonedaExtranjera>
        1865378,00
      </MonedaExtranjera>
      <MonedaReajustable/>
      <MonedaTotal>
        15066018,00
      </MonedaTotal>
    </codigoBalanceIFI>
          .
          .
          .
  </codigosBalances>
</ReportesBancarios>
```

8.1.7. Ejemplo 7

Consulta

```
http://api.sbif.cl/api-sbif/recursos
  /balances/2009/12/cuentas/1000000/instituciones/012
  ?apikey=SBIF9990SBIF44b7SBIF7f4c5a537d02358e1099
  &formato=xml
```

Resultado

Entrega un archivo en formato xml con los valores correspondientes para la Cuenta ACTIVOS (Código 1000000) para el Banco del Estado (Código 012) para el mes de diciembre del año 2009.

Código de Resultado

```
<ReportesBancarios
  xsi:schemaLocation="http://api.sbif.cl/XMLSchema
  ReportesBancarios-v1.0.xsd" SchemaVersion="1.0">
  <codigosBalances>
    <codigoBalanceIFI>
      <codigoCuenta>
        1000000
      </codigoCuenta>
      <DescripcionCuenta>
        ACTIVOS
      </DescripcionCuenta>
      <codigoInstitucion>
        012
      </codigoInstitucion>
      <NombreInstitucion>
        BANCODELESTADODECHILE
      </NombreInstitucion>
      <Anho>
        2009
      </Anho>
      <Mes>
        12
      </Mes>
      <MonedaChilenaNoReajustable>
        7976294,00
      </MonedaChilenaNoReajustable>
      <MonedaReajustablePorIPC>
        7092272,00
      </MonedaReajustablePorIPC>
```

```
      <MonedaReajustablePorTipoDeCambio>
        16557,00
      </MonedaReajustablePorTipoDeCambio>
      <MonedaExtranjera>
        1808405,00
      </MonedaExtranjera>
      <MonedaReajustable/>
      <MonedaTotal>
        16893528,00
      </MonedaTotal>
    </codigoBalanceIFI>
  </codigosBalances>
</ReportesBancarios>
```

8.1.8. Ejemplo 8

Consulta

```
http://api.sbif.cl/api-sbif/recursos
  /balances/periodo3/06/cuentas/1000000/instituciones/012
  ?apikey=SBIF9990SBIF44b7SBIF7f4c5a537d02358e1099
  &formato=xml
```

Resultado

Entrega un archivo en formato xml con los valores correspondientes para la Cuenta ACTIVOS (Código 1000000) para el Banco del Estado (Código 012) para el mes de junio de los años 2009 y 2010 (ambos incluidos en el periodo3).

Código de Resultado

```
<ReportesBancarios
  xsi:schemaLocation="http://api.sbif.cl/XMLSchema
  ReportesBancarios-v1.0.xsd"
  SchemaVersion="1.0">
  <codigosBalances>
    <codigoBalanceIFI>
      <codigoCuenta>
        1000000
      </codigoCuenta>
      <DescripcionCuenta>
        ACTIVOS
      </DescripcionCuenta>
      <codigoInstitucion>
        012
      </codigoInstitucion>
      <NombreInstitucion>
        BANCODELESTADODECHILE
      </NombreInstitucion>
      <Anho>
        2009
      </Anho>
      <Mes>
        6
      </Mes>
      <MonedaChilenaNoReajustable>
        6944071,00
      </MonedaChilenaNoReajustable>
      <MonedaReajustablePorIPC>
```

```
    6640698,00
  </MonedaReajustablePorIPC>
  <MonedaReajustablePorTipoDeCambio>
    21870,00
  </MonedaReajustablePorTipoDeCambio>
  <MonedaExtranjera>
    1806646,00
  </MonedaExtranjera>
  <MonedaReajustable/>
  <MonedaTotal>
    15413285,00
  </MonedaTotal>
</codigoBalanceIFI>
<codigoBalanceIFI>
  <codigoCuenta>
    1000000
  </codigoCuenta>
  <DescripcionCuenta>
    ACTIVOS
  </DescripcionCuenta>
  <codigoInstitucion>
    012
  </codigoInstitucion>
  <NombreInstitucion>
    BANCODELESTADODECHILE
  </NombreInstitucion>
  <Anho>
    2010
  </Anho>
  <Mes>
    6
  </Mes>
  <MonedaChilenaNoReajustable>
   7496930,00
  </MonedaChilenaNoReajustable>
  <MonedaReajustablePorIPC>
    7386168,00
  </MonedaReajustablePorIPC>
  <MonedaReajustablePorTipoDeCambio>
    15870,00
  </MonedaReajustablePorTipoDeCambio>
  <MonedaExtranjera>
    2444909,00
  </MonedaExtranjera>
  <MonedaReajustable/>
```

```
      <MonedaTotal>
        17343877,00
      </MonedaTotal>
    </codigoBalanceIFI>
  </codigosBalances>
</ReportesBancarios>
```

8.2. Códigos de error del API de la SBIF

8.2.1. Rango de errores del 60 - 69

Errores operacionales relacionados al servidor que provee los recursos

Código de error API.SBIF:	60
Código de respuesta http:	503 (Service Unavailable)
Mensaje:	Momentáneamente no es posible acceder a el o los recursos solicitados, por favor intente nuevamente.
Detalle:	Nuestro servidor presenta problemas técnicos que dificultan o imposibilitan la entrega de resultados. Estos errores pueden ser de acceso a la base de datos, servidores caídos, etc.

Código de error API.SBIF:	61 (uso administrativo, no visible desde fuera)
Código de respuesta http:	503 (Service Unavailable)
Mensaje:	Momentáneamente no es posible crear una nueva cuenta, por favor intente nuevamente.
Detalle:	Nuestro servidor presenta problemas técnicos que dificultan o imposibilitan la entrega de resultados. Estos errores pueden ser de acceso a la base de datos, servidores caídos, etc.

Código de error API.SBIF:	62 (uso administrativo, no visible desde fuera)
Código de respuesta http:	503 (Service Unavailable)
Mensaje:	Momentáneamente no es posible renovar la cuota de esta cuenta, por favor intente nuevamente.
Detalle:	Nuestro servidor presenta problemas técnicos que dificultan o imposibilitan la entrega de resultados. Estos errores pueden ser de acceso a la base de datos, servidores caídos, etc

Código de error API.SBIF:	63 (uso administrativo, no visible desde fuera)
Código de respuesta http:	503 (Service Unavailable)
Mensaje:	Momentáneamente no es posible renovar la API Key de esta cuenta, por favor intente nuevamente.
Detalle:	Nuestro servidor presenta problemas técnicos que dificultan o imposibilitan la entrega de resultados. Estos errores pueden ser de acceso a la base de datos, servidores caídos, etc

8.2.2. Rango de errores del 70 - 79

Errores relacionados al ingreso de parámetros por parte del usuario de la API.

Código de error API.SBIF:	70
Código de respuesta http:	404 (not found)
Mensaje:	Alguno de los campos de la fecha presenta errores tipográfico
Detalle:	Alguno de los valores ingresados como fecha de consulta presentan errores tipográficos que los dejan sin sentido semántico.

Código de error API.SBIF:	71
Código de respuesta http:	404 (not found)
Mensaje:	Fecha especificada no es valida
Detalle:	Si bien los valores ingresados como fecha no presentan errores tipográficos, son valores que no tienen sentido en la realidad.

Código de error API.SBIF:	72
Código de respuesta http:	404 (not found)
Mensaje:	Fecha especificada es posterior a la fecha actual
Detalle:	Los valores ingresados como fecha representan una fecha posterior al día actual.

Código de error API.SBIF:	73
Código de respuesta http:	404 (not found)
Mensaje:	Tipo de retorno especificado no es provisto por el API o posee errores tipográficos
Detalle:	El tipo de retorno pedido al API actualmente no está soportado o es un tipo de retorno erróneo.

Código de error API.SBIF:	74
Código de respuesta http:	404 (not found)
Mensaje:	Cuenta especificada no es valida
Detalle:	El valor ingresado como código de cuenta es erróneo.

Código de error API.SBIF:	75
Código de respuesta http:	404 (not found)
Mensaje:	Institución especificada no es valida
Detalle:	El valor ingresado como código de cuenta es erróneo.

8.2.3. Rango de errores del 80 - 89

Errores relacionados a los datos en las bases de datos de la SBIF.

Código de error API.SBIF:	80
Código de respuesta http:	404 (not found)
Mensaje:	No hay datos disponibles para los parámetros ingresados
Detalle:	La consulta no retorna datos.

Código de error API.SBIF:	81
Código de respuesta http:	404 (not found)
Mensaje:	El recurso correspondiente al día actual aun no ha sido cargado
Detalle:	Retornado cuando una consulta realizada implícitamente sobre la fecha actual no retorna valores.

8.2.4. Rango de errores del 90 - 99

Errores relacionados a la autentificación/autorización a la API de la SBIF.

Código de error API.SBIF:	90
Código de respuesta http:	420
Mensaje:	Su cuota de acceso al servicio ha sido superada
Detalle:	Retornado cuando un usuario intenta acceder a un servicio usando un API Key que ha superado su cuota de acceso.

Código de error API.SBIF:	91
Código de respuesta http:	421
Mensaje:	API Key no valida
Detalle:	Retornado cuando un usuario intenta acceder a un servicio usando un API Key que no existe dentro de los registros de la SBIF.

Código de error API.SBIF:	92
Código de respuesta http:	422
Mensaje:	API Key no ha sido suministrada
Detalle:	Retornado cuando un usuario intenta acceder a un recurso sin suministrar un API Key.

8.3. Manual de uso de la solución

8.3.1. Uso de parámetros GET

La API fue implementada para que las URIs puedan ser utilizadas haciendo uso de parámetros GET complementarios a la consulta. Esta decisión se tomó con el objetivo de indicar las características de la petición que se ejecuta de forma separada del recursos solicitado.

Los parámetros utilizados en la API SBIF son los siguientes:

- *apikey*: permite incluir el valor de la clave que se haya entregado al usuario.
- *formato*: permite indicar si el archivo de salida será en formato JSON o XML; si no se utiliza este parámetro, la salida será en XML. Los parámetros se pueden indicar en mayúsculas (JSON—XML) o en minúsculas (json—xml).
- *callback*: este parámetro es opcional y sólo se emplea cuando se pide un archivo de salida del tipo JSON (para obtener la respuesta usando JSONP). En caso de solicitarse una salida JSON y no especificarse una función de callback, se retorna JSON normal.

En cada uno de los recursos que se detallan a continuación se explica de manera más específica la forma de incluirlos.

8.3.2. Uso de parámetros inmersos en las URIs

Muchas de las URIS deben incluir la fechas en su contenido, por lo que se debe escribir el año, el mes y el día, si corresponde, para obtener la información. En cada caso, la inclusión se hace de la siguiente manera:

- anho: se deben incluir cuatro números del año en el formato AAAA
- mes: se deben incluir dos números del mes en el formato MM
- dia: se deben incluir dos números del día en el formato DD

8.3.3. Códigos de Error

Cualquier llamado a los recursos que no cumpla con la entrega de la API Key o que tenga parámetros errados, generará un código de error perteneciente al protocolo HTTP, un código de error propietario de la API y un mensaje explicativo del error. Esto quiere decir, un error generará un archivo XML, JSON o JSONP según sea la negociación, con la información correspondiente al código propietario y al mensaje. El código HTTP de error viene en la cabecera de la respuesta HTTP.

8.3.4. Recursos disponibles por la API

Recursos referentes a la UF

UF del día actual:

```
http://api.sbif.cl/api-sbif/recursos
  /uf
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de un año:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de un año y mes:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de una fecha específica:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de años posteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/posteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de meses posteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
```

```
  /uf/posteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de fechas posteriores a la fecha:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/posteriores/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de años anteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/anteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de meses anteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/anteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de meses anteriores a la fecha:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/anteriores/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de período definido por fechas:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/periodo/<anho>/<mes>/<dia>/<anho2>/<mes2>/<dia2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
```

```
  &callback=<funcion_de_callback>
```

UF de período definido por meses:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/periodo/<anho>/<mes>/<anho2>/<mes2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UF de período definido por años:

```
http://api.sbif.cl/api-sbif/recursos
  /uf/periodo/<anho>/<anho2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Recursos referentes al IPC

IPC del mes actual:

```
http://api.sbif.cl/api-sbif/recursos
  /ipc
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

IPC de un año:

```
http://api.sbif.cl/api-sbif/recursos
  /ipc/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

IPC de un año y mes:

```
http://api.sbif.cl/api-sbif/recursos
  /ipc/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

IPC de años posteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /ipc/posteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

IPC de meses posteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /ipc/posteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

IPC de años anteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /ipc/anteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

IPC de meses anteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /ipc/anteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

IPC de período definido por meses:

```
http://api.sbif.cl/api-sbif/recursos
  /ipc/periodo/<anho>/<mes>/<anho2>/<mes2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

IPC de período definido por años:

```
http://api.sbif.cl/api-sbif/recursos
  /ipc/periodo/<anho>/<anho2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Recursos referentes a la UTM

UTM del mes actual:

```
http://api.sbif.cl/api-sbif/recursos
  /utm
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UTM de un año:

```
http://api.sbif.cl/api-sbif/recursos
  /utm/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UTM de un año y mes:

```
http://api.sbif.cl/api-sbif/recursos
  /utm/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UTM de años posteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /utm/posteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UTM de meses posteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /utm/posteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UTM de años anteriores al año:

```
  http://api.sbif.cl/api-sbif/recursos
  /utm/anteriores/<anho>
  ?apikey=<api_key>
```

```
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UTM de meses anteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /utm/anteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UTM de período definido por meses:

```
http://api.sbif.cl/api-sbif/recursos
  /utm/periodo/<anho>/<mes>/<anho2>/<mes2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

UTM de período definido por años:

```
http://api.sbif.cl/api-sbif/recursos
  /utm/periodo/<anho>/<anho2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Recursos referentes al precio del Dolar

Dólar del día actual:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de un año:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de un año y mes:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de una fecha específica:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de años posteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/posteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de meses posteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/posteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de fechas posteriores a la fecha:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/posteriores/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de años anteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/anteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de meses anteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/anteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de meses anteriores a la fecha:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/anteriores/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de período definido por fechas:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/periodo/<anho>/<mes>/<dia>/<anho2>/<mes2>/<dia2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de período definido por meses:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/periodo/<anho>/<mes>/<anho2>/<mes2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Dólar de período definido por años:

```
http://api.sbif.cl/api-sbif/recursos
  /dolar/periodo/<anho>/<anho2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Recursos referentes al precio del Euro

Euro del día actual:

```
http://api.sbif.cl/api-sbif/recursos
  /euro
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de un año:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de un año y mes:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de una fecha específica:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de años posteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/posteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de meses posteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/posteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de meses posteriores a la fecha:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/posteriores/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de años anteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/anteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de meses anteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/anteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de meses anteriores a la fecha:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/anteriores/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de período definido por fechas:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/periodo/<anho>/<mes>/<dia>/<anho2>/<mes2>/<dia2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de período definido por meses:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/periodo/<anho>/<mes>/<anho2>/<mes2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Euro de período definido por años:

```
http://api.sbif.cl/api-sbif/recursos
  /euro/periodo/<anho>/<anho2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Recursos referentes a la Tasa TAB

Tasa TAB del día actual:

```
http://api.sbif.cl/api-sbif/recursos
  /tab
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de un año:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de un año y mes:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de una fecha específica:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de años posteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/posteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de meses posteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/posteriores/<anho>/<mes>
  ?apikey=<api_key>
```

```
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de meses posteriores a la fecha:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/posteriores/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de años anteriores al año:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/anteriores/<anho>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de meses anteriores al mes del año:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/anteriores/<anho>/<mes>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de meses anteriores a la fecha:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/anteriores/<anho>/<mes>/<dia>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de período definido por fechas:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/periodo/<anho>/<mes>/<dia>/<anho2>/<mes2>/<dia2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de período definido por meses:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/periodo/<anho>/<mes>/<anho2>/<mes2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Tasa TAB de período definido por años:

```
http://api.sbif.cl/api-sbif/recursos
  /tab/periodo/<anho>/<anho2>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Recursos referentes a los balances de instituciones financieras

Lista de instituciones existentes durante el mes del año especificado:

```
http://api.sbif.cl/api-sbif/recursos
  /balances/<anho>/<mes>/instituciones
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Balance de una institución para el año especificado:

```
http://api.sbif.cl/api-sbif/recursos
  /balances/<anho>/instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Balance de una institución para el mes del año especificado:

```
http://api.sbif.cl/api-sbif/recursos
  /balances/<anho>/<mes>/instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Balance de una institución para el mes especificado durante un periodo predeterminado de años (periodo1 = [95-07], periodo2 = [08], periodo3 = [09-hoy]):

```
http://api.sbif.cl/api-sbif/recursos
  /balances/<periodo1|periodo2|periodo3>/<mes>
    /instituciones/<codigoInstitucion>
  ?apikey=<api_key>
```

```
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Lista de cuentas existentes durante el mes del año especificado:

```
http://api.sbif.cl/api-sbif/recursos
  /balances/<anho>/<mes>/cuentas
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Detalle de cuenta durante el mes del año especificado para todas las instituciones:

```
http://api.sbif.cl/api-sbif/recursos
  /balances/<anho>/<mes>/cuentas/<codigoCuenta>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Detalle de cuenta durante el año especificado para todas las instituciones:

```
http://api.sbif.cl/api-sbif/recursos
  /balances/<anho>/cuentas/<codigoCuenta>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Detalle de cuenta durante el año especificado para la institución especificada:

```
http://api.sbif.cl/api-sbif/recursos
  /balances/<anho>/cuentas/<codigoCuenta>
    /instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Detalle de cuenta durante el mes del año especificado para la institución especificada:

```
http://api.sbif.cl/api-sbif/recursos
  /balances/<anho>/<mes>/cuentas/<codigoCuenta>
    /instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Detalle de cuenta durante el mes especificado durante un periodo predeterminado de años (periodo1 = [95-07], periodo2 = [08], periodo3 = [09-hoy]) para la institución especificada:

```
http://api.sbif.cl/api-sbif/recursos
  /balances/<periodo1|periodo2|periodo3>/<mes>/cuentas
    /<codigoCuenta>/instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Recursos referentes a los estado de resultados de instituciones financieras

Lista de instituciones existentes durante el mes del año especificado:

```
http://api.sbif.cl/api-sbif/recursos
  /resultados/<anho>/<mes>/instituciones
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Estado de resultados de una institución para el año especificado:

```
http://api.sbif.cl/api-sbif/recursos
  /resultados/<anho>/instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Estado de resultados de una institución para el mes del año especificado:

```
http://api.sbif.cl/api-sbif/recursos
  /resultados/<anho>/<mes>/instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Estado de resultados de una institución para el mes especificado durante un periodo predeterminado de años (periodo1 = [95-07], periodo2 = [08], periodo3 = [09-hoy]):

```
http://api.sbif.cl/api-sbif/recursos
  /resultados/<periodo1|periodo2|periodo3>/<mes>
    /instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Lista de cuentas existentes durante el mes del año especificado:

```
http://api.sbif.cl/api-sbif/recursos
  /resultados/<anho>/<mes>/cuentas
```

```
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Detalle de cuenta durante el mes del año especificado para todas las instituciones:

```
http://api.sbif.cl/api-sbif/recursos
  /resultados/<anho>/<mes>/cuentas/<codigoCuenta>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Detalle de cuenta durante el año especificado para todas las instituciones:

```
http://api.sbif.cl/api-sbif/recursos
  /resultados/<anho>/cuentas/<codigoCuenta>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Detalle de cuenta durante el año especificado para la institución especificada:

```
http://api.sbif.cl/api-sbif/recursos
  /resultados/<anho>/cuentas/<codigoCuenta>
    /instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Detalle de cuenta durante el mes del año especificado para la institución especificada:

```
http://api.sbif.cl/api-sbif/recursos
  /resultados/<anho>/<mes>/cuentas/<codigoCuenta>
    /instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

Detalle de cuenta durante el mes especificado durante un periodo predeterminado de años (periodo1 = [95-07], periodo2 = [08], periodo3 = [09-hoy]) para la institución especificada:

```
http://api.sbif.cl/api-sbif/recursos/resultados
  /<periodo1|periodo2|periodo3>/<mes>/cuentas/<codigoCuenta>
    /instituciones/<codigoInstitucion>
  ?apikey=<api_key>
  &formato=<json|JSON|xml|XML>
  &callback=<funcion_de_callback>
```

8.4. Participantes de la primera exposición del API de la SBIF

Nombre	*Empresa*	*Área*
Ricardo Arroyo	SBIF	Contenidos
Jorge Barahona	AyerViernes	Diseño
Alejandro Barros	Consultor eGov	Programación
Ds.Jacob Bustamante	U. de Chile - Diseño	Diseño
Héctor Cárcamo	SVS	Programación
Herbert Spencer	U. Católica Valparaíso - Diseño	Diseño / Programación
Rodrigo Duarte	XML Ideas	Diseño / Programación
Tomás Hermosilla J.	MOP	Programación
Claudio Loyola	ChileCompras	Programación / Contenidos
Juan Paulo Madriaza	U. de Chile - Diseño	Diseño
Felipe Mancini	Transparencia SegPres	Contenidos
Paulo Saavedra	SDG	Diseño / Contenidos
Nicolás Silva	Transparencia SegPres	Programación
José Soler	U. Católica - Periodismo	Diseño
Daniel Urbina	SDG	Programación / Contenidos
Darcy Vergara	ChileCompras	Contenidos
Héctor Vergara	Presidencia	Programación
Isabel Villavicencio		Programación

8.5. Lista de testers del API de la SBIF

Nombre	*Empresa*
Hector Vergara	Presidencia
Felipe Mancini	Transparencia SegPres
Alex Yañez	
Tomas Hermosilla	
Juan Camus	SBIF
Ricardo Arroyo	SBIF
Hugo Martinez	SBIF
Manuel Fuenzalida	SBIF
Maria Corvalan	SBIF
Maria Lagos	DesarrolloDigital
Nicolás Riesco	Comisión de Probidad y Transparencia
Basilio Cáceres	Ayer Viernes
Julio Améstica	Amestica
Phillip Neuman	Amestica
	Webtec
Pablo Zúñiga	
Danton Viñales	Punto Ticket S.A.
Boris Cabezas	
Jaime Oyarzun	
Pablo Poo	Inventies
Gabriel Vilaboa M.	
Alvaro Graves	
Felipe Comparini	TVN - Online
Carlos Lillo	PreviRed
Gonzalo Seriche	Ibex