Un proyecto presentado por RedCLARA. Financiado por el Fondo de Bienes Públicos Regional del Banco Interamericano de Desarrollo (BID). Con la participación de Argentina, Brasil, Chile, Colombia, Ecuador, México, Perú y Venezuela

**Proyecto RG-T1684**
**Desarrollo e implementación de las soluciones**
**Prueba piloto del Componente III**
**Informe Número 1.**

# Informe de avance
# Implementación herramientas de back-end (3-III).

**Lautaro Matas**
**11/04/2013**

**Historia de Revisión**

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 11-04-2013 | 1.0 | Primera versión del documento | Lautaro Matas |

# 1 Tabla de contenidos

# 2 Introducción al documento

## 2.1 Propósito

El presente documento tiene como objetivo informar el estado de avance de las tareas de consultoría para el desarrollo e implementación de las soluciones *"Prueba piloto del Componente III - Perfil Desarrollador para las herramientas de back-end (3-III)"* en el marco del proyecto *"Estrategia Regional y Marco de Interoperabilidad y Gestión para una Red Federada Latinoamericana de Repositorios Institucionales de Documentación Científica"*. Las actividades descriptas corresponden a los primeros treinta días de la ejecución de la mencionada consultoría.

## 2.2 Referencias

Como insumo para el desarrollo y las decisiones de implementación fueron considerados los siguiente documentos provistos por el coodinador del componente III, Emiliano Marmonti:

Informe nro. 1: Requerimientos de software
Informe nro. 2: Arquitectura de software

## 2.3 Sobre la metodología de desarrollo

El desarrollo del software responde los lineamientos generales especificados en los documentos de requerimientos y arquitectura proporcionados por el coordinador del componente III. Durante el proceso de desarrollo, en forma diaria, se intercambian consultas, requerimientos y alternativas de soluciones, por medios virtuales, tanto con el coordinador Emiliano Marmonti como con el desarrollador del front-end Antonio Razo.
La metodología aplicada es de desarrollo ágil basada en la liberación temprana de versiones, estableciendo objetivos generales y particulares mediante un sistema de gestión de incidencias y control de código. Esto permite planificar iteraciones que concluyen con versiones operativas del software (milestones) de acuerdo a plazos establecidos por el coodinador. De esa manera, iteración a iteración, se cubren progresivamente los requerimientos funcionales y no funcionales del proyecto mientras se provee de material de prueba a los puntos focales.

## 2.4 Sobre el lenguaje y marco de desarrollo utilizado

De acuerdo al análisis de requerimientos y la arquitectura general diseñada se decidió, en forma conjunta con el coordinador, el uso de lenguaje Java y del marco de trabajo Spring como base para el desarrollo. A continuación se destaca, en forma no exhaustiva, las características consideradas que fundamentan la decisión tomada.

- Java: Independencia de plataforma, permitiendo escenarios de cambio de tecnologías de sistema y arquitectura de software operativo de base.

- Spring-IOC: inversión del control, que permite la configuración de los componentes de aplicación y la administración del ciclo de vida de los objetos Java, se lleva a cabo principalmente a través de la inyección de dependencias. En el marco del proyecto esto permite la actualización y cambio de distintos componentes sin necesidad recompilación general, simplemente agregando librerías y cambiando archivos de configuración.

- Spring-AOP: programación orientada a aspectos que habilita la implementación de rutinas transversales. En el marco del proyecto esto permite la implementación ordenada y clara de los aspectos transversales como la autenticación o el control y registro de errores de las distintas etapas del procesamiento en forma independiente.

- Spring-Data: herramientas de Mapeo objeto relacional con bases de datos (SQL/NOSQL). Esto permite la implementación del modelo de dominio con un alto nivel de abstracción de la tecnología finalmente utilizada para persistirlo, de esta manera pueden soportarse futuros escenarios de cambio en el motor de base de datos.

- Spring-Data: gestión de transacciones: unifica distintas APIs de gestión y coordina las transacciones para los objetos Java, esto permite asegurar la coherencia y ejecutar con seguridad de datos las distintas tareas del módulo back-end.

- Spring-MVC: aplicativos webs según el paradigma modelo/vista/controlador, basado en HTTP y servlets, que provee herramientas para la extensión y personalización de aplicaciones web y servicios web REST. Esto permite al back-end ofrecer una familia de servicios web seguros y consumibles por distintos clientes y que constituyen el principal punto de administración.

- Spring-Security: autenticación y autorización, procesos de seguridad configurables que soportan un rango de estándares, protocolos, herramientas y prácticas tales como LDAP entre otros.

- Spring-JMX: administración remota, configuración de visibilidad y gestión de objetos Java para la configuración local o remota vía JMX. Esto permite ofrecer servicios de monitoreo y control de los procesos en ejecución a cualquier cliente implementando el estándar JMX.

- Spring-Testing: Soporte de clases para desarrollo de unidades de prueba e integración, permitiendo un desarrollo ordenado y verificable en cada iteración.

# 3 Estado de avance de las tareas de implementación

## 3.1 Infraestructura de desarrollo para el equipo

De acuerdo a los términos de referencia se ejecutaron las tareas de puesta en marcha de la infraestructura de desarrollo para el equipo. Dado que ya se contaba con un servidor de desarrollo funcional provisto por la RedCLARA, con Java y Mysql operativos, las tareas se concentraron en la elección y configuración del sistema de control de versiones de código y de gestión de incidencias.

De acuerdo a experiencias propias con diversos productos se recomendó al coordinador la adopción del sistema integral y de uso gratuito GitHub (https://github.com/) que actualmente provee de estos servicios a los proyectos de código abierto más relevantes, tales cómo el kernel Linux, para citar un ejemplo. El coordinador estuvo de acuerdo con el uso de GitHub, en lugar de SVN/Trac, se procedió entonces a la configuración completa de la herramienta, generación de credenciales para los desarrolladores y creación del repositorio principal del proyecto.

El repositorio es público y puede accederse a través de la dirección https://github.com/lareferencia/main/, allí puede seguirse la evolución de código, solución de incidencias y nivel de actividad de los desarrolladores del proyecto.

## 3.2 Implementación de requerimientos

Previo a la implementación se realizó un análisis completo de la arquitectura y en conjunto con el coordinador y el desarrollador del front-end, se discutieron los requerimientos, las distintas alternativas de solución y se estableció un plan de trabajo que luego fue reflejado en el gestor de incidencias.

Durante los primeros treinta días de la consultoría se fijó como objetivo la implementación de una primera versión del back-end. denominada “Milestone1”, se centró en el cumplimiento de los siguientes requerimientos: permitir la definición de redes nacionales y sus orígenes de metadatos, cosecha concurrente multihilo de los metadatos expuestos por la redes mediante el protocolo OAI-PMH, validación de los metadatos de acuerdo a las directivas driver, la transformación básica de metadatos (en los casos que resulte posible) para cumplir con las directivas, diagnóstico de los errores más comunes, almacenamiento de la información y cálculo de estadísticas básicas sobre los metadatos cosechados.

El objetivo general fue el de proveer, desde la primera versión, una herramienta capaz de facilitar las tareas de diagnóstico de situación de las redes nacionales, conforme a lo definido por la coordinación del proyecto. Esta primera versión se implementó en forma exitosa y se haya disponible en el repositorio de código del proyecto.

### 3.2.1 Modelado e implementación del modelo de dominio

Se definieron objetos persistentes que modelan el dominio de datos de las redes nacionales, sus datos básicos, orígenes de cosecha. También se modeló el concepto de instantánea (snapshot) de una red en un momento determinado y las información de los registros completa de los metadatos asociados. Los objetos se corresponden con el modelo trazado en la arquitectura y están relacionados de acuerdo a lo allí consignado. Los objetos del modelo de dominio se hayan estructurados en el paquete org.lareferencia.backend.domain y puede consultarse su última versión en: https://github.com/lareferencia/main/tree/master/backend/src/main/java/org/lareferencia/backend/domain

La persistencia es manejada por Spring Data JPA, utilizando Hibernate como motor de persistencia de objetos y MYSQL como motor de base de datos. Estos parámetros pueden ser modificados sin mayor impacto en el resto del desarrollo.

El alta, baja y modificación de objetos es provista en forma automática por el marco de trabajo Spring y las interfaces de acceso se hayan implementadas en el paquete org.lareferencia.backend.repositories y puede consultarse su última versión en: https://github.com/lareferencia/main/tree/master/backend/src/main/java/org/lareferencia/backend/repositories

### 3.2.2 Implementación de los requerimientos de cosecha de metadatos

Se implementó un esquema de procesamiento concurrente mutihilo y programable de acuerdo a los lineamientos establecidos en el documento de arquitectura. Se cuenta con un objeto (SnapshotManager) responsable de el lanzamiento de procesos (Workers) encargados de la cosecha de los metadatos de cada red nacional. El lanzamiento puede darse en forma directa o programarse (Scheduler) mediante expresiones cronológicas persistidas. Cada proceso (Worker) se encarga en forma independiente de la cosecha de una red mediante la creación de objeto IHarvester que se comunica en forma asincrónica mediante la recepción de eventos. De esa manera recibe y almacena los metadatos en forma progresiva, informando los errores ocurridos en el proceso. Actualmente el cosechador se haya implementado sobre la versión original del OCLC Harvester2, a través de una capa de abstracción que permite manejar reintentos con intervalos de tiempo creciente, centralizar errores y potencialmente permitiría el cambio de la implementación del cosechador sin impacto en el resto del código.

El esquema de procesamiento se haya implementado en el paquete org.lareferencia.backend.tasks y puede consultarse su última versión en: https://github.com/lareferencia/main/tree/master/backend/src/main/java/org/lareferencia/backend/tasks

Las interfaces de cosechador y la capa de abstracción sobre el OCLC Harverster2 se haya implementada en el paquete org.lareferencia.backend.harvester y puede consultarse su última versión en:

https://github.com/lareferencia/main/tree/master/backend/src/main/java/org/lareferencia/backend/harvester

### 3.2.3 Implementación de validación y transformación de metadatos

Se implementó un validador estructurado en forma jerárquica, de manera que la validez de un registro depende directamente validez de las ocurrencias de los campos que contiene y su obligatoriedad de acuerdo a las directivas DRIVER. Todo el esquema de validación puede ser configurado sin necesidad de recompilación del código a través de un archivo xml, en futuras iteraciones se planifica almacenar la configuración en el modelo de domino, permitiendo que el administrador puede revisarlo y actualizarlo en forma periódica de acuerdo a los posibles cambios en las directivas.

La validación de ocurrencias es definida utilizando reglas de contenido, en la primera versión se han implementado validaciones por listado de valores controlados, expresiones regulares y longitud de cadenas; de momento cubre todas las necesidades de validación detectadas pero no se descarta la implementación de otras modalidades.

El proceso de validación genera además información detallada que permite conocer en forma exacta el resultado de la aplicación de las reglas a cada ocurrencia de cada campo y su peso en la validez del registro. Esto permite alimentar el módulo de estadísticas y contar con reportes detallados de estado de situación de una red en términos del cumplimiento de las directivas DRIVER.

El proceso de validación se encuentra implementado en el módulo org.lareferencia.backend.validator y puede consultarse su última versión en:
https://github.com/lareferencia/main/tree/master/backend/src/main/java/org/lareferencia/backend/validator

En esta primera versión se a implementado un proceso de trasformación básico y estático, en base a la información y experiencia transmitida por el coordinador del componente III.

La actual versión permite la traducción de los errores más comunes y se planifica implementar un transformador configurable en próximas iteraciones.
El proceso de transformación se encuentra implementado en el módulo org.lareferencia.backend.transformer y puede consultarse su última versión en:
https://github.com/lareferencia/main/tree/master/backend/src/main/java/org/lareferencia/backend/transformer

El proceso de validación, durante esta primera etapa es ejecutado sobre los registros ya cosechados, sin embargo se planifica que en futuras iteraciones esté incorporado al módelo de procesamiento concurrente junto con las demás tareas. Esto permitirá validar, y en caso de ser necesario transformar, los metadatos en la medida que son cosechados, en forma concurrente, aprovechando mejor los recursos disponibles.

### 3.2.4 Módulo de estadísticas

De acuerdo a las necesidades de diagnóstico definidas por el coordinador del componente se implementó un módulo básico que permite el cálculo de distintos conteos basados en los resultados de la validación y permiten delinear la evolución del estado de las redes en las sucesivas cosechas.

Este módulo irá evolucionando en las futuras iteraciones y el objetivo final es implementar un conjunto de herramientas que permitan calcular y almacenar estadísticas brindando un servicio web REST/JSON que pueda ser consumido por el propio back-end y otros clientes para la generación de reportes numéricos y gráficos que den cuenta de la historia y calidad de los metadatos de las redes cosechadas.

La funcionalidad básica de estadísticas se encuentra implementado en el módulo org.lareferencia.backend.stats y puede consultarse su última versión en:
https://github.com/lareferencia/main/tree/master/backend/src/main/java/org/lareferencia/backend/stats

### 3.2.5 Interfaces de administración del back-end

En esta primera versión se priorizó, con acuerdo del coordinador del componente, la implementación de los servicios de cosecha, validación, transformación y estadísticas, dejando para futuras implementaciones los aspectos de interfaz de alta, baja y modificación de datos y administración de procesos.

No obstante, el tema fue analizado y se planifica implementar una capa de servicios web REST/JSON (seguros) que brinden acceso a todas las tareas de administración de administración de redes nacionales y administración general de la plataforma. Esto se hará a través del Spring-REST provisto por el marco de trabajo Spring. Estos servicios serán consumidos por una aplicación web HTML5 que permitirá el despliegue de los datos consumidos de los servicios web, la actualización de datos de la plataforma y la graficación de estadísticas. Esta separación en capas permite brindar servicios a otras aplicaciones (incluido el front-end) de forma centralizada y transparente.
Se contempla también la implementación de servicios JMX, a través de Spring-JMX para el monitoreo y seguimiento de los procesos involucrados.

### 3.2.6 Protocolo de autenticación

En esta etapa del proyecto y ante la necesidad de concentrar el esfuerzo de desarrollo en otras funcionalidades, no se han implementado protocolos de autenticación. Sin embargo se han analizado y considerado este aspecto a la hora de la elección del marco de trabajo. A través de Spring-AOP y Spring-Security pueden implementarse sin mayor impacto en el resto del código diversos esquemas de autenticación locales o remotos tales como LDAP, entre otros.

# 4 Resultados de pruebas del software implementado (milestone1)

En esta sección se presentan algunos resultados de las pruebas del software desarrollado sobre las redes nacionales actualmente disponibles. No es objetivo de este informe presentar un diagnóstico de la situación de las redes sino probar la implementación actual en un contexto de uso real, por esa razón se omiten detalles de datos que sí fueron brindados como insumo al coordinador del componente III.

El proceso de cosecha concurrente fue lanzando sobre ocho repositorios de redes nacionales provistos por el coordinador. Dependiendo del tamaño y velocidad de respuesta de cada red las cosechas en fueron concluyendo forma progresiva, cuando finalizaron las ocho cosechas se contaba con un conjunto de **405480** registros de metadatos almacenados.

El proceso de validación y transformación fue lanzado sobre la totalidad de registros de las ocho redes cosechadas, generando como resultado tablas de datos de estadísticas y reportando un total de **153034 registros válidos** según las directivas driver especificadas en el validador.

El consumo máximo memoria durante el proceso de cosecha, de las ocho redes en forma concurrente fue de 350 megabytes de RAM física, este consumo es independiente del tamaño de las redes, pues la aplicación está diseñada manteniendo una cantidad de registros acotada (persistiéndolos cuando ya no son usados), asegurando la **escalabilidad del sistema** considerando el crecimiento futuro de las redes en cantidad de registros. Cabe aclarar que el consumo de memoria se incrementará levemente al incorporar los procesos de validación pero se estima que el impacto será poco significativo. Por otra parte se cuenta con la posibilidad de configurar la cantidad de hilos máxima que se puede admitir y de forma transparente, a través de colas de espera puede regularse la cantidad de cosechas concurrentes, pudiendo acotar el consumo de recurso en función de las capacidades de hardware disponible.

# 5 Conclusiones

A treinta días de comenzado el trabajo se cuenta con una primera versión funcional del software, que puede ser probada y mejorada en futuras iteraciones, pero que desde un comienzo opera sobre el escenario real de uso generando datos que resultan útiles a otras componentes del proyecto general.

El proceso de desarrollo ágil iterativo, dirigido por el reporte y solución de incidencias ha resultado una metodología exitosa para el cumplimiento de los objetivos planteados para la primera etapa de la consultoría.

El marco de trabajo elegido, Spring, ha resultado una base flexible y confiable para la implementación tanto de requerimientos ya documentados como detectados posteriormente, cumpliendo en gran medida con los lineamientos establecidos por la arquitectura recibida como insumo.

11 de Abril de 2013.

Lautaro Matas
Consultor Desarrollador Back-end Componente III