Universidad de las Ciencias Informáticas

Facultad No I

Propuesta arquitectónica para el desarrollo del Gestor de Documentos Administrativos eXcriba 2.0

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Michel David Suárez

Tutores: Ing. Misael Fonseca Mata

Ing. Annia Surós Vicente

Co-Tutor: Ing. Marcel Sánchez Góngora

Ciudad de la Habana, Julio de 2011

Declaración de autoría

Declaro ser el autor de la presente tesis y reconozco a la UCI los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Michel David Suárez

_____	_____	_____
Ing.	Ing.	Ing.
Misael Fonseca Mata	Annia Surós Vicente	Marcel Sánchez Góngora

Resumen

El presente trabajo es una propuesta de arquitectura de software sobre la cual desarrollar la versión 2.0 del gestor de documentos administrativos(GDA) eXcriba. Durante el desarrollo del mismo se realiza un análisis de los elementos fundamentales que constituyen las arquitecturas de software con el objetivo de lograr mejorar cuatro aspectos fundamentales dentro del entorno donde se desarrolla la solución eXcriba, productividad, rendimiento, mantenibilidad y calidad.

El éxito de eXcriba 2.0 tiene mucha implicación por las decisiones arquitectónicas que han sido tomadas durante su desarrollo, la definición de tecnologías empleadas, los aspectos esenciales del diseño y la suministración a los miembros del equipo de desarrollo de una idea bien clara de lo que se está desarrollando.

En este documento se plasman los resultados del estudio realizado acerca de la arquitectura de software y sus principales corrientes. Se realiza una investigación sobre el estado de los diferentes tipos de patrones que intervienen en la materialización de una arquitectura para un sistema de software, así como la descripción y construcción de la solución propuesta. Además se evalúa la propuesta a partir de la presentación de un prototipo ejecutable, un método basado en demostración y un conjunto de criterios claves que permiten dar los elementos, suficientes para ponerla en marcha.

Índice General

Introducción

Las dos últimas décadas del siglo XX se caracterizaron por un gran movimiento de la información que se volvió incluso más rápido que el movimiento físico. El sociólogo Manuel Castells denominó al período como la *Era de la Información.*[2] La obra de igual nombre al término que el autor definiría entre los años 1996 y 1998, hace un marcado énfasis al proceso de transformación tecnológico que por aquel entonces comienza a experimentarse, como se evidencia en la cita *[...]producto a la habilidad para crear una interfaz entre los campos tecnológicos a través de un lenguaje digital común en el que la información es generada, almacenada, recuperada, procesada y retransmitida.*[2]

El análisis de Castells se desarrolla a lo largo de tres dimensiones básicas producción, poder y experiencia. Sustenta a priori cómo han de ser entendidas en sus propios términos, así como en relación con las demás. Su análisis aplicado sobre el desarrollo de Internet, enfatiza los papeles del Estado (en lo militar y académico), movimientos sociales (piratas informáticos y activistas sociales) y empresas en el moldeado de la infraestructura en relación con sus conflictivas agendas[2].

Aún cuando el término proviene desde mucho antes que la conocida *Era de la Información*, puede decirse que la *Sociedad de la Información*[1] resurge con nuevos bríos con el devenir de los nuevos procesos de transformaciones tecnológicas.[3] Sin embargo no es hasta que la adopción de formas, métodos, maneras de abordar y resolver problemas, que puede hablarse de conocimiento en lugar de información. Con el conocimiento llega también la *Sociedad del Conocimiento*,[4] donde la información y el conocimiento tienen un lugar en la sociedad y la cultura.

La *Sociedad del Conocimiento* implica el surgimiento de una nueva etapa caracterizada por usar el conocimiento como elemento fundamental para generar valor y riqueza por medio de su transformación

[1]Según Mattlelard el término responde a la sociedad en la cual las tecnologías que facilitan la creación, distribución y manipulación de la información juegan un papel importante en las actividades sociales, culturales y económicas.

a información. Este nuevo período en el que la sociedad está inmersa es llamado *Economía Basada en Conocimiento*.[5] La *Economía del Conocimiento* no genera valor y riqueza por medio de su transformación en información; sino que crea valor añadido en los productos y servicios en cuyo proceso de creación o transformación participa.[5] Paralelamente, el crecimiento de la información que los medios tecnológicos manejan, dificulta paulatinamente la veracidad, prueba y testimonio de los recursos que la contienen.

Ante esta dificultad aparecieron una serie de soluciones que de una forma u otra, intentan resolver el problema; tal es el caso de los Sistemas de Gestión Integral de Documentos (SGID), alternativa superior a los sistemas de gestión de documentos tradicionales[2] y que en uno de sus componentes involucran un gestor de documentos administrativos (GDA). El desarrollo y comercialización de un sistema GDA, cumple el perfil de explotación de la *Economía Basada en Conocimiento* dado que suelen brindarse un conjunto de servicios aparejados al producto. Como es de evidenciarse en la medida que los desarrolladores de soluciones de este tipo, sean capaces de agilizar el o los procesos involucrados en el desarrollo y prestación de los servicios agregados al producto, más valor será generado.

Una de las aristas para incrementar la capacidad de proveer servicios y desarrollar el producto GDA recae directamente en la infraestructura de desarrollo que se tenga creada.

La propia necesidad de ofrecer cada vez más servicios y mejorar el producto ha hecho incluso replantear la posibilidad extender las funciones del GDA eXcriba con lógicas muy particulares de los negocios en los que se haya implantado, así como la de desarrollar otras aplicaciones con otros requerimientos que se extienden más allá de los básicos para un GDA[3]. Esto trae como consecuencia afectaciones que influyen directamente sobre los cuatros criterios fundamentales que sustentan la presente propuesta y que de una forma u otra afectan la capacidad de la infraestructura en ofrecer esos productos y servicios. Estas afectaciones hacen afirmar que el GDA eXcriba es incapaz adaptarse a la incorporación de nuevas funcionalidades que se identifican en los procesos de la gestión documental, pues el diseño

[2]La gestión documental tradicional, intenta resolver el mismo problema excluyendo las herramientas informáticas.

[3]Estos requerimientos están descritos en la especificación de MoReq. Ver apunte bibliográfico: *Proyecto UNE-ISO 15489*.

arquitectónico actual no garantiza la autonomía de los módulos, dificulta su mantenimiento, además, involucra más coste de desarrollo y hace que el sistema tenga un bajo rendimiento. Es por ello que se ha planteado el siguiente **problema de investigación:** ¿Cómo lograr que la arquitectura del GDA eXcriba garantice la independencia entre sus módulos, que el software sea mantenible y que disminuya el rendimiento del sistema?

El **objeto de estudio** en el que está centrado la investigación es: Las arquitecturas de software. Para dar cumplimiento al objetivo general se han establecido los siguientes **objetivos específicos**:

- Analizar los estilos arquitectónicos que puedan usarse para desarrollar la propuesta.
- Formular la propuesta de arquitectura a partir de un prototipo funcional viable.

Para dar cumplimiento a los objetivos se deben desarrollar las siguientes tareas de investigación:

- Realización de un estudio referente a los siguientes temas:
    1. Patrones de arquitectura de software.
    2. Patrones de diseño de software.
    3. Diferentes tecnologías, marcos de trabajo, funcionalidades y herramientas.
    4. Técnicas de construcción de proyectos.
- Confección del diseño del prototipo ejecutable de la arquitectura.
- Comparación los resultados esperados con los resultados obtenidos.

Ésta investigación tiene como idea a defender: El rediseño de la arquitectura del GDA eXcriba facilita a la infraestructura el incremento de la capacidad de prestación de productos y servicios.

Para el cumplimiento de las tareas antes plateadas se utilizaron los siguientes métodos:

Analítico-Sintético, el cual facilitó identificar, analizar y seleccionar los conceptos y definiciones más importantes relacionados con la presente investigación.

Análisis Histórico-Lógico, el cual permitió determinar la evolución y desarrollo hasta la actualidad de las arquitecturas de software para de esta forma determinar las tendencias actuales del software.

El trabajo se estructura como se muestra a continuación:

- Capítulo I: Las Arquitecturas de Software, su papel en la prestación de productos y servicios.

  Aborda conceptos relacionados con las arquitecturas de software, el acontecer nacional e internacional relacionado con el tema, además de como influye la arquitectura en la capacidad de generación de productos y servicios.

- Capítulo II: Propuesta de diseño arquitectónico para el GDA eXcriba.

  Expone la intención de la propuesta, definiendo los elementos técnicos de la propuesta. Define los patrones, el diseño de clases, los medios empleados, el modelo de despliegue de la arquitectura, las restricciones de diseño, los aspectos de mejoras, entre otros elementos.

- Capítulo III: Evaluación y aceptación de la arquitectura del GDA eXcriba.

  Evaluación basada en demostración que explica cómo la arquitectura debe dar garantías de que la solución diseñada es realizable de acuerdo a los atributos de calidad.

Capítulo 1

Las Arquitecturas de Software, su papel en la prestación de productos y servicios

Las estrategias comerciales de las empresas consisten en llevar a cabo un conjunto de acciones para lograr un determinado objetivo relacionado con la mercadotecnia, algunos de ellos son capturar un mayor número de clientes, incentivar las ventas, dar a conocer nuevos productos, lograr una mayor cobertura o exposición de los productos. Philip Kootler en el 2001 definió el producto como cualquier cosa que se puede ofrecer a un mercado para satisfacer un deseo o una necesidad.[6] Este interesante punto de vista, hace comprender muchas de las estrategias para el producto que las empresas utilizan para lograr posicionarse en el mercado, entre las que se destacan:

- Incluir nuevas características al producto: Darle nuevas mejoras, nuevas utilidades, nuevas funciones, nuevos usos.
- Incluir nuevos atributos al producto: Darle un nuevo empaque, un nuevo diseño, nuevos colores, nuevo logo.
- Lanzar una nueva línea de producto: Si el producto es zapatos de mujer, lanzar otra línea como pantalones de mujer.
- Ampliar la línea de producto: Aumentar el menú del restaurante, o sacar un nuevo tipo de champú para otro tipo de cabello.

- Lanzar una nueva marca (sin necesidad de sacar del mercado la que ya existe): Una nueva marca dedicada a otro tipo de mercado, quizás uno de mayor poder adquisitivo.
- Incluir nuevos servicios al cliente, que les brinden mayor disfrute del producto: Incluir la entrega a domicilio, el servicio de instalación, nuevas garantías, nuevas facilidades de pago, una mayor asesoría en la compra.

Un poco más allá se encuentran las ventas de servicios y productos como software y hardware pues estos requieren de una metodología especial para su comercialización, por ser de alta tecnología y en la mayoría de los casos de tipo intangible.

ALBET Ingeniería y Sistemas, es una empresa cubana, cuyo origen y desarrollo se vincula estrechamente a la Universidad de Ciencias Informáticas (UCI), modelo de universidad productiva que agrupa más de doce mil profesionales, técnicos y estudiantes.[7] ALBET posee los derechos comerciales de todos los productos y servicios que desarrolla la UCI y mediante la alianza con otras prestigiosas entidades ofrece soluciones integrales en la esfera de las tecnologías de la información y las comunicaciones. La citada empresa cubana y por ende la UCI, no escapan de estos nuevos modelos de comercialización.

1.1. Crecimiento de la Información

Una investigación de IDC patrocinada por EMC, mide y prevé la cantidad y el tipo de información digital que se crea y se copia en el mundo, y si la generan particulares o empresas.

El estudio, denominado *The Expanding Digital Universe: A Forecast of Worldwide Information Growth Through 2010*[1], reveló la cantidad de información que el mundo crea y copia en un determinado año. Además, realizó predicciones sobre todo este "universo digital" hasta el año 2010, e identificó las geografías y los tipos específicos de información que contribuyen a su crecimiento. Los resultados del informe tienen diversas consecuencias para los particulares, las empresas y la sociedad.

Principales conclusiones:

[1]En español: *El universo digital en expansión: un pronóstico del crecimiento de la información mundial hasta 2010.*

- En 2006, el universo digital tenía un tamaño de 161.000 millones de gigabytes (161 exabytes).
- IDC proyectó una sextuplicación anual de la información de 2006 a 2010.
- Mientras que casi 70 % del universo digital fue generado por particulares para 2010, las organizaciones fueron responsables de la seguridad, la privacidad, la confiabilidad y el cumplimiento con las normas de al menos 85 % de la información.[8]

En 2006, se crearon y copiaron 161 exabytes de información digital, continuando un período sin precedentes de aumento de la información. Este universo digital equivale a unas tres millones de veces la información almacenada en todos los libros jamás escritos, o a 12 pilas de libros, cada de una de las cuales se extiende por más de 150 millones de kilómetros, la distancia que hay de la tierra al sol. Según IDC, la cantidad de información creada y copiada en 2010 aumentó más de seis veces a 988 exabytes, una tasa de crecimiento anual compuesta de 57 %.

La mayoría de la información digitalizada que las organizaciones necesitan proteger es generada fuera de los centros de datos, y a menudo fuera de las propias compañías por usuarios cada vez más móviles –trabajadores, clientes, proveedores, socios–, lo que añade más complejidad a la hora de gestionar y asegurar la información.

En una actualización del informe de IDC se dice que la cantidad de información que se considera sujeta a normativas gubernamentales y que debe ser almacenada y accesible para las autoridades reguladoras y auditores, supondrá el 35 % del Universo Digital en 2012, mientras que en 2008 representaba el 25 %. Entre estos datos figuran información de identificación personal, archivos de correo electrónico de empleados, registros de recursos humanos y financieros y documentos legales.[9]

La crisis financiera provocó una mayor intervención reguladora de los gobiernos, lo que generó más información digitalizada, ha hecho que las administraciones públicas son cada vez más protagonistas de la necesidad de auditar la información, de controlar la veracidad, las pruebas y los testimonios de cualquier actividad o hecho ocurrido.[9] En la tesis doctoral de la prestigiosa, en aquel entonces Máster y ahora Doctora Mayra M. Mena Mujica, en la segunda conclusión parcial del capítulo dos, la doctora afirma que:

La introducción de herramientas gerenciales de organización y el impacto de las TICs en las administraciones públicas provocaron la modificación de sus modelos comunicacionales y consecuentemente la forma de los documentos en que estos se asentaban. La aparición de fenómenos como el "espacio virtual de trabajo compartido", "la instantaneidad" y las formas documentos interactivas, experienciales y dinámicas –muy diferentes de los documentos escritos, fijos y textuales– minaron el modelo de organización burocrática. Esto ha conducido a una "crisis de transparencia y responsabilidad" en las administraciones públicas, en la que los sistemas de gestión de documentos electrónicos, estructurados sobre la base de requisitos funcionales archivísticos, son esenciales para que las administraciones puedan superar esta crisis. Estos sistemas son, por tanto, una herramienta esencial para identificar, probar, disminuir o eliminar la corrupción en las organizaciones[10].

1.2. Gestión Documental

La norma UNE-ISO 15489-I ha definido la gestión documental como área de gestión responsable de un control eficaz y sistemático de la creación, la recepción, el mantenimiento, el uso y la disposición de documentos de archivo, incluidos los procesos para incorporar y mantener en forma de documentos la información y prueba de las actividades y operaciones de la organización[11]. De cierta manera esta es el área que puede lograr de en alguna forma controlar en ambiente entrópico de la información en las empresas y sobre todo en aquellas que pertencen a la administración pública.

UNE-ISO 15489-I quien fue creada con el propósito de regular la gestión de documentos de las organizaciones que los han producido, ya sean públicas o privadas, para clientes externos e internos, dice que la gestión documental tiene un objetivo concreto: facilitar el uso cotidiano de documentos en el desarrollo de las actividades de la organización.

En un estudio realizado en Febrero de 2009 Documents and Records Management: An Analysis of Market Trends to 2013 [2][12], se exponen algunas interesantes ideas, una de ellas plantea que los sectores con mayor índice de soluciones para la gestión de documentos y archivos, como los centros

[2]En español *Gestión de Documentos y Archivos: Un análisis de las tendencias del mercado para el 2013.*

hospitalarios, bancos, sectores de la administración pública y de seguro, son además los que dirigen las mayores inversiones a lo largo de las principales áreas de la gestión de documentos y archivos. Por otro lado el análisis revela que una gran proporción de las instituciones y organizaciones que circulan alrededor de estos sectores, están planeando pequeñas y medianas inversiones en soluciones de gestión de documentos y archivos. En el sector de la salud, el 63 % de las instituciones están planeando la inversión de sistemas de gestión documental y archivos en los próximos dos años; a más corto plazo una de dos instituciones están analizando la posibilidad de invertir en sistemas de gestión de activos digitales.

ALBET, la UCI y los demás asociados a la producción de software para empresas no escapan de las tendencias a desarrollar soluciones de este tipo, es por ello que existe un proyecto para desarrollar el GDA eXcriba en la universidad con el propósito de ser el sistema que se replique a través de un conjunto de servicios, donde se personalize, y además agregue otros servicios como las consultorías en procesos documentales.

1.3. Definiciones formales de la Arquitectura de Software

1.3.1. Orígenes

Sólo el documento digital tiene esta sección.

1.3.2. Corrientes Arquitectónicas

- **Arquitectura estructural, basada en un modelo estático de estilos, ADLs y vistas.**

  Constituye la corriente fundacional y clásica de la disciplina. Los representantes de esta corriente son todos académicos, mayormente de la Universidad Carnegie Mellon en Pittsburgh: Mary Shaw, Paul Clements, David Garlan, Robert Allen, Gregory Abowd, John Ockerbloom. En toda la corriente, el diseño arquitectónico no sólo es el de más alto nivel de abstracción, sino que además

no tiene por qué coincidir con la configuración explícita de las aplicaciones; rara vez se encontrarán referencias a lenguajes de programación o piezas de código.

- **Arquitectura como una etapa de ingeniería y diseño orientada a objetos**.

  Esta corriente es una alternativa de la descrita anteriormente. Es el modelo de James Rumbaugh, Ivar Jacobson, Grady Booch, Craig Larman y otros, ligado estrechamente al mundo de UML y RUP. En esta postura, la arquitectura se restringe a las fases iniciales y preliminares del proceso y concierne a los niveles más elevados de abstracción. Importa más la abundancia y el detalle de diagramas y técnicas disponibles que la simplicidad de la visión de conjunto. La definición de arquitectura que se promueve en esta corriente tiene que ver con aspectos formales a la hora del desarrollo.

  Las definiciones revelan que la AS, en esta perspectiva, concierne a decisiones sobre organización, selección de elementos estructurales, comportamiento, composición y estilo arquitectónico susceptibles de ser descritas a través de las cinco vistas clásicas del modelo 4+1 de Kruchten.

- **Arquitectura basada en patrones**

  Esta corriente se basa principalmente en la redefinición de los estilos como patrones POSA, el diseño consiste en identificar y articular patrones preexistentes, que se definen en forma parecida a los estilos de arquitectura.

- **Arquitectura procesual y metodologías**

  Esta surge desde comienzos del siglo XXI, con centro en el SEI y con participación de algunos arquitectos de Carnegie Mellon de la primera generación y muchos nombres nuevos de la segunda: Rick Kazman, Len Bass, Paul Clements, Felix Bachmann, Fabio Peruzzi, Jeromy Carrière, Mario Barbacci y Charles Weinstock, intenta establecer modelos de ciclo de vida y técnicas de diseño, análisis, selección de alternativas, validación, comparación, estimación de calidad y justificación económica específicas para la arquitectura de software.

1.3.3. Estilos Arquitectónicos

Los "estilos arquitectónicos" de software son marcos de referencias arquitectónicas, formas comunes o clases de sistemas.

Los estilos de arquitectura se definen como las 4C:

- Componentes
- Conectores
- Configuraciones
- Restricciones

A la hora de definir un estilo arquitectónico es necesario tener en cuenta el tipo de aplicación ya que puede imponer restricciones que acotan la interacción de los componentes, además se tiene en cuenta el patrón de organización general.

Algunos de los principales estilos arquitectónicos que se usan en la actualidad están divididos por Clases de Estilos las que engloban una serie de estilos arquitectónicos específicos:

- **Estilos de flujo de datos**

  Esta familia de estilos enfatiza la reutilización y la modificabilidad. Es apropiada para sistemas que implementan transformaciones de datos en pasos sucesivos.

  - Tuberías y Filtros (estilo arquitectónico específico): Una tubería es una popular arquitectura que conecta componentes computacionales a través de conectores, de modo que el procesamiento de datos se ejecuta como un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. Este estilo se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada.

- **Estilos centrados en datos**

  Esta familia de estilos enfatiza la integrabilidad de los datos. Se estima apropiada para sistemas que se fundan en acceso y actualización de datos en estructuras de almacenamiento.

  - Arquitecturas de Pizarra o repositorio: Esta arquitectura está compuesta por dos componentes principales: una estructura de datos que representa el estado actual y una colección de componentes independientes que operan sobre él. Estos sistemas se han usado en aplicaciones que requieren complejas interpretaciones de proceso de señales (reconocimiento de patrones, reconocimiento de habla).

- **Estilos de Llamada y Retorno**

  Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala.

  - Arquitectura en Capas: En este estilo arquitectónico cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior, al dividir un sistema en capas, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás. La división de un sistema en capas facilita el diseño modular, en la que cada capa encapsula un aspecto concreto del sistema y permite además la construcción de sistemas débilmente acoplados, lo que significa que si se minimiza las dependencias entre capas, resulta más fácil sustituir la implementación de una capa sin afectar al resto del sistema.
  - Arquitectura Orientada a Objetos: Los componentes de este estilo son los objetos, o más bien instancias de los tipos de dato abstractos. En la caracterización clásica de David Garlan y Mary Shaw,[13] los objetos representan una clase de componentes que ellos llaman managers (En español controladores), debido a que son responsables de preservar la integridad de su propia representación. Un rasgo importante de este aspecto es que la representación interna de un objeto no es accesible desde otros objetos.

- Arquitectura basada en Componentes: Los sistemas de software basados en componentes se basan en principios definidos por una ingeniería de software específica. Los componentes son las unidades de modelado, diseño e implementación, las interfaces están separadas de las implementaciones, y conjuntamente con sus interacciones son el centro de incumbencias en el diseño arquitectónico. Los componentes soportan algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución.

- **Estilo de Código Móvil**

  Esta familia de estilos enfatiza la portabilidad. Ejemplos de la misma son los intérpretes, los sistemas basados en reglas y los procesadores de lenguaje de comando.

  - Arquitectura de Máquinas Virtuales: Esta arquitectura se conoce como intérpretes basados en tablas o sistemas basados en reglas. Estos sistemas se representan mediante un seudo-programa a interpretar y una máquina de interpretación. Estas variedades incluyen un extenso espectro que está comprendido desde los llamados lenguajes de alto nivel hasta los paradigmas declarativos no secuenciales de programación.

- **Estilos Peer To Peer (En español, Punto a Punto)**

  Esta familia se conoce también como componentes independientes, enfatiza la modificabilidad por medio de la separación de las diversas partes que intervienen en la computación. Consiste por lo general en procesos independientes o entidades que se comunican a través de mensajes.

  - Arquitecturas Basadas en Eventos: Estas se han llamado también arquitectura de invocación implícita, se vinculan con sistemas basados en publicación-suscripción. La idea dominante en la invocación implícita es que, en lugar de invocar un procedimiento en forma directa un componente puede anunciar mediante difusión uno o más eventos.
  - Arquitecturas Orientadas a Servicios (SOA): Esta construye toda la topología de la aplicación como una topología de interfaces, implementaciones y llamados a interfaces; es una

relación entre servicios y consumidores de servicios, ambos lo suficientemente amplios como para representar una función de negocio completa.

- Arquitecturas Basadas en Recursos: Esta define recursos identificables y métodos para acceder y manipular el estado de esos recursos. El caso de referencia es nada menos que la World Wide Web, donde los URIs identifican los recursos y HTTP es el protocolo de acceso. El argumento central es que HTTP mismo, con su conjunto mínimo de métodos y su semántica simplísima, es suficientemente general para modelar cualquier dominio de aplicación. Conocido como Representational State Transfer (REST, por sus siglas en inglés), este estilo arquitectónico fue originalmente introducido para construir largos y escalables sistemas de hypermedia distribuidos. Los cuatro principios que lo distinguen son:

  1. Identificación del recurso a través de una URI.[14]
  2. Interfaz uniforme: Los recursos son manipulados usando un conjunto fijo de operaciones: PUT, GET, POST, DELETE.
  3. Mensajes de auto descripción: Los recursos están desacoplados de su representación por lo que sus contenidos pueden ser accedidos en múltiples variedades de formatos, es por ello que es requerido descripciones del recurso, a manera de meta-datos, que permitan para implementar detección de errores, negociación adecuada de la correcta representación, realizar control de acceso en los recursos entre otros.
  4. Estado completo de las interacciones a través de los enlaces(del inglés hyperlinks): Cada interacción con un recurso tiene un estado desconectado, sin embargo para lograr el mantenimiento de este, pueden usarse técnicas de reescrituras de URLs, cookies, y manejo de campos ocultos dentro del cuerpo de la respuesta.

  Los servicios REST se perciben como simples porque delega la implementación de su mecanismo a través de estándares bien conocidos de la W3C/IETF como HTTP, URI y MIME. Los clientes y servidores HTTP están disponibles para casi todos los lenguajes de programación y sistemas operativos, es por ello que REST es una infraestructura ligera donde los

servicios pueden ser construidos con el mínimo de herramientas. El esfuerzo requerido para construir un cliente de servicios REST es muy pequeño, pues los programadores pueden comenzar a probar sus servicios desde un navegador ordinario, sin tener que desarrollar un software personalizado para acceder a dichos servicios. Desplegar una aplicación completamente basada en servicios REST (conocido en inglés como RESTful), es muy similar a construir un sitio web dinámico.[15]

1.3.4. Relación entre estilos y patrones

Una vez que se ha decidido que estilos se van a usar en la aplicación los patrones que tienen relación con estos estilos ayudaran a derivar de esa formulación arquitectónica el diseño y ulteriormente la programación correspondiente.

¿Qué son los Patrones?

Los patrones fueron sugeridos por Christopher Alexander en 1977[16], el cual escribió una serie de libros y artículos referentes a ¿qué son los patrones?, los textos frecuentemente encontrados se refieren a patrones arquitectónicos en cuanto a arquitectura real, o sea, de edificios y ciudades y no de arquitectura de aplicaciones de software.

Una de las ideas brillantes de la década de los 90 fue vincular las estructuras recurrentes, aquellas entidades que ocurrían una y otra vez; a problemas en determinados contextos con esta idea de Alexander que estaba referida como se dijo anteriormente a la arquitectura real.

- Un patrón es la solución a un problema en un contexto.
- Un patrón codifica conocimiento específico acumulado por la experiencia en un dominio.
- Un sistema bien estructurado está lleno de patrones.
- Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y luego describe

el núcleo de la solución a este problema, de tal manera que puede usar esa solución un millón de veces más, sin hacer jamás la misma cosa dos veces.

Los patrones pueden dividirse o clasificarse en:

- Patrones de Arquitectura: Relacionados a la interacción de objetos dentro o entre niveles arquitectónicos, problemas arquitectónicos, adaptabilidad a requerimientos cambiantes, rendimiento, modularidad, acoplamiento.
- Patrones de Diseño: Que fueron construidos dado los problemas con la claridad de diseño, multiplicación de clases, adaptabilidad a requerimientos cambiantes, proponiendo como solución: Comportamiento de factoría, Clase-Responsabilidad-Contrato (CRC).
- Patrones de Análisis: Usualmente específicos de aplicación.
- Patrones de Proceso o de Organización: Que tratan lo concerniente con el desarrollo o procesos de administración de proyectos, o técnicas, o estructuras de organización.
- Patrones de Idioma: Que reglan la nomenclatura en la cual se escriben, se diseñan y desarrollan los sistemas.

Patrones de diseño, implicación en la experiencia del usuario

Todos los días se ponen en funcionamiento nuevas aplicaciones de negocio en redes de área extensa (wide area networks – WAN). Las aplicaciones en red que dan buen rendimiento a los usuarios facilitan los negocios saneados y prósperos. Pero cuando las aplicaciones que tienen buen rendimiento a través de redes de área local (local area networks – LAN) se trasladan a una WAN, las características del diseño de las aplicaciones y la distancia frecuentemente suponen un serio obstáculo a una experiencia satisfactoria por parte del usuario. A menudo tales retos de negocio frustran al usuario, menoscaban el rendimiento y comprometen los objetivos de negocio.

Mejorar el rendimiento de las aplicaciones en red tiene un determinado valor de negocio, cuya naturaleza y magnitud varían dependiendo de quién realiza la evaluación. La experiencia de los usuarios es la medida óptima del rendimiento y, dependiendo del papel del evaluador dentro de su organización, los beneficios de negocio, asociados a una buena experiencia del usuario, se perciben de diferente manera.

Muchos patrones de diseño están orientados a mejorar el rendimiento de las aplicaciones. La denormalización, el empleo de patrones *Observer, Singleton, Data Access Object, Fast-Lane Reader, Front Controller, Page by Page, Session Facade*[3], entre otros son un ejemplo de ellos.

Es por ello que puede afirmarse que uno de los aspectos a tener en cuenta para lograr que un producto de software capte como una oportunidad de negocio, la experiencia del usuario final que lo consume recae desde las decisiones tempranas donde se han escogido los patrones de diseño.

1.3.5. Mantenibilidad del software

Cecilia Bastarriaca[4], arquitecta de software, establece en 2008, la relación entre el costo de mantenimiento y la arquitectura de software[17]. La preocupante comienza con la distribución del costo del software. La profesora en su seminario define cuatro aspectos esenciales dentro del desarrollo de software: Diseño, Implementación, Prueba y Mantenimiento, a los que otorgó un valor de costo de 10 %, 20 %, 15 %, 55 % respectivamente. En el ciclo de mantenimiento del sistema destaca que más del 50 % del tiempo del programador estaba dedicado a comprender el código y la documentación del sistema. La arquitecta en aquel momento afirmó que la arquitectura de software aclara las intenciones, hace explícitas las decisiones y permite el análisis a nivel de sistema; razón que contrasta directa e indirectamente con los costos de mantenimiento del mismo software.

Otros aspectos paralelos a la reutilización de componentes[5] y que vienen intencionados en la re-

[3]Se han nombrado los patrones de acuerdo a sus nombres originales del idioma Inglés.

[4]Miembro de la Universidad de Chile, ver reseña y publicaciones en http://www.dcc.uchile.cl/~cecilia/.

[5]Recordar al lector que la reutilización de componentes es el nexo con la Productividad que se persigue en la presente ponencia.

ducción de los costos de mantenimiento del software son expuestos en el seminario de la chilena. Ellos son modificabilidad, escalabilidad, portabilidad y seguridad, los que a medida que el sistema crece, la solución radica en la arquitectura.

1.4. Fábrica de software para sistemas de gestión integral de documentos y archivos, productos y servicios

En Junio de 2009 se realizó una propuesta de fábrica de software para implementar Sistemas de Gestión Integral de Documentos y Archivos, la que le valió el título de ingeniero en ciencias informáticas a los autores. Esta investigación hace una análisis del desarrollo de software en la UCI. Para los autores la fábrica tiene la misión de desarrollar soluciones de software orientado a entidades y organismos nacionales e internacionales con el fin de facilitar la producción, control, tramitación, almacenamiento, conservación y difusión de sus documentos administrativos, en cualquier soporte.

El macro proceso de producción de la propuesta establecía:

- Macro Proceso de conceptualización de nueva línea temática: Este proceso sería iniciado cuando la fábrica decidiese iniciar una nueva estrategia de desarrollo de una línea temática.
- Macro Proceso de conceptualización de nuevo producto: Este se iniciaría cuando se hace una solicitud de un producto concreto a la fábrica por parte de un cliente determinado.
- Método de desarrollo específico: En el conjunto de actividades que se desarrollan, ya sea para conceptualizar una nueva línea temática o un producto solicitado por un cliente, se hace necesario culminar con el proceso de desarrollo.
- Procesos de Soporte: En paralelo al desarrollo productivo, se deben desarrollar dentro de la fábrica un grupo de procesos de soporte que se hacen imprescindibles para el buen funcionamiento, eficiencia y organización dentro de la misma. Aquí se había concebido los subprocesos de formación académica, investigación, gestión de los recursos humanos y de la calidad.

1.4.1. El método desarrollo específico

El método de desarrollo específico de la propuesta tenía características propias:

- Hay un rol que ejecuta las tareas y otro supervisa. Por tanto hasta que el que supervisa no apruebe la propuesta, esta no va a Revisión Técnica Formal (RTF).
- Se utilizan solamente los artefactos necesarios para documentar el producto.
- Se basa en la reutilización de productos tecnológicos temáticos.
- Los flujos se integran a través de la arquitectura de software y de negocio.
- Se hacen pruebas continuas sobre los productos y los cambios se hacen a tiempo. Antes de poner un producto tecnológico temático o una línea temática desarrollada en el repositorio se hacen pruebas unitarias, y cuando se va a utilizar para desarrollar un producto a la medida se hacen pruebas generales sobre el mismo, de igual manera se hace el mismo procedimiento antes de liberar el producto.
- Las áreas de servicio están especializadas en temas de apoyo a la producción que es el elemento fundamental de la Fábrica.

La arquitectura de software de este método de producción está estrechamente ligada al concepto de productividad que no es más que la relación entre lo producido y los medios empleados. El vínculo existe gracias al término *Reutilización de Componentes*.

El estudio de Sametinger revela algunos indicadores acerca de la reutilización de componentes en el desarrollo de software:

- Entre el 40 % y 60 % del código fuente de una aplicación es reutilizable en otra similar.
- Aproximadamente el 60 % del diseño del código fuente de aplicaciones administrativas es reutilizable.

- Aproximadamente el 75 % de las funciones son comunes a más de un programa.
- Solo el 15 % del código encontrado en muchos sistemas es novedoso a una aplicación específica. El rango general de uso recurrente está entre el 15 % y el 85 %.[18]

A partir de estos indicadores es fácil deducir el impacto y la importancia que tiene la reutilización de componentes en el proceso de desarrollo de software. Cuánto más sea capaz de reutilizarse en el método de desarrollo específico de la fábrica, menos esfuerzo habrá que invertirse en el desarrollo continuo de personalizaciones del GDA eXcriba, haciendo el negocio más productivo.

1.5. Métodos de evaluación de la arquitectura

El objetivo de evaluar una arquitectura es saber si puede habilitar los requerimientos, atributos de calidad y restricciones para asegurar que el sistema a ser construido cumple con las necesidades de los stakeholders. La evaluación de una arquitectura de software es una tarea no trivial, puesto que se pretende medir propiedades del sistema en base a especificaciones abstractas, como por ejemplo los diseños arquitectónicos. Por ello, la intención es más bien la evaluación del potencial de la arquitectura diseñada para alcanzar los atributos de calidad requeridos. Las mediciones que se realizan sobre una arquitectura de software pueden tener distintos objetivos, dependiendo de la situación en la que se encuentre el arquitecto y la aplicabilidad de las técnicas que emplea. Algunos de estos objetivos son: cualitativos, cuantitativos y máximos y mínimos teóricos[19]

- La **medición cualitativa** se aplica para la comparación entre arquitecturas candidatas y tiene relación con la intención de saber la opción que se adapta mejor a cierto atributo de calidad. Este tipo de medición brinda respuestas afirmativas o negativas, sin mayor nivel de detalle.
- La **medición cuantitativa** busca la obtención de valores que permitan tomar decisiones en cuanto a los atributos de calidad de una arquitectura de software. El esquema general es la comparación con márgenes establecidos, como lo es el caso de los requerimientos de desempeño,

para establecer el grado de cumplimiento de una arquitectura candidata, o tomar decisiones sobre ella. Este enfoque permite establecer comparaciones, pero se ve limitado en tanto no se conozcan los valores teóricos máximos y mínimos de las mediciones con las que se realiza la comparación.

- La **medición de máximo y mínimo teórico** contempla los valores teóricos para efectos de la comparación de la medición con los atributos de calidad especificados. El conocimiento de los valores máximos o mínimos permite el establecimiento claro del grado de cumplimiento de los atributos de calidad.[20]

De forma general el planteamiento anterior presenta los objetivos para efectos de la medición de los atributos de calidad. Sin embargo, en ocasiones, la evaluación de una arquitectura de software no produce valores numéricos que permiten la toma de decisiones de manera directa. Ante la posibilidad de efectuar evaluaciones a cualquier nivel del proceso de diseño, con distintos niveles de especificación, Rick Kazman[6] propone un esquema general en relación a la evaluación de una arquitectura con respecto a sus atributos de calidad. En este sentido, Kazman y sus colegas afirman que de la evaluación de una Arquitectura no se obtienen respuestas del tipo "si–no", "bueno–malo" o "6.75 de 10", sino que explica cuáles son los puntos de riesgo del diseño evaluado[19]

1.5.1. Características de una Evaluación de Arquitectura

- Es uno de los principales puntos de evaluación dentro del proyecto, ya que errores en ella, pueden traer que el proyecto fracase.

- Puede ser realizada por gente Interna o Externa al proyecto, aunque lo más interesante es que sea realizada por gente Externa (Mentores o Arquitectos del Área).[21]

[6] Miembro del Software Ingineering Institute (En español Instituto de Ingeniería de Software), de la Universidad de Carnegie Mellon, reseña disponible en http://www.sei.cmu.edu/about/people/rkazman.cfm.

1.5.2. ¿Por qué evaluar una Arquitectura?

"El propósito de realizar evaluaciones a la arquitectura, es para analizar e identificar riesgos potenciales en su estructura y sus propiedades, que puedan afectar al sistema de software resultante, verificar que los requerimientos no funcionales estén presentes en la arquitectura, así como determinar en qué grado se satisfacen los atributos de calidad. Cabe señalar que los requerimientos no funcionales también son llamados atributos de calidad".[1] Un atributo de calidad es una característica de calidad que afecta a un elemento. Donde el término "característica" se refiere a aspectos no funcionales y el término "elemento" a componente.[1]

- Cuanto más temprano se encuentre un problema en un proyecto de software, mejor.[21]
- Realizar una evaluación de la arquitectura es la manera más económica de evitar desastres.[21]
- Analizar y evaluar la calidad exigida por los usuarios.[21]
- Decisiones de diseño.[21]
- Restricciones de Implementación.
    - Fija la estructura organizacional, tanto del desarrollo, construcción y ejecución del sistema.
    - Logra los atributos de calidad.
    - Permite el prototipado.
    - Permite estimaciones más certeras.
- Abstracción transferible entre sistemas.[21]

1.5.3. ¿Cuándo una Arquitectura puede ser evaluada?

Es posible realizarla en cualquier momento según Kazman, pero propone dos variantes que agrupan dos etapas distintas: temprano y tarde.[19]

- Temprana: No es necesario que la arquitectura esté completamente especificada para efectuar la evaluación, y esto abarca desde las fases tempranas de diseño y a lo largo del desarrollo.
- Tarde: Cuando ésta se encuentra establecida y la implementación se ha completado. Es el caso general que se presenta al momento de la adquisición de un sistema ya desarrollado.

1.5.4. Reglas de oro para determinar el momento de realizar evaluaciones[1]

1. Realizar una evaluación cuando el equipo de desarrollo inicia a tomar decisiones que afectan directamente a la arquitectura.
2. Cuando el costo de no tomar estas decisiones podría tomar más, que el costo de realizar una evaluación.

1.5.5. ¿Quiénes participan en una Evaluación?

Generalmente las evaluaciones a la arquitectura se hacen por miembros del equipo de desarrollo, arquitecto, diseñador, entre otros. Sin embargo puede haber también situaciones en las que intervengan personas especialistas en el tema. Otro que también se interesa por los resultados de una evaluación es el cliente, ya que en dependencia de los resultados puede tomar decisiones de continuar o no con el proyecto.[1]

1.5.6. Técnicas de Evaluación

Existen un grupo de técnicas para evaluar que se clasifican en cualitativas y cuantitativas[21]:

- Técnicas de cuestionamiento o cualitativas: Utilizan preguntas cualitativas para preguntarle a la arquitectura.
- Cuestionarios. En momentos tempranos.
- Escenarios: Especificas del Sistema para arquitecturas avanzadas.

- Listas de chequeo: Especifico del Dominio de la aplicación.
- Técnicas de mediciones: Sugiere hacerle medidas cuantitativas a la arquitectura. Utiliza métricas arquitectónicas, como acoplamiento, cohesividad en los módulos, profundidad en herencias, modificabilidad.
- Simulaciones, prototipos y experimentos.

Por lo regular, las técnicas de evaluación cualitativas son utilizadas cuando la arquitectura está en construcción, mientras que las técnicas de evaluación cuantitativas, se usan cuando la arquitectura ya ha sido implantada[1]

1.5.7. ¿Por qué cualidades puede ser evaluada una Arquitectura?

A grandes rasgos, Bass establece una clasificación de los atributos de calidad en dos categorías[19]:

- Observables vía ejecución: aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución. La descripción de algunos de estos atributos se presenta en la tabla 1.
- No observables vía ejecución: aquellos atributos que se establecen durante el desarrollo del sistema. La descripción de algunos de estos atributos se presenta en la tabla 2.

Atributo de Calidad	Descripción 2.0
Disponibilidad	Es la medida de disponibilidad del sistema para el uso (Barbacci et al, 1995).
Confidencialidad	Es la ausencia de acceso no autorizado a la información (Barbacci et al, 1995).
Funcionalidad	Habilidad del sistema para realizar el trabajo para el cual fue concebido (Kazman et al., 2001).

Continúa en la próxima página.

Atributo de Calidad	Descripción 2.0
Desempeño	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. (IEEE 610.12).
Confiabilidad	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo (Barbacci et al., 1995).
Seguridad externa	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información (Barbacci et al, 1995).
Seguridad interna	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos (Kazman et al., 2001).

Tabla 1. Descripción de atributos de calidad observables vía ejecución[19]

Atributo de Calidad	Descripción 2.0
Configurabilidad	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema (Bosch et al., 1999).
Integrabilidad	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados. (Bass et al. 1998).
Integridad	Es la ausencia de alteraciones inapropiadas de la información (Barbacci et al., 1995).
Modificabilidad	Es la habilidad de realizar cambios futuros al sistema. (Bosch et al. 1999).
Configurabilidad	Es la capacidad de someter a un sistema a reparaciones y evolución (Barbacci et al., 1995). Capacidad de modificar el sistema de manera rápida y a bajo costo (Bosch et al. 1999).

Continúa en la próxima página.

Atributo de Calidad	Descripción 2.0
Portabilidad	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos (Kazman et al., 2001).
Reusabilidad	Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones (Bass et al. 1998).
Escalabilidad	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental (Pressman, 2002).
Capacidad de Prueba	Es la medida de la facilidad con la que el software, al ser sometido a una serie de pruebas, puede demostrar sus fallas. Es la probabilidad de que, asumiendo que tiene al menos una falla, el software fallará en su próxima ejecución de prueba (Bass et al. 1998).

Tabla 2. Descripción de atributos de calidad no observables vía ejecución[19]

1.6. Métodos de Evaluación de Arquitecturas

- ATAM (Architecture Trade-off Analysis Method): Está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM (Software Architecture Analysis Method). El nombre del método ATAM surge del hecho de que revela la forma en que una arquitectura específica satisface ciertos atributos de calidad, y provee una visión de cómo los atributos de calidad interactúan con otros. El método se concentra en la identificación de los estilos arquitectónicos o enfoques arquitectónicos utilizados. Estos elementos representan los medios empleados por la arquitectura para alcanzar los atributos de calidad, así como también permiten describir la forma en la que el sistema puede crecer, responder a cambios,

e integrarse con otros sistemas, entre otros.[19] El método de evaluación ATAM comprende nueve pasos, agrupados en cuatro fases (Presentación, Investigación y Análisis, Pruebas y Reporte).

- Bosch (2000): Que plantea que: "El proceso de evaluación debe ser visto como una actividad iterativa, que forma parte del proceso de diseño, también iterativo. Una vez que la arquitectura es evaluada, pasa a una fase de transformación, asumiendo que no satisface todos los requerimientos. Luego, la arquitectura transformada es evaluada de nuevo".[19] Este método consta de 5 pasos divididos en dos etapas.

- ADR (Active Design Review): "ADR es utilizado para la evaluación de diseños detallados de unidades del software como los componentes o módulos. Las preguntas giran en torno a la calidad y completitud de la documentación y la suficiencia, el ajuste y la conveniencia de los servicios que provee el diseño propuesto".[19]

- ARID (Active Reviews for Intermediate Design). ARID es un método de bajo costo y gran beneficio, el mismo es conveniente para realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo.[19] ARID es un híbrido entre ADR y ATAM.[22] Se basa en ensamblar el diseño de los stakeholders para articular los escenarios de usos importantes o significativos, y probar el diseño para ver si satisface los escenarios. Como resultado de la aplicación de dicho procedimiento se obtiene un diseño de alta fidelidad acompañado de una alta familiarización con el diseño de los stakeholders. Este método consta de 9 pasos agrupados en dos fases (Actividades Previas y Evaluación).[1]

- Losavio (2003): Es un método para evaluar y comparar arquitecturas de software candidatas, que hace uso del modelo de especificación de atributos e calidad adaptado del modelo ISO/IEC 9126. La especificación de los atributos de calidad haciendo uso de un modelo basado en estándares internacionales ofrece una vista amplia y global de los atributos de calidad, tanto a usuarios como arquitectos del sistema, para efectos de la evaluación.[19] El método contempla siete actividades.

1.7. Conclusiones parciales

- El crecimiento actual de la información a nivel nacional e internacional ha llevado que sistemas como eXcriba tengan una necesidad de ampliar las oportunidades de negocio.
- La arquitectura del software es uno de los elementos claves en el aumento de la capacidad de desarrollo del sistema eXcriba toda vez que contribuya a:
  - Hacerlo productivo, concibiendo componentes reutilizables, evaluando los aspectos de modificabilidad, escalabilidad, portabilidad y seguridad.
  - Reducir los costos de mantemiento haciendo que la arquitectura deje clara las intenciones, explícitas las decisiones y la evaluación del análisis a nivel de sistema.
- La arquitectura de software aumenta la experiencia del usuario con el producto final si incluye desde edades tempranas los patrones de diseño que aumenten el rendimiento del sistema.
- Los requisitos de atributos de calidad son las principales guías para el diseño de la arquitectura, dado que están orientados a crear un producto que:
  - Signifique al usuario final un producto con comportamiento, rendimiento, seguridad, confiabilidad y usabilidad.
  - Para el grupo de mantenimiento cuente con un alto grado de modificabilidad y para el cliente que lo requiere implique bajos costos, rápidez y pocos cambios
- El método ATAM evalúa con más profundidad, en relación con otros métodos, cuestiones referentes a la arquitectura, como son: los atributos de calidad.
- Hasta el momento se han presentado varios métodos de evaluación de arquitectura algunos más completos que otros, pero independientemente de esto todos tienen algo en común: Utilizar la técnica de escenarios como vía de constatar en qué medida la arquitectura responde a los atributos de calidad.

Capítulo 2

Propuesta de diseño arquitectónico para el GDA eXcriba

Los casos de usos han sido adoptados casi universalmente para la captura de requisitos de sistemas de software en general, y de sistemas basados en componentes en general[...]. Dirigen el proceso de desarrollo en su totalidad. Los casos de uso son la entrada fundamental cuando se identifican y especifican clases, subsistemas e interfaces, cuando se identifican y especifican casos de pruebas, y cuando se planifican las iteraciones del desarrollo y la integración del sistema. Para cada iteración, nos guían a través del conjunto completo de flujos de trabajo, desde la captura de requisitos, pasando por el análisis, diseño e implementación, hasta la prueba, enlazando estos diferentes flujos de trabajo.[1][23]
Ésta sería la definición previa que los creadores del proceso unificado anticiparan antes de escribir el importante capítulo 4 de la obra *El Proceso Unificado de Desarrollo de Software* donde acotarían de una vez y por todas que los casos de usos no son suficientes, pues para conseguir un sistema completo de trabajo se necesitan más "cosas", las que son conocidas como la arquitectura de software[23].

2.1. El contexto de eXcriba

eXcriba surge como un grupo de proyecto que intenta implementar soluciones de gestión documental a finales de 2007, cuya misión, visión y objetivos son los siguientes:

[1]Definición anticipada de los autores, usada sólo con propósitos pedagógicos

Misión:

eXcriba tiene la misión fundamental de brindar productos y servicios para la gestión documental que apoyen a las actividades de los procesos documentales, siguiendo políticas y estándares nacionales e internacionales.

Visión:

- Tener un producto de gestión documental con gran impacto nacional e internacional.
- Ser el equipo multidisciplinario comprometido y capacitado para brindar productos y servicios de excelencia que desee el cliente.

Objetivo General:

Desarrollar productos y brindar servicios que cumplan con los requisitos de la gestión documental y satisfagan las necesidades del cliente.

Principales servicios que realiza el grupo de proyecto:

- Comercialización del producto insignia del grupo.
- Personalización del software.
- Consultorías de procesos documentales empresariales.
- Servicios de migración documental.
- Soporte y capacitación a los clientes que adoptan parte de la solución.

2.1.1. Composición del equipo de desarrollo

Comprender la composición del equipo de desarrollo da una idea del alcance que puede tener la propuesta de arquitectura de software.

El equipo que actualmente mantine y desarrolla la versión 1.0 de eXcriba, cuenta con amplios conocimientos sobre el uso del marco de trabajo CodeIgniter, los lenguajes de programación PHP, JavaScript, los de marcado HTML, XML, los de formato de notación literal JSON, un tanto menor, pero no despreciable de los lenguajes JAVA, y la tecnología base sobre la que está implementada Alfresco.

2.1.2. Aspectos generales de la arquitectura

El software que compone eXcriba no puede verse como una aplicación única, escrita y desarrollada a la medida de las necesidades del cliente sino como dijera el Sr. Potts en su obra *Alfresco developer guide*[24] *"La libertad del repositorio de Alfresco, particularmente su disponibilidad de ser expuesto fácilmente como un conjunto de servicios hacen de Alfresco una plataforma ideal para las aplicaciones centradas en contenidos*[2]*[...]."* Reafirmando el hecho, la evolución del GDA eXcriba ha atravesado una amplia gama de soluciones que intentan encontrar, siempre como base el repositorio de contenidos Alfresco, las soluciones a los problemas de la gestión documental. Es por ello que se han derivado un conjunto de soluciones que competen el desarrollo de aplicaciones de escritorio y el desarrollo de aplicaciones web.

Sólo el documento digital tiene esta imagen.

Figura 1. Distribución de soluciones eXcriba para la gestión documental

La figura anterior es una vista general y de alto nivel de algunas de las soluciones que el grupo del proyecto eXcriba ha podido desarrollar. Sin embargo el proyecto que más fuerza cobra dentro de esta gama de soluciones es el cliente web de igual nombre que el grupo[3]. En el mismo año 2009, otro trabajo de diploma, titulado *Implementación de la Interfaz Web para el Gestor de Contenido Empresarial Alfresco* tenía como objetivo principal: Implementar una interfaz amigable al usuario, para

[2]Traducción libre de la frase original: *The openness of the Alfresco repository, particularly its ability to be easily exposed as a set of services, makes Alfresco an ideal platform for content-centric applications.*

[3]Citado en el apartado anterior como "producto insignia del grupo"

una adecuada Gestión Documental y de Archivo. El desarrollo de esta tesis ha tenido un significado sustancial para la evolución del cliente web de eXcriba que ha sido concebido y diseñado para que los usuarios involucrados en los procesos de gestión documental y administrativos de las instituciones a las que pertenecen, cuenten con una interfaz amigable y adaptable a sus necesidades. Esto hace que dicha aplicación tenga una importancia relevante en relación a las demás, pues es la arteria principal convergente con el cumplimiento de los objetivos del grupo.

Para la implementación de este cliente web se optó usar PHP como lenguaje de programación, CodeIgniter como marco de trabajo para escribir la lógica de negocio y JQuery debajo de la interfaz de usuario.

Sólo el documento digital tiene esta imagen.

Figura 2. Vista general de la arquitectura de eXcriba 1.0

La imagen anterior hace referencia a la distribución en capas de la solución de eXcriba 1.0. En ella es posible apreciar la forma en que colaboran los principales componentes o elementos, los cuales pueden describirse como:

- **Capa de Presentación:** En esta capa se encuentra el conjunto de interfaces de usuario, que les hace posible al cliente y la aplicación establecer la comunicación, manipular los datos, así como representar en términos de componentes visuales, toda la información necesaria, consultada y/o generada por el par aplicación-usuario. La comunicación entre esta capa y la subyacente es mediante el protocolo HTTP.

- **Capa de Aplicación:** En esta capa se ejecutan todos los procesos de negocio que han sido previamente implementados, se preparan a su vez las transformaciones de datos, sirviendo como un mediador entre las demandas del cliente y las respuestas de los datos. Controla y dirige el flujo de la aplicación en sentido general. La comunicación entre esta capa y la subyacente es mediante el protocolo SOAP.

- **Capa de Acceso a Datos:** Por demás, esta capa que se implementa encima de la API remota que brinda el repositorio de contenidos Alfresco, es la encargada de interactuar directamente con el repositorio de contenidos, permitiéndole a la capa de aplicación abstraerse de la forma en que deben persistir los datos, en su totalidad y cómo deben ser recuperados. En general esta capa escrita con el SDK de Alfresco para PHP le permite a eXcriba 1.0 acceder y manipular directamente el repositorio de contenidos. La comunicación entre esta capa y la subyacente es mediante bibliotecas de clase.

- **Repositorio de Contenidos:** El repositorio de contenidos de Alfresco es toda una infraestructura que escapa más allá del propósito de esta investigación, y que ha sido vista desde este ámbito como una sola capa, cuya similitud con los sistemas de almacenamiento de datos sólo existe en un nivel abstracto donde el repositorio sí almacena toda la información que circula por él[4].

eXcriba 1.0 fue acotado dentro de los requerimientos, restricciones de diseño y atributos de calidad siguientes:

1. **Requerimientos:**

   RF 1 Autenticar Usuario.

   RF 2 Gestionar Espacios.

   - RF 2.1 Cortar.
   - RF 2.2 Copiar.
   - RF 2.3 Ver detalles.
   - RF 2.4 Editar.
   - RF 2.5 Eliminar.
   - RF 2.6 Búsqueda.

   RF 3 Gestionar Contenido.

   - RF 3.1 Cortar.

[4]Véase anexo A.1

RF 3.2 Copiar.

RF 3.3 Ver detalles.

RF 3.4 Editar.

RF 3.5 Eliminar.

RF 3.6 Descargar.

RF 3.7 Búsqueda.

RF 4 Gestionar Metadatos.

RF 4.1 Editar.

RF 5 Gestionar Reglas

RF 5.1 Ejecutar.

RF 6 Firmar Contenido.

2. **Restricciones de diseño**

Rendimiento:

- Tiempo de respuestas rápidos, aproximadamente de 2 segundos, al igual que la velocidad de procesamiento de la información.

Hardware de las estaciones de trabajo clientes:

- Requiere como mínimo de RAM 512 MB.
- El disco duro sólo requiere el espacio mínimo que necesite el sistema operativo que está instalado en conjunto con el necesario para que el navegador web funcione, esto no debe exceder los 5 GB de capacidad.

Software del servidor:

- Servidor web Apache 2.2.x.
- Módulo rewrite activo para el servidor Apache.

- PHP 5.0.
- Alfresco 3.0.

Portabilidad:

- Necesidad de que el sistema sea multiplataforma.

Seguridad:

- Garantizar que la información sea editada únicamente por las personas que tienen permisos para realizar esta actividad.
- Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

Interfaz:

- El producto debe ser legible y con colores adecuados, agradables y poco llamativos.
- Construcción de enlaces rápidos o anclas para los documentos muy largos.

3. **Atributos de calidad**

Usabilidad:

- Para utilizar el sistema es necesario poseer conocimientos elementales de computación y sobre el ambiente Web en sentido general.
- El sistema podrá ser usado de forma fácil por cualquier persona que posea permisos en la institución para gestionar los documentos.

Confiabilidad:

- La información contenida en el sistema debe ser totalmente confiable.

Legislabilidad:

- El empleo de este producto no debe violar ninguna ley o licencia por lo que la plataforma escogida para el desarrollo de la aplicación, está basada en la licencia GPLv2.

A manera de observación, es importante destacar que eXcriba 1.0 fue diseñado para darle cumplimiento sólo a los requerimientos que se enunciaron anteriormente, si bien es cierto que el uso de los marcos de trabajo y la aplicación de estilos arquitectónicos como el Modelo-Vista-Controlador garantizan una separación entre las partes y posibilitan la extensión del software, no quedó reflejado en el diseño un mecanismo que respondiera a la agregación o desagregación de los componentes implementados. Justamente lo que más contraste genera en el grupo de proyecto, pues las perspectivas desde cada cliente, varían entre la asimilación, modificación o agregación de un requerimiento.

Usar el protocolo SOAP para lograr la comunicación entre las capas de aplicación y acceso a datos, significó un coste de rendimiento y mantenibilidad imposible de reducir. La capa de servicios de Alfresco para el protocolo SOAP retarda el proceso de comunicación, además, capacitar al equipo con las tecnologías empleadas para extender los servicios de Alfresco por SOAP implica un coste en mantenimiento todavía más alto. La figura 3 ilustra a grandes rasgos las contribuciones que harían falta para extender eXcriba 1.0, si mantiene la misma arquitectura. Cabe señalar que estas contribuciones serían sobre proyectos que el propio equipo de desarrollo de Alfresco mantiene y distribuye a su conveniencia, cuyo propósito no es el de ser extensible sino el de acceder a los servicios que previamente han sido implementados, y que no proveen la mayor flexibilidad, ni el mejor rendimiento. En tanto la alternativa de usar servicios REST permitirá tomar control de los contenidos que se administran, a la vez, se puede proveer un acceso uniforme para una amplia variedad de aplicaciones clientes y de servicios también como lo pueden ser navegadores, portales, motores de búsquedas. La naturaleza distribuida de REST le permiten al repositorio que contiene los distintos documentos que pueden ser entrelazados igual que la web.

Sólo el documento digital tiene esta imagen.

Figura 3. Representación gráfica de los componentes para extender eXcriba 1.0

La interfaz de servicios REST que provee Alfresco es un mecanismo pensado para ser extensible, usando el framework Web Script el cual permite:

- Usar URLs para identificar recursos.
- Usar las propiedades de los métodos HTTP.
- Representación de URIs mediante plantillas.
- Ejecutar un servicio con autenticación de usuario.
- Internacionalización de los mensajes.
- Procesamiento de formularios.
- Acceso a las cabeceras de la petición.
- Procesamiento del cuerpo de la petición.
- Negociación del formato de la representación.
- Control de la caché.
- Asociación de los servicios.

Los beneficios de REST sobre el repositorio de contenidos convierten al framework Web Script[5], en la estrategia ideal para construir eXcriba, pues el desarrollo es mucho más intuitivo, el lenguaje es más acorde con los que domina el equipo de desarrollo. Se pueden personalizar los servicios para hacer las operaciones más puntuales, sin necesidad de construir una API cliente que añada más coste de rendimiento, además el acceso a las funcionalidades del repositorio es mucho más eficiente que por la API remota de Alfresco.

[5]Tecnología sobre la cual se construyen los servicios en Alfresco

2.2. El entorno de desarrollo

Cuando se va a hacer un desarrollo de software de cualquier índole, es buena idea planear e implementar un entorno de desarrollo desde el principio. De forma tal que todo el que se involucre en él sepa cómo construir el proyecto, corregir errores, cómo está organizado, cómo desplegarlo y cómo probarlo.

2.2.1. Herramientas horizontales

Plataforma de desarrollo

El sistema GDA eXcriba por definición es multiplataforma, lo que da la posibilidad de que el software pueda ser desarrollado sobre muchos sistemas operativos, en particular los basados en UNIX, cualquiera de la familia GNU/Linux y los de la familia Windows NT tales como Windows 7, Vista y XP. Como principal recomendación se propone el uso de las plataformas Nova 2011 o Ubuntu Maverick.

2.2.2. Herramientas Verticales

Herramientas de compilación y construcción

Embeber el ECM Alfresco dentro de la solución eXcriba, desde el ámbito de desarrollo, tiene muchas ventajas entre ellas la aparición de Apache ANT una herramienta de construcción y compilación escrita en y para JAVA. Sin embargo, por la cantidad de tareas que incluye, y los aspectos de portabilidad que ofrece al ser completamente escrita en JAVA se propone como herramienta de contrucción y dado que el cliente web de eXcriba estará escrito íntegramente en PHP, se propone el uso del intérprete de PHP para las técnicas de compilación.

Entornos de desarrollo integrados

Zend Studio 7.0.0:

Se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones Web. Consta de dos partes, las funcionalidades de parte del cliente y las del

servidor. Ambas partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP. La interfaz está compuesta por un explorador de archivos, los menús, una ventana de depuración, y otra para mostrar el código de las páginas. Su editor permite escribir los scripts, es bastante útil para la programación en PHP. Además al estar basado en Eclipse permite la integración de constructores personalizados de Apache Ant.

Lenguajes de programación

Los lenguajes de programación necesarios para implementar esta propuesta son:

- PHP 5.3.x:

  El uso de PHP es una restricción heredada en función de la versión 1.0 del sistema y del conocimiento del equipo de desarrollo sobre este lenguaje. Sin embargo es relevante destacar que el lenguaje PHP permite la implementación de sistemas web complejos y multiplataforma con costos bajos y un esquema de seguridad muy completo. PHP es un lenguaje de programación simple y eficiente, hecho pensando en la web.

- JavaScript: JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS)[25]. Al igual que la enumerada anterior, el uso de este lenguaje es de facto una restricción heredada.

Marcos de Trabajo

CodeIgniter:

CodeIgniter es un entorno de desarrollo abierto que permite crear webs dinámicas con PHP. Su principal objetivo es ayudar a que los desarrolladores, puedan realizar proyectos mucho más rápido que creando toda la estructura desde cero.[26]. Para la formulación de la propuesta se propone la versión 1.7.2.

JQuery:

JQuery es una biblioteca de JavaScript rápida y concisa que simplifica el manejo del documento HTML, del los eventos, de las animaciones, y las interacciones Ajax para el desarrollo para agilizar el desarrollo web. JQuery está diseñado para cambiar la forma en que se escribe JavaScript.[27]

Ambos marcos de trabajos han sido heredados del sistema en su versión previa, sin embargo por la excelencia que ofrecen para el cumplimiento de los objetivos de la presente propuesta no se requieren nuevas adiciones. Para la formulación de la propuesta se propone la versión 1.3.2.

Aplicaciones, APIs y bibliotecas adicionales

En la elaboración de esta propuesta de arquitectura se requiere:

- ECM Alfresco Labs 3.0 Stable:

  Versión sobre la cual el equipo de desarrollo de eXcriba, mantiene y extiende. Por lo que, se apuesta por esta versión indistintamente de las versiones libres superiores que existen estables hasta el momento.

- Módulo de PHP para CURL:

  Este módulo integra a PHP, libcurl, una librería creada por Danile Stenberg, que permite la conexión y comunicación con varios tipos de servidores diferentes y con muchos tipos de protocolos diferentes. Actualmente, libcurl soporta los portocolos http, https, ftp, gopher, telnet, dict, file y ldap. Además libcurl también soporta certificados HTTPS, métodos HTTP POST y HTTP PUT, envío por FTP (que también puede realizarse con la extensión ftp de PHP), envío mediante HTTP de archivos en formularios HTML, servidores proxy, cookies y autenticación usuario+contraseña.[28]

Servidores de Aplicaciones

Servidor Web Apache 2.2.x:

El servidor Web Apache provee un mecanismo para ofrecer la aplicación Web a los usuarios. Es estable y tiene la capacidad para soportar aplicaciones críticas. Apache es el servidor Web más usado en sistemas Linux.

Servidor Web Apache Tomcat 6.0.x:

Tomcat es un servidor web con soporte de servlets y JSPs, escrito totalmente en Java, y portable a cualquier plataforma. Su uso sólo tiene el propósito de contener al ECM Alfresco.

2.3. Requerimientos, restricciones de diseño y atibutos de calidad

2.3.1. Requerimientos

A los requerimientos funcionales heredados de la versión anteriores se agregan los siguientes:

- Para lograr un sistema de fácil mantenimiento es necesario contar un mecanismo simple para el acceso y extensión de los servicios del repositorio Alfresco.
- El sistema además debe proveer un mecanismo que reduzca el transporte de datos entre las peticiones. Esto garantizará incrementar la experiencia del usuario y el rendimiento del sistema.
- El sistema debe contar además con un subsistema que maneje la seguridad, de manera que todas las extensiones nuevas que se desarrollen usen un mecanismo único para gestionar la seguridad.

2.3.2. Restricciones de diseño

A las restricciones de diseños heredadas se decide cambiar el protocolo de comunicación entre la capa de aplicación y de acceso a datos, anteriormente SOAP por HTTP, descrito formalmente en la

restricción siguiente:

- Dado que el repositorio de contenidos Alfresco provee una interfaz más flexible implementada con servicios REST, esta debe ser la tecnología que se debe usar para la comunicación con la capa de acceso a datos.

2.3.3. Atributos de calidad

A los atributos de calidad heredados se decide agregar los siguientes:

- Constructibilidad: El sistema debe poder ser construible en un plazo de seis meses.
- Seguridad: El sistema denegará el acceso no autorizado de usuarios.

2.4. Vista lógica

La vista lógica de la arquitectura apoya principalmente los requerimientos funcionales, desde el ámbito en donde ellos son proveídos en términos de servicios a sus usuarios. En esta sección el sistema es descompuesto en un conjunto de abstracciones claves, tomadas principalmente del dominio de aplicación. Se explotan los principios de la abstracción, encapsulación y la herencia. Esta descomposición no es sólo por el bien del análisis funcional, sino también sirve para identificar los mecanismos comunes y elementos de diseño a través de las diferentes partes del sistema. Otras de las particularidades de esta vista son:

- Identifica mecanismos y diseña elementos comunes a través del sistema.
- Utiliza los diagramas de clases y la notación de Booch.
- Utilizar el estilo arquitectónico orientado a objetos.

2.4.1. Subsistemas y Capas

Vista de Paquetes

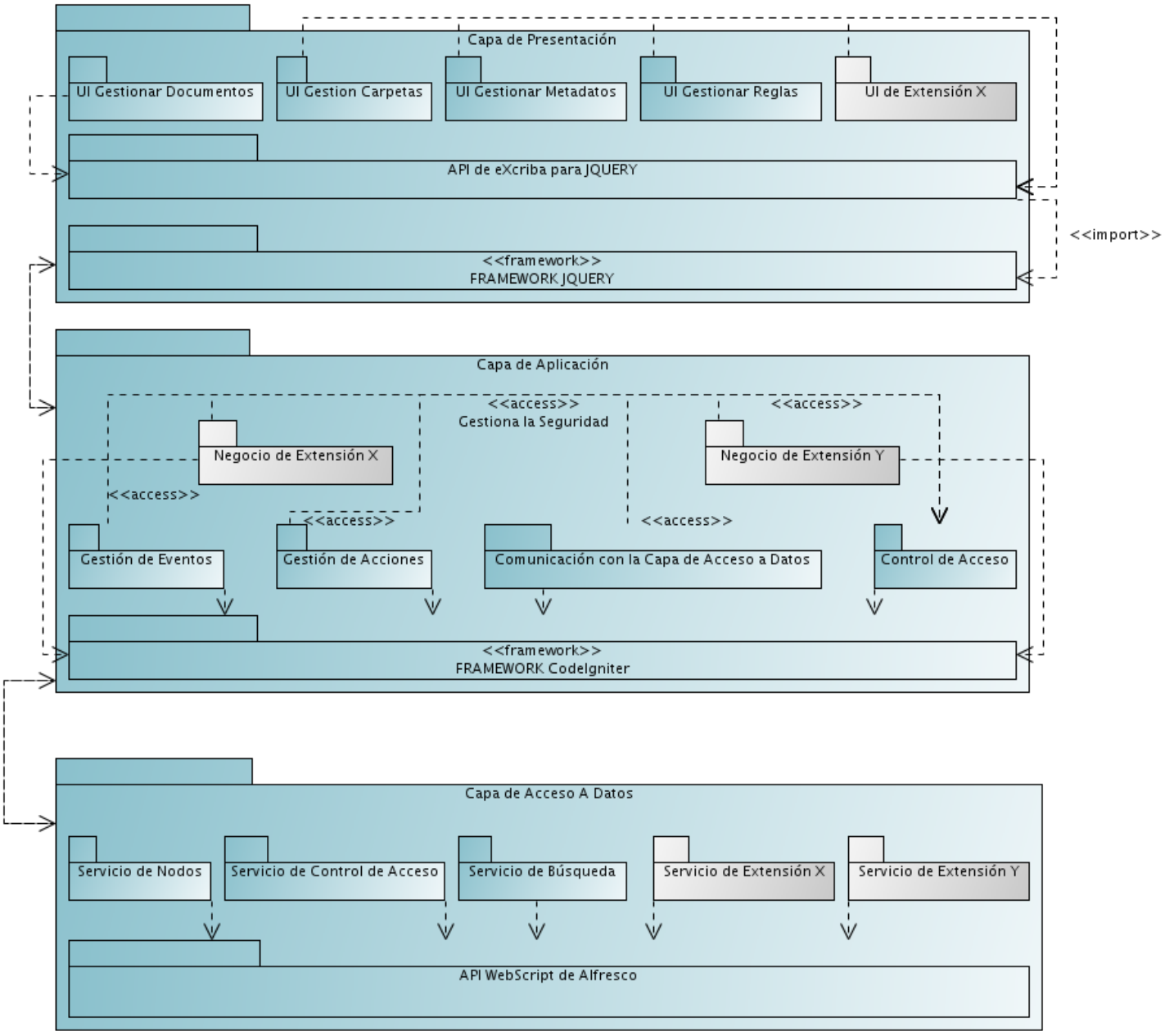

Figura 4. Representación gráfica de la vista de paquetes

Subsistema: Gestión de eventos

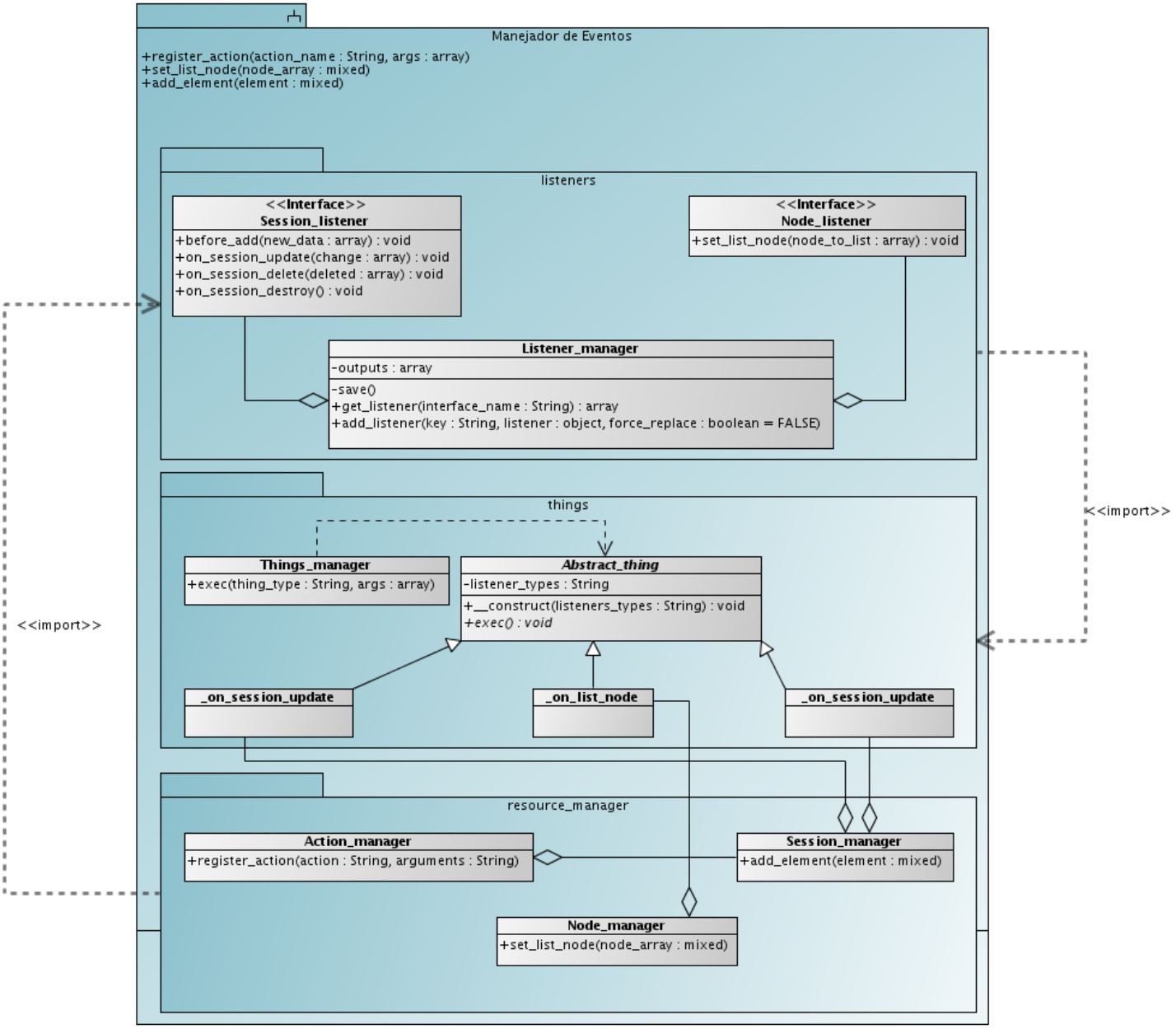

Figura 5. Representación gráfica del subsistema Gestión de Eventos

Subsistema: Gestión de acciones

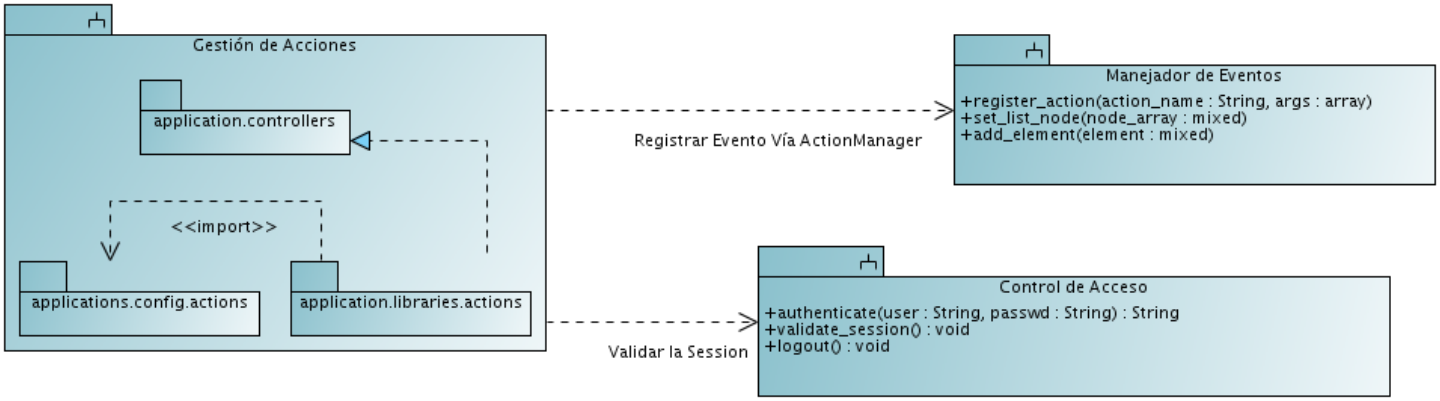

Figura 6. Representación gráfica del subsistema Gestión de Acciones

Subsistema: Control de acceso

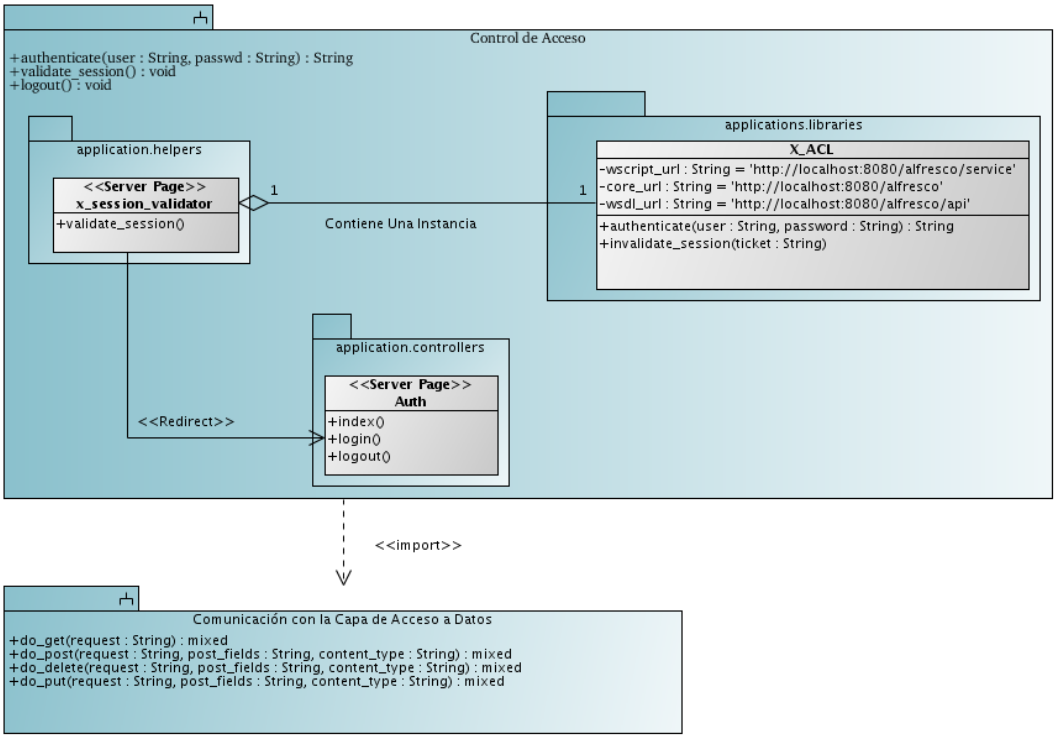

Figura 7. Representación gráfica del subsistema Control de Acceso

Subsistema: Comunicación con la capa de acceso a datos

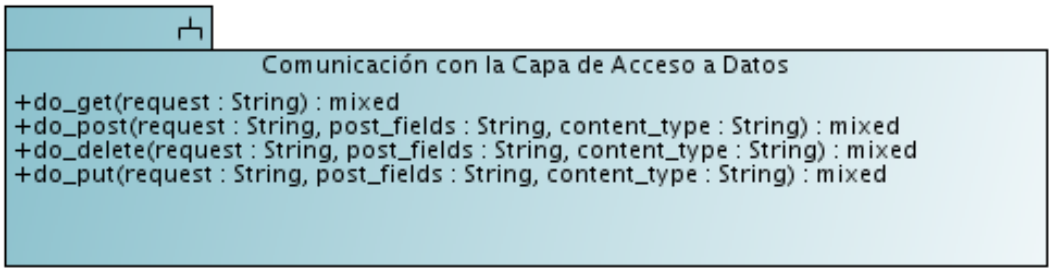

Figura 8. Representación gráfica del subsistema Comunicación con la Capa de Acceso a Datos

2.5. Vista de despliegue

La vista de despliegue se refiere a la implementación en módulos y fragmentación en muchas capas. Colecciona las categorías de clases y grupos. Describe el mapeo del software en el hardware y toma en cuenta los requerimientos funcionales del sistema, tales como: confiabilidad, respuesta y escalabilidad.

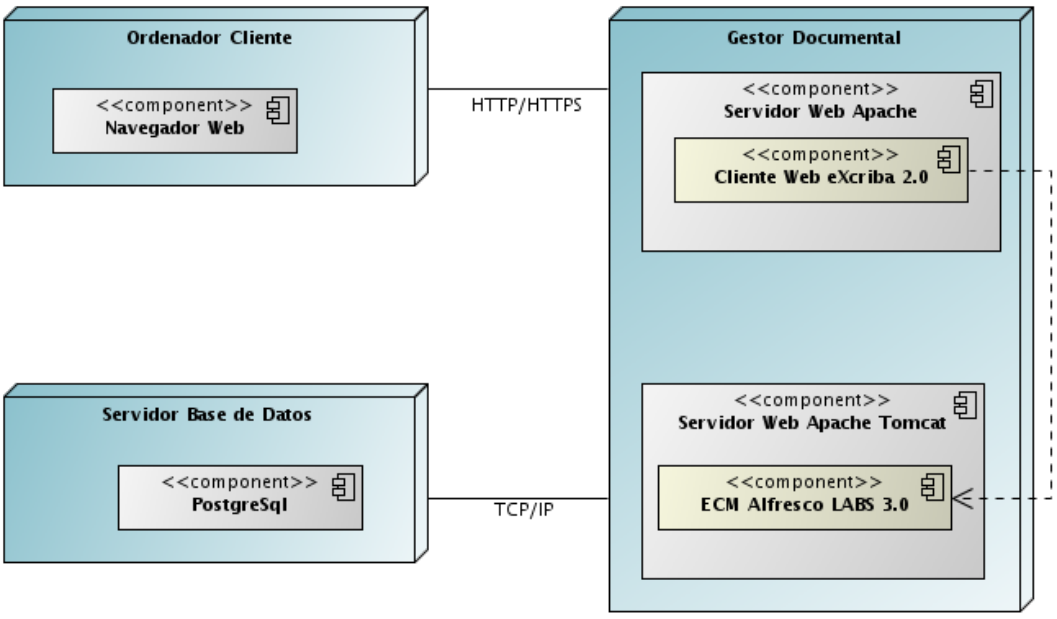

Figura 8. Representación gráfica de la vista de despliegue

- **Ordenador Cliente**:

  Se refiere a las estaciones de trabajo que el usuario utilizará para acceder a la aplicación y transcribir sus datos. En el ordenador cliente debe residir un navegador web.

- **Gestor Documental**:

  Se refiere al núcleo del sistema, donde reside la lógica de aplicación para lograr la conexión del sistema con el Ordenador Cliente se utiliza HTTP como protocolo de comunicación. Dentro del nodo del gestor documental residen dos servidores web: Apache que contiene el cliente web de eXcriba 2.0 y Apache Tomcat que contiene el ECM Alfresco Labs 3.0 Stable.

- **Servidor de Bases de Datos**:

  Se refiere al servidor que radica en cada nodo regional donde se van a estar centralizados los datos recopilados. El servidor de base de datos está especificado por el gestor postgresql.

2.6. Vista de implementación

La vista de implementación contiene la organización de los módulos en términos de paquetes y capas, pueden incluirse también la trazabilidad de la vista lógica. Es representada por un diagrama de componentes o especificaciones de paquetes que son básicamente un subconjunto del modelo de despliegue. Se obtiene cuando se realiza el flujo de trabajo de implementación.

2.6.1. Vista general de la implementación

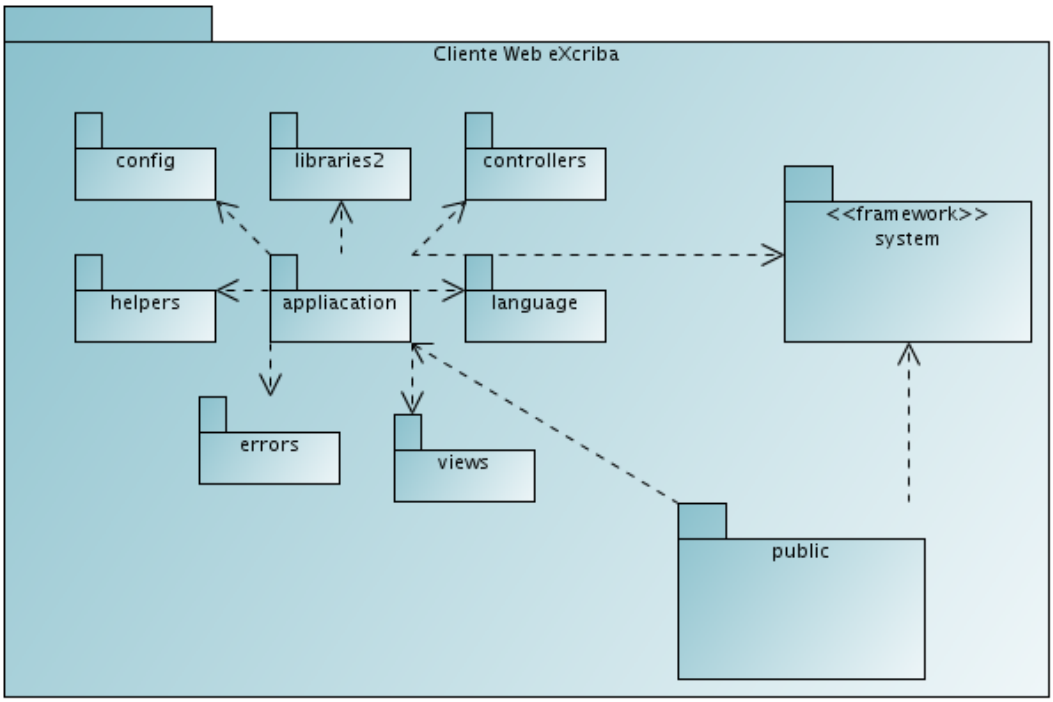

Figura 9. Representación gráfica de la vista de implementación

El marco de trabajo CodeIgniter está construido sobre el estilo arquitectónico Modelo-Vista-Controlador, aunque con un enfoque bastante débil del patrón porque no implementa los requerimientos del modelo. La figura 9, muestra una vista de la distribución física de su estructura de carpetas. Por defecto permite separar la implementación de la aplicación de la propia del framework[6], esto facilita la actualización del framework y la claridad en el desarrollo de cualquier sistema.

Descripción de los paquetes

- **system:** Es el directorio que contiene la implementación concreta del framework CodeIgniter, el cual se puede actualizar, mantener y mejorar sin alterar la implementación de la aplicación.
- **application:** Es el directorio que contiene la implementación particular del sistema que se está

[6]Marcado en la imagen con la dependencia del paquete system por su homólogo application

desarrollando con CodeIgniter, es una copia homóloga de la estructura de carpetas del directorio "system", con la intención de reemplazar aquellas clases nativas de CodeIgniter que deben tener un comportamiento diferente y de contener toda la implementación adicional del sistema que se va a desarrollar.

- **config:** Es el directorio de aplicación donde se almacenan todas las configuraciones necesarias para hacer que la aplicación tome un sentido parametrizado.
- **libraries:** Las bibliotecas son un conjunto de clases que se almacenan dentro del directorio libraries con propósitos generales y que el framework implementa para satisfacer sus propias metas arquitectónicas que pueden reemplazarse, redefinirseles alguna propiedad o comportamiento, o agregarse una nueva con características propias de la aplicación que para nada tienen que ver con las metas arquitectónicas del marco de trabajo.
- **controllers:** CodeIgniter define el concepto "controller" como una clase a la que se le puede asignar una URI. El directorio controllers tiene la responsabilidad de almacenar todos los controladores de la aplicación. Estos son la forma correcta y única por la cual hacer la interacción entre el usuario y la lógica de la aplicación.
- **helpers:** Un helper es una especie de asistente que ayuda a la realización de una tarea específica, consiste en una colección simple de funciones agrupadas en una categoría en particular, como principales características de los helpers, se tiene que no deben tener dependencia con otras funciones, pues cada función del helper debe realizar una única tarea en específico.
- **language:** El directorio language contiene la capacidad de albergar los distintos ficheros de mensajes en los diversos idiomas.
- **errors:** El directorio errors es el encargado de manejar las plantillas de errores.
- **views:** El directorio views almacena las vistas de la interfaz de usuarios generadas apartir de la lógica existente.

- **public:** El directorio public es el encargado de contener la lógica de presentación en el cliente, contiene un espacio para almacenar los ficheros en javascript, las hojas de estilo, así como las imágenes necesarias para crear la interfaz de usuario.

2.6.2. Vista de implementación del manejador de eventos

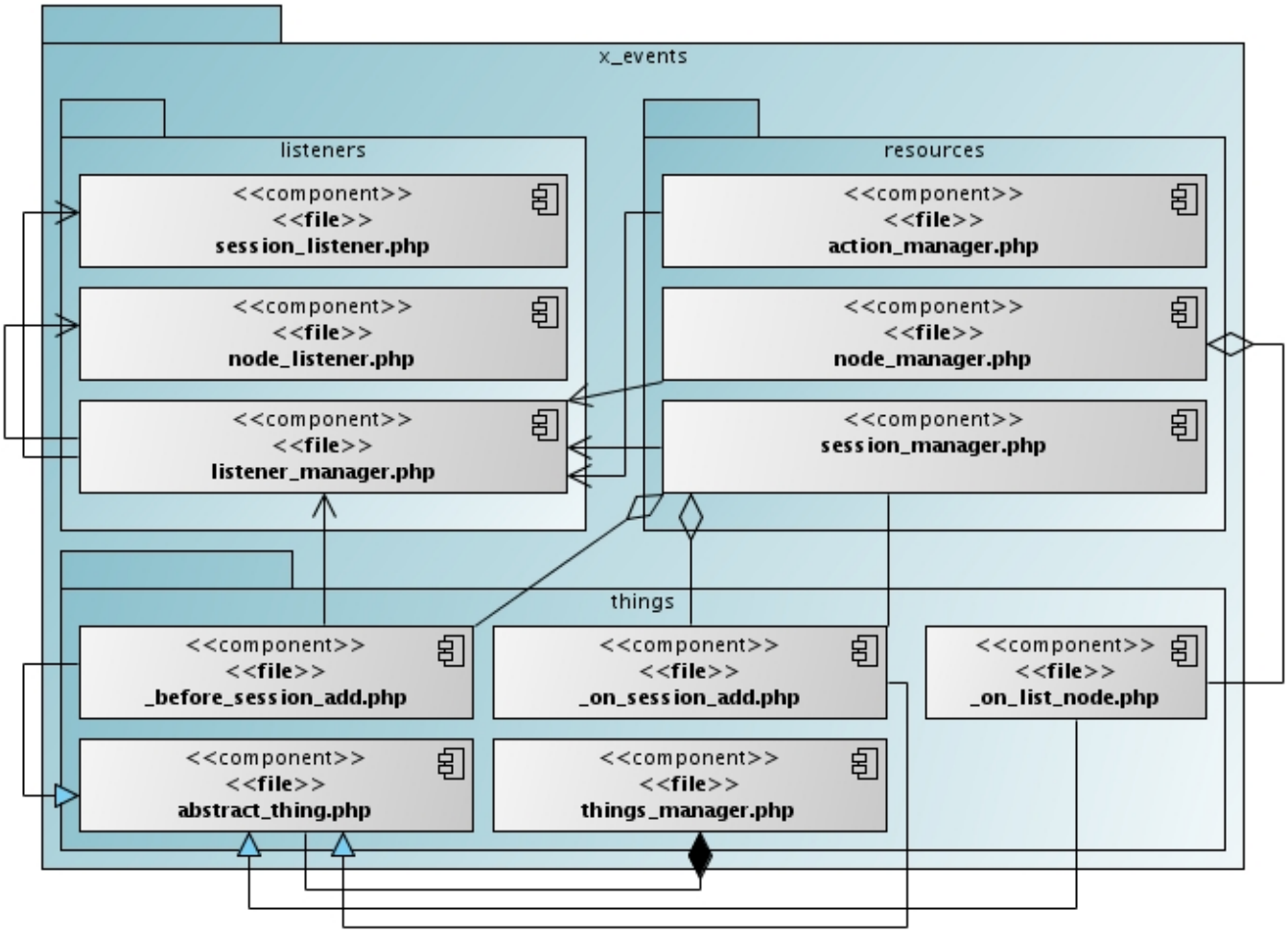

Figura 10. Representación gráfica del manejador de eventos

- **Componente: session_listener.php**

  Este componente implementa la clase interfaz Session_listener del diseño, su ubicación final relativa al paquete application es *libraries/x_events/listeners/session_listeners.php*. Aunque se

encuentra dentro del directorio libraries, su función no es ser una clase biblioteca del sistema, sino la de responder a la descripción del diseño propio de la aplicación.

- **Componente node_listener.php:**

  Este componente implementa la clase interfaz Node_listener, su ubicación final relativa al paquete application es *libraries/x_events/listeners/node_listener.php*. Aunque se encuentra dentro del directorio libraries, su función no es ser una clase biblioteca del sistema, sino la de responder a la descripción del diseño propio de la aplicación.

- **Componente action_manager.php:**

  Este componente implementa la clase Action_manager, su ubicación final relativa al paquete application es *libraries/x_events/resource_manager/action_manager.php*. Aunque se encuentra dentro del directorio libraries, su función no es ser una clase biblioteca del sistema, sino la de responder a la descripción del diseño propio de la aplicación.

- **Componente node_manager.php:**

  Este componente implementa la clase Node_manager, su ubicación final relativa al paquete application es *libraries/x_events/resource_manager/node_manager.php*. Aunque se encuentra dentro del directorio libraries, su función no es ser una clase biblioteca del sistema, sino la de responder a la descripción del diseño propio de la aplicación.

- **Componente session_manager.php:** Este componente implementa la clase Session_manager, su ubicación final relativa al paquete application es *libraries / x_events / resource_manager / session_listener.php*. Aunque se encuentra dentro del directorio libraries, su función no es ser una clase biblioteca del sistema, sino la de responder a la descripción del diseño propio de la aplicación.

- **Componente _before_session_add.php:**

Este componente implementa la clase _before_session_add, su ubicación final relativa al paquete application es *libraries/x_events/things/_before_session_add.php*. Aunque se encuentra dentro del directorio libraries, su función no es ser una clase biblioteca del sistema, sino la de responder a la descripción del diseño propio de la aplicación.

- **Componente _on_session_add.php:**

  Este componente implementa la clase _on_session_add, su ubicación final relativa al paquete application es *libraries/x_events/things/_on_session_add.php*. Aunque se encuentra dentro del directorio libraries, su función no es ser una clase biblioteca del sistema, sino la de responder a la descripción del diseño propio de la aplicación.

- **Componente _on_list_node.php:**

  Este componente implementa la clase _on_list_node, su ubicación final relativa al paquete application es *libraries/x_events/things/_on_list_node.php*. Aunque se encuentra dentro del directorio libraries, su función no es ser una clase biblioteca del sistema, sino la de responder a la descripción del diseño propio de la aplicación.

- **Componente abstract_thing.php:**

  Este componente implementa la clase Abstract_thing, su ubicación final relativa al paquete application es *libraries/x_events/things/abstract_thing.php*. Aunque se encuentra dentro del directorio libraries, su función no es ser una clase biblioteca del sistema, sino la de responder a la descripción del diseño propio de la aplicación.

- **Componente things_manager.php:**

  Este componente implementa la clase Things_manager, su ubicación final relativa al paquete application es *libraries/x_events/things/things_manager.php*. Aunque se encuentra dentro del directorio libraries, su función no es ser una clase biblioteca del sistema, sino la de responder a la descripción del diseño propio de la aplicación.

2.6.3. Vista de implementación del control de acceso

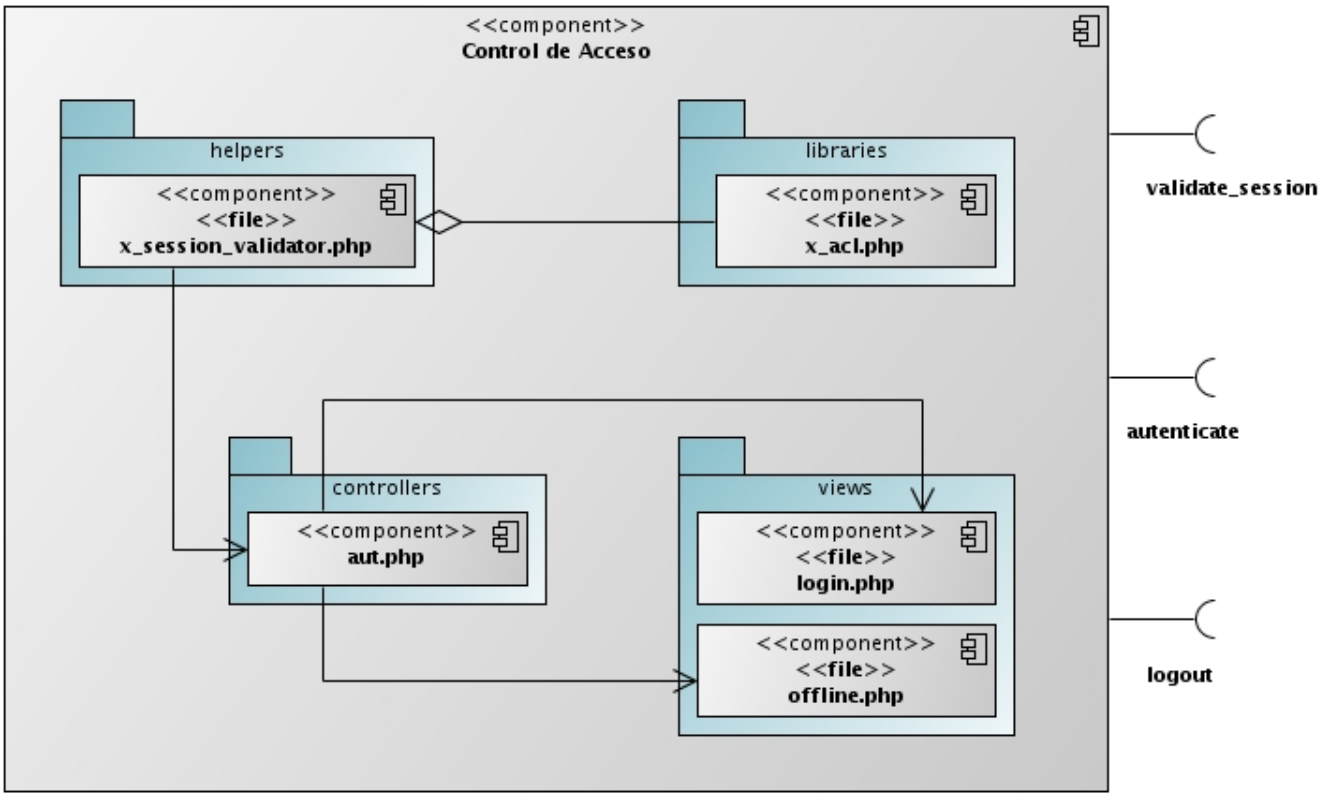

Figura 10. Representación gráfica del componente control de acceso.

El subsistema control de acceso en la vista lógica aquí tiene su homólogo, un componente distribuido físicamente entre los directorios controllers, libraries y helpers, su función principal está indicada en las interfaces del componente, validar la sesión de un usuario previamente autenticado, así como autenticarlo y no autenticarlo.

- **Componente: x_session_validator.php**

  Este componente implementa la página servidora x_session_validator del diseño es un fichero PHP destinado a cumplir los requerimientos de los helpers del marco de trabajo CodeIgniter.

- **Componente x_acl.php:**

Este componente implementa la clase X_ACL del diseño es un fichero PHP destinado a cumplir los requerimientos de las bibliotecas del marco de trabajo CodeIgniter, con la particularidad de que no está destinada a extender recursos del framework, si no que su propósito es el de satisfacer los requerimientos de la aplicación.

- **Componente auth.php:**

  Este componente implementa la página servidora Auth del diseño es un fichero PHP destinado a cumplir los requerimientos de las bibliotecas del marco de trabajo CodeIgniter. Este componente tiene una relación con otros dos ficheros en el paquete vista, su principal objetivo es manejar la aparición u ocurrencia de estos elementos en el flujo de eventos.

- **Componente login.php:**

  Este componente contiene el html necesario para crear una vista de autenticación del sistema.

- **Componente offline.php:**

  Este componente contiene el html necesario para generar una respuesta ante fallos cuando el gestor documental Alfresco no se encuentre disponible.

2.6.4. Vista de implementación de la comunicación con la capa de acceso a datos

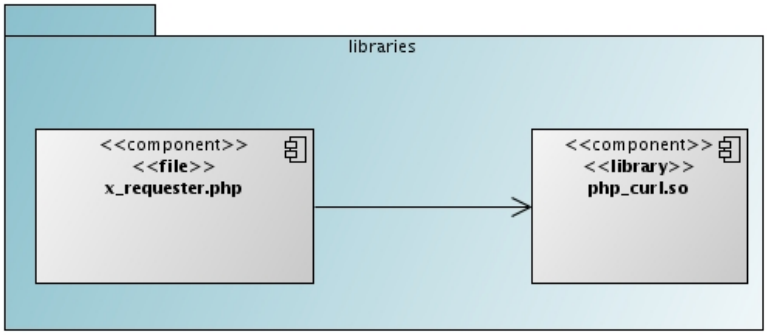

Figura 11. Representación gráfica de la comunicación con la capa de acceso a datos

- **Componente: x_requester.php:**

  Este componente implementa las operaciones del subsistema de la comunicación con la capa de acceso a datos mediante el uso de una extensión de PHP para usar el protocolo HTTP.

- **Componente php_curl.so:**

  Este componente representa la extensión de PHP para el manejo del protocolo HTTP.

2.7. Interfaz de Servicios RESTful

La capa de acceso a datos debe ser escrita completamente sobre el framework web script que brinda el gestor empresarial. UML no tiene ninguna extensión para modelar estilos arquitectónicos RESTful, por lo que en lugar de modelarse en esta sección se describen los componentes arquitectónicos del framework, cómo interactuar con él para que los datos deseados persistan o sean recuperados desde la capa de aplicación.

La interfaz de un web script comprende una URI, métodos HTTP, y formatos de documentos. La implementación, por otra parte, involucra:

- Documento de descripción: Describe la URI del servicio, además provee un nombre corto al servicio, así como la descripción, permitiendo definir el tipo de autenticación y las necesidades de transacción.

- Controlador JavaScript (Opcional): Realiza el trabajo actual. En este fichero se puede consultar al repositorio, construir los datos, conocido como modelos, generar una respuesta para la URI (o varias) que intentan modificar los datos del repositorio.

- Una o varias plantillas de respuesta freemarker: Se le conocen como representaciones, encargadas de generar una respuesta en el formato correcto. Los que pueden ser HTML, ATOM, XML, RSS, JSON, CVS o cualquier combinación de estas. La URL de respuesta es generada mediante una de las plantillas proveídas donde la plantilla escogida está basada en los requerimientos del atributo

content-type de la respuesta o de la salida del estado. Esta plantilla tiene acceso a todos los argumentos de la URI, a los datos comunes del repositorio, y a cualquier dato incluido en el script controlador opcional.

Las implementaciones avanzadas de un web script pueden realizarse en JAVA, los diferentes componentes del web script son: documento de descripción, controlador y plantillas son unidos a través de una convención de documentos.

Los web scripts son registrados y ejecutados por el framework web script, tal y como se muestra en la figura 12. En el anexo A.2 se detallan algunos de los servicios de extensión de eXcriba 2.0.

Sólo el documento digital tiene esta imagen.

Figura 12. Representación gráfica de la arquitectura del framework Web Script

2.8. Conclusiones parciales

La formulación de la propuesta presentada en este capítulo permitió concluir que:

- Se analizó el aspecto general de la arquitectura base de la versión 1.0 del sistema arrojando la necesidad de sustituir el mecanismo de comunicación con la capa de acceso a datos SOAP, por REST.
- La propuesta definió algunos aspectos de gestión de configuración y cambio referidos como herramientas horizontales y verticales, que permiten establecer adecuadamente el entorno de desarrollo.
- Se ajustaron los requerimientos, restricciones y atributos de calidad en condición de agregación o eliminación según las necesidades actuales de la propuesta.
- Se cubrieron los aspectos arquitectónicamente significativos según Krutchen, y una descripción textual de la capa de acceso a datos.
- Finalmente quedó formulada la esencia de la propuesta arquitectónica.

Capítulo 3

Evaluación y aceptación de la arquitectura del GDA eXcriba.

La razón principal que justifica priorizar la arquitectura es demostrar lo más pronto posible que la solución (el diseño) es correcta para resolver el objetivo (los requisitos). La arquitectura es punto de transición entre la fase de ingeniería, donde se toman las decisiones más importantes, y la fase de producción, donde aquellas decisiones se traducen en los mayores gastos del proyecto. En la primera intervienen usualmente un máximo de 2 ó 3 personas, quienes idean la solución y toman las decisiones críticas. En la segunda fase, un equipo de desarrolladores de tamaño variable en función de la complejidad del proyecto, colabora en la implementación de la solución durante un período de tiempo mayor.

La arquitectura es la materialización temprana, inmadura y de mínimo coste de las decisiones más importantes. El prototipo ejecutable de arquitectura debe permitir demostrar que la solución ya es madura, es decir, que tales decisiones son efectivas para resolver los principales casos de uso del software que se va a construir antes de realizar los gastos pertinentes asociados a la fase de producción, en la cual un equipo de desarrolladores implementará e integrará la mayor parte del código durante un período de tiempo casi siempre superior al de la fase de ingeniería.

La arquitectura debe dar garantías de que la solución diseñada es realizable dentro de las restricciones de tiempo, personal, y presupuestos, o sea, que el proyecto es viable.

3.1. Método de la evaluación

Para la selección del método de evaluación se decide escoger el MECABIC[1], propuesto en la UCI, por los arquitectos de ERP[20] y que está deribado del Architecture Tradeoffs Analysis Method (ATAM, por sus siglas en Inglés), por tener este un conjunto de pasos, un equipo de evaluación y un conjunto de salidas mejor definidas y no presenta ninguna restricción con respecto a la característica de calidad a evaluar. MECABIC incluye en alguno de sus pasos un enfoque de integración de elementos, inspirado en la construcción de partes arquitectónicas adoptado por el Architecture Based Design (ABD). Además se propone un árbol de utilidad inicial basado en el modelo de calidad ISO-9126 instanciado para Arquitectura de Software propuesto por Losavio. La adopción de este modelo por parte del MECABIC permite concentrarse en características que dependen exclusivamente de la arquitectura, y al ser un estándar facilita la correspondencia con características de calidad consideradas por los métodos estudiados.

3.1.1. El equipo de la evaluación

El equipo de la evaluación está compuesto por tres grupos de trabajo. La siguiente tabla muestra la composición del equipo de desarrollo

Equipo	Definición	Fases
Arquitectos	Responsables de generar y documentar una Arquitectura de Software para el sistema estudiado.	Todas
Evaluador	Integrado por personas expertas en asuntos de calidad quienes guiarán el proceso de evaluación de la arquitectura.	Todas
Relacionados	Son las personas involucradas de alguna manera con el sistema: programadores, usuarios, gerentes, entre otros.	Fases 1, 3, 4.

Tabla 1. Composición del equipo de desarrollo.

[1]Método de Evaluación de la Calidad de Arquitecturas Basadas en la Integración de Componentes

3.1.2. Instrumentos y técnicas de evaluación

Para la especificación de la calidad se hace uso de un árbol de utilidad, el cual tiene como nodo raíz la "bondad" o "utilidad" del sistema. En el segundo nivel del árbol se encuentran los atributos de calidad. Las hojas del árbol de utilidad son escenarios, los cuales representan mecanismos mediante los cuales extensas y ambiguas declaraciones de cualidades son hechas específicas y posibles de evaluar. La generación del árbol de calidad incluye un paso que permite establecer prioridades. Para el análisis de la arquitectura se utiliza la técnica de escenarios, soportada por cuestionarios, para identificar lo que desean los participantes.

3.1.3. Fases del Método

En la primera fase, la de presentación, se da a conocer el método entre todos los grupos, el sistema y su organización, y finalmente se presenta cuál es la arquitectura que debe ser evaluada.

La segunda es de investigación y análisis, y en ella se determina de qué manera se va estudiar la arquitectura, se pide a los involucrados que expresen de una manera precisa qué escenarios de calidad se desean y se analiza si la arquitectura es apropiada o se requieren modificaciones para que lo sea. En esta fase sólo participan el grupo evaluador y grupo de arquitectos.

La tercera fase es de prueba, consiste en la revisión de la segunda fase y en ella participan todos los grupos.

En la última fase se lleva a cabo la presentación de los resultados. En esta fase participan todos los grupos.

3.1.4. Árbol de utilidad

Característica	Subcaracterística	Escenario
Funcionalidad	Interoperabilidad	El sistema posee componentes capaces de leer datos provenientes de otros sistemas.
		El sistema posee componentes capaces de producir datos para otro sistema
	Precisión	Los resultados ofrecidos por los componentes sistema son exactos.
		La comunicación entre los componentes no altera la exactitud de los datos.
	Seguridad	El sistema detecta la actuación de un intruso e impide acceso a los componentes que manejen información sensible.
		El sistema asegura que los componentes no pierdan datos ante un ataque (interno o externo).
	Obediencia (Compliance)	Los componentes respetan un estándar de fiabilidad.
		La comunicación entre los componentes no viola los estándares de fiabilidad.
Fiabilidad	Madurez	Los componentes del sistema manejan entradas de datos de datos incorrectas.
	Tolerancia a fallas	Todas las operaciones ejecutadas por los componentes se realizan correctamente bajo condiciones adversas.
	Capacidad de restablecimiento o recuperación	Los componentes del sistema no fallan bajo ciertas condiciones especificadas.

Continúa en la siguiente página

Característica	Subcaracterística	Escenario
Fiabilidad	Capacidad de restablecimiento o recuperación	Ante problemas con el ambiente un subconjunto determinado de los componentes puede continuar prestando sus servicios.
Eficiencia	Tiempo de comportamiento	El sistema debe recibir los servicios de sus componentes en el transcurso de un tiempo indicado.
	Recursos utilizados	Los componentes pueden compartir recursos adecuadamente.
		El sistema controla que ningún componente se quede sin recursos cuando los necesita.
Mantenibilidad	Habilidad de cambio, estabilidad, prueba	Es posible verificar el estado de los componentes del sistema.
		El sistema brinda facilidad para adaptar un componente.
		El sistema debe facilitar la sustitución/adaptación de un componente.
Portabilidad	Adaptabilidad	El sistema debe continuar funcionando correctamente aun cuando los servicios de los componentes provistos por el ambiente varíen.
	Capacidad de Instalación	Los componentes pueden instalarse fácilmente en todos los ambientes donde debe funcionar.
	Co-existencia	Los componentes manejan adecuadamente recursos compartidos del sistema.

Tabla 2. Características a medir por el grupo de expertos.

La tabla mostrada anteriormente relaciona las principales características por las cuales el grupo de expertos puede medir un prototipo ejecutable de la arquitectura. Para ello se puede realizar mediante la formulación de un conjunto de preguntas clasificables dentro de cada característica o subcaracterística

de la siguiente manera.

Característica	Subcaracterística	Preguntas del Análisis
Funcionalidad	Precisión	¿Puede la comunicación entre los componentes introducir imprecisiones en los servicios ofrecidos por los componentes?
	Interoperabilidad	¿Dónde se encuentran los componentes que permiten al sistema interactuar con otros sistemas?
Fiabilidad	Madurez	¿Existen decisiones para minimizar el manejo incorrecto de datos en la comunicación entre componentes?
	Tolerancia a fallas	¿Cómo se detecta el funcionamiento incorrecto de un componente?
Eficiencia	Tiempo de comportamiento	¿Cómo es la relación entre el número de componentes de las diferentes partes de la arquitectura?
Mantenibilidad	Habilidad de cambio, estabilidad, prueba	¿Cómo se verifica el funcionamiento correcto de un componente?
		¿Cómo se verifica el estado de una comunicación entre componentes?
		¿Cómo se facilita el reemplazo de un componente?

Tabla 3. Clasificación de las preguntas de acuerdo a las diferentes características.

3.2. Prototipo ejecutable de la arquitectura

Para la elaboración del prototipo ejecutable de la arquitectura se han escogido una serie de requerimientos disponibles en la versión 1.0 del sistema, además se ha propuesto un cambio de interfaz de usuario a solicitud de una personalización del software para los centros productivos de la universidad. Otro aspecto a tener en cuenta, radica en la adición de otros requerimientos funcionales hasta sumar

un total de 51[2], los que fueron implementados completamente.

La estructura física del código fuente del software se dividió en un total de 15[3] proyectos:

- **excriba-core**: Implementación de la arquitectura base, conjunto de funciones comunes, principales subsistemas, aspectos arquitectónicamente significativos.
- **x-accordion**: componente, que gestiona la navegación por carpetas en formas de árbol, así como las distintas zonas de navegación, incluye soporte además para gestión de portapapeles.
- **x-audit**: Implementa los requerimientos de auditoría sobre los documentos y carpetas.
- **x-basic_operation**: Implementa los requerimientos de las operaciones comunes que se realizan con la herramienta tales como copiar, mover, editar metadatos de documentos y carpetas, entre otros.
- **x-breadcrumb**: Implementa los requerimientos de las migas de navegación, historial de los últimos visitados y navegación por caminos (paths).
- **x-classification**: este proyecto implementa la gestión de cuadros de clasificación funcionales.
- **x-global-actions**: En este proyecto se desarrollan los requerimientos de las acciones globales asignadas a la dirección de navegación en que se encuentre el usuario en cada momento.
- **x-control-version**: Implementa los requerimientos de control de versiones sobre los documentos.
- **x-header**: Gestiona la vista de interfaz de usuario referente a la cabecera del software.
- **x-loader**: Da soporte a la manipulación mediante AJAX, a la comunicación entre la capa de presentación y la lógica de negocio.
- **x-permissions**: Gestiona el control de acceso de los documentos y carpetas.

[2]La validez de esta cifra puede ser comprobada en el expediente de proyecto eXcriba

[3]Idem.

- **x-record**: Implementa los requerimientos de gestión de expedientes.
- **x-rules**: Implementa los requerimientos de reglas de negocio.
- **x-search**: Da soporte a los requerimientos de búsqueda y recuperación de información.
- **x-toolbar**: Soporta los requerimientos que por su importancia deben existir en una zona de visibilidad alta del sistema.

3.3. Resultado de la evaluación

En esta sección se le dará respuesta a las principales preguntas del análisis así como se hará una evaluación de otros aspectos que sirvan para emitir el resultado final de la evaluación.

Preguntas del análisis:

- **¿Puede la comunicación entre los componentes introducir imprecisiones en los servicios ofrecidos por los componentes?**: La comunicación entre componentes fluye a través del subsistema de gestión de eventos, haciendo que el patrón observador, comunique un componente con otro únicamente en el momento, y delegando la responsabilidad de exclusión mutua a los miembros comunicados.
- **¿Dónde se encuentran los componentes que permiten al sistema interactuar con otros sistemas?**: El sistema cuenta con un subsistema de comunicación con la capa de acceso a datos, integrado con CURL. Este es el punto dónde se le debe dar soporte para que interactúe con otros sistemas en dependencia del tipo de comunicación que quiera hacerse.
- **¿Existen decisiones para minimizar el manejo incorrecto de datos en la comunicación entre componentes?**: Existen a través de la validación de los datos y soporte de transacciones en la capa de acceso a datos.
- **¿Cómo se detecta el funcionamiento incorrecto de un componente?**: Sin soporte.

- **¿Cómo es la relación entre el número de componentes de las diferentes partes de la arquitectura?**: Cada definición de componente es serializada, y no el contenido completo de información del componente que es guardado en ficheros de configuración (las partes estáticas), almacenadas en el cuerpo del HTML (las partes paramétricas) o reconsultas al repositorio(las partes dinámicas) entre cada transacción; haciendo que persistan la mayor cantidad de componentes, y equilibrándolos con un conjunto de consultas, cortas, inclusiones periódicas (activa el sistema de cacheo), y usando el paso de argumentos por los métodos POST, y GET entre las capas de presentación y aplicación. Esto facilita la reusabilidad de componentes existentes, tanto en el momento de la implementación de un nuevo caso de uso, como a la hora de utilizar las instancias activas del sistema.
- **¿Cómo se verifica el funcionamiento correcto de un componente?**: Un componente funciona correctamente si no manda un mensaje de error, en el cuerpo de la respuesta.
- **¿Cómo se verifica el estado de una comunicación entre componentes?**: Sin soporte.
- **¿Cómo se facilita el reemplazo de un componente?**: El sistema permite el reemplazo de un componente ayudado por ANT y el bajo acoplamiento que existe entre los componentes pues un componente se comunica con otro a través de un proxy, único del sistema, creando un mecanismo de envío y retorno. Todos los componentes envían la solicitud al proxy del núcleo, parte de la arquitectura base, y mediante el mecanismo de gestión de eventos se disparan las diferentes respuestas de cada componente.

3.4. Conclusiones parciales

En la presentación de este capítulo se evaluaron los distintos esenarios propuestos por el método MECABIC, permitiéndo formular un conjunto de preguntas del análisis que responden satisfactoriamente a la aceptación de la arquitectura, a pesar que existen algunas debilidades en cuanto a tolerancia a fallas y habilidad de cambios.

Conclusiones

Al finalizar la presente propuesta se pude afirmar que se realizó un estudio acerca de varios estilos y patrones arquitectónicos que existen en la actualidad, para hacer uso de las mejores técnicas de diseño. Se analizó las formas en que la arquitectura incide sobre la prestación de productos y servicios. Pudo concretarse qué aspectos dentro de ella desencadenan un incremento en la productividad del desarrollo, las mejoras que inciden en su rendimiento, cómo disminuir los costos de mantenibilidad, además del aumento general en la calidad del software.

La arquitectura propuesta permitió desagregar el software en un conjunto de componentes con un bajo nivel de acoplamiento, permiendo así que puedieran ser incorporados o retirados del sistema, sin afectar el funcionamiento del mismo, asegurando aspectos como la reutilización de estos componentes y esto a su vez garantiza una respuesta más rápida en la personalización del software.

Se demostró cuan viable es la arquitectura permitiendo ser aplicada en el desarrollo actual y futuro del sistema GDA eXcriba.

Se cumplieron todos los objetivos trazados en esta investigación y a la vez que se dio respuesta a la pregunta planteada en el problema científico a partir de definir un marco arquitectónico para el GDA eXcriba 2.0.

Recomendaciones

- Hacer un estudio más profundo de la modelación de servicios REST con UML y en caso de no existir ninguna alternativa proponer una normalización sobre el cual modelar estos tipos de servicios.
- Completar el cúmulo de pruebas al prototipo funcional que incluyan aspectos de rendimiento, seguridad y tolerancia a fallos.
- Hacer un estudio de formas más óptimas de implementar el subsistema de gestión de eventos.

Glosario de términos

ADL del inglés, Arquitecture Description Language. Lenguaje de Descripción de la Arquitectura

Ajax del inglés, Asynchronous JavaScript And XML. (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas

ATOM Formato de Redifusión Atom es un fichero en formato XML usado para Redifusión web.

CVS del inglés, Concurrent Versions System. es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto.

ECM del inglés, Enterprise Content Manager. Gestor de contenidos empresariales.

Entrópico Relativo a entropía Medida de la incertidumbre existente ante un conjunto de mensajes, de los cuales se va a recibir uno solo.

GB del Inglés gigabyte. Gigaocteto es un múltiplo del byte u octeto que equivale a 10^9 bytes.

HTTP del inglés, HyperText Transfer Protocol. Protocolo de transferencia de hipertexto es el protocolo usado en cada transacción de la Web (www). El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido.

IDC International Data Corporation. Firma especializada en investigación y análisis de mercado.

JSPs del inglés, JavaServer Pages. Tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo

KB del Inglés Kilobyte. Kiloocteto es un múltiplo del byte u octeto que equivale a 10^3 bytes.

MB del Inglés megabyte. Megaocteto es un múltiplo del byte u octeto que equivale a 10^6 bytes.

Migas de Navegación Es un recurso de la navegación, usado en las interfaces de usuarios que permite guardar el registro de las localidades visitadas dentro de los documentos y/o programas

POSA del inglés, Pattern-Oriented Software Architecture,

RAM del Inglés, Ramdom Access Memory Memoria de acceso aleatorio

RSS del inglés, Really Simple Syndication. Es un formato XML para sindicar o compartir contenido en la web.

RUP del inglés, Rational Unified Process. Proceso Unificado de Rational. Métodología de desarrollo de software

SEI del inglés, Software Engineering Institute. Es un instituto federal estadounidense de investigación y desarrollo, fundado por el Congreso Estadounidense en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software, que dieran respuesta a los problemas que generaba al ejército estadounidense la programación e integración de los sub-sistemas de software en la construcción de complejos sistemas militares. Financiado por el Departamento de Defensa estadounidense y administrado por la Universidad Carnegie Mellon.

Servlets Los servlets, son objetos que corren dentro del contexto de un contenedor de servlets (ej: Tomcat) y extienden su funcionalidad.

Script Un script (cuya traducción literal es guión) o archivo de órdenes o archivo de procesamiento por lotes es un programa usualmente simple, que generalmente se almacena en un archivo de texto plano

SDK Kit de Desarrollo de Software: Del término en inglés *Software Development Kit*, el cual identifica al conjunto de bibliotecas y clases que posibilitan el desarrollo de determinada tecnología o sistema informático valga la redundancia.

SGID Sistena de Gestión Integral de Documentos. Conjunto de estructuras, funciones, procedimientos operativos y recursos asignados al control universal y material de la información producida y utilizada por los organismos, desde que se genera hasta que se deposita en los archivos históricos.[29]

Stakeholders Término inglés utilizado por primera vez por R. E. Freeman en su obra: "Strategic Management: A Stakeholder Approach", (Pitman, 1984) para referirse a «quienes pueden afectar o son afectados por las actividades de una empresa»

UCI Universidad de las Ciencias Informáticas.

UML Lenguaje Unificado de Modelado UML, del inglés, Unified Modeling Language. Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad

URI del inglés, Uniform Resource Locator. Es un localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

WWW del inglés, World Wide Web. Es un sistema de documentos de hipertexto enlazados y accesibles a través de Internet. Con un navegador Web, un usuario visualiza páginas Web que pueden contener texto, imágenes u otros contenidos multimedia.

XML del inglés, Extensible Markup Language. Lenguaje de Marcas Extensibles. Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C) que permite definir la gramática de lenguajes específicos.

Referencias bibliográficas

[1] L. J. Aguilar, "Evaluando arquitecturas de software. parte 1.," 2007.

[2] M. Castells, *LA ERA DE LA INFORMACION: ECONOMIA, SOCIEDAD Y CULTURA. LA SOCIEDAD RED*, vol. I. 1ª ed., 2005.

[3] A. Mattlelard, *La Sociedad de la Información.* 2007 de todas las ediciones en castellano ed.

[4] D. D. C. Druetta, "SOCIEDAD DE LA INFORMACIÓN y EL CONOCIMIENTO algunos deslindes imprescindibles," *Crovi Druetta, Delia (Coordinadora) 2004. Sociedad de la información y el conocimiento. Entre lo falaz y lo posible. UNAM y La Crujía Ediciones. Buenos Aires, Argentina. Pp. 17 – 56.*

[5] K. Piech, "The Knowledge-Based economy in central and east european countries – a review of some research results and policies," tech. rep.

[6] P. Kotler and G. Armstrong, *Fundamentos de Marketing.*

[7] A. I. y sistemas, "Misión de ALBET." http://www.albetsa.net.

[8] I. Company, *The Expanding Digital Universe: A Forecast of Worldwide Information Growth Through 2010.* Mar. 2007.

[9] L. J. Aguilar, "El universo digital de los datos y su impacto en sistemas informáticos." http://www.youblisher.com/files/publications/20/116379/pdf.pdf, 2011.

[10] M. M. Mena Mugica, *PROPUESTA DE REQUISITOS FUNCIONALES PARA LA GESTIÓN DE DOCUMENTOS ARCHIVÍSTICOS ELECTRÓNICOS EN LA ADMINISTRACIÓN CENTRAL DEL ESTADO CUBANO*. PhD thesis, Ciudad de La Habana, 2006.

[11] *UNE ISO 15489-I*, vol. I. 2005.

[12] "Documents and records management: An analysis of market trends to 2013 (Strategic focus)," tech. rep., Jan. 2009.

[13] *An introduction to software architecture*. 1994.

[14] R. F. T. Berners-Lee and L. M. Uniform, *Resource Identifier (URI): generic syntax. IETF RFC 3986*. Jan. 2005.

[15] U. Cei and P. Lucidi, *Alfresco 3 Web Services: Build Alfresco applications using Web Services, WebScripts, and CMIS*. 2010.

[16] *A Pattern Language: Towns, Buildings, Construction (APL)*. Oxford University Press, 1977.

[17] C. Bastarriaca, "Seminario de arquitectura de software," 2008.

[18] N. A. Jonás A. Montilva C. and J. A. Colmenares, "Desarrollo de software basado en componentes." http://webdelprofesor.ula.ve/ingenieria/jonas/Productos/Publicaciones/Congresos/CAC03

[19] F. C. Erika Camacho and G. Nuñez, "Arquitecturas de software," 2004.

[20] E. C. G. Yoan Arlet Carrascoso Puebla and A. C. Vega, "Procedimiento para la evaluación de arquitecturas de software basadas en componentes." http://www.gestiopolis.com/administracion-estrategia/archivo/procedimiento-para-la-evolucion-de-las-arquitecturas-de-software.rar, 2008.

[21] N. P. Gustavo Andrés Brey, Gastón Escobar and J. Arias, "Arquitectura de proyectos de it. evaluación de arquitecturas." http://www.gestiopolis.com/administracion-estrategia/archivo/procedimiento-para-la-evolucion-de-las-arquitecturas-de-software.rar, 2005.

[22] P. Clement, "Sei (software engeniering institute)."

[23] I. Jacobson, G. Booch, and J. Rumbaugh, *El Proceso Unificado de Desarrollo de Software*. 1999.

[24] J. Potts, *Alfresco Developer Guide*. Packt Publishing, 2009.

[25] "JavaScript - wikipedia, la enciclopedia libre." http://es.wikipedia.org/wiki/JavaScript.

[26] "CodeIgniter - open source PHP web application framework." http://es.codeigniter.com/.

[27] "jQuery: the write less, do more, JavaScript library." http://jquery.com/.

[28] "PHP: introducción - manual." http://www.php.net/manual/es/intro.curl.php.

[29] M. M. L. Martha Marina Ferriol Marchena, Olga María Pedierro Valdés and M. M. Llovet, *Manual de procedimientos para el tratamiento documental*. 2008.

Bibliografía

- "*Documents and Records Management: An Analysis of Market Trends to 2013 (Strategic Focus)*".2009, [Consultado Enero 2011].
- "*ALBET S.A. Ingeniería y sistemas. Misión de ALBET.*", {http://www.albetsa.net}.[Consultado Enero 2011].
- Bastarriaca, Cecilia. "*Seminario de Arquitectura de Software.*".2008,[Consultado Enero 2011].
- Cristian Bailey. "*El proceso de ventas de tecnologías informáticas.*"
- Fleitas Chang, Erick, Peña Yantá, Robert Alberto. "*Diseño de la arquitectura del proyecto TeleBanca.*" 2007.[Consultado Enero 2011].
- IDC Company. "*The Expanding Digital Universe: A Forecast of Worldwide Information Growth Through 2010.*" March 2007.[Consultado Enero 2011].
- Manuel Castells. "*LA Era de la Información*" 2005.[Consultado Enero 2011].
- Mena Mugica, Mayra Marta. "*Tesis doctoral*" 2006.[Consultado Enero 2011].
- Kotler, Philip and Armstrong, Gary . "*Fundamentos de Marketing.*"[Consultado Enero 2011].
- Proyecto UNE-ISO 15489. January 2005.
- Santana Pérez de Alejo, Katia, and Collazo Alfonso, Lisbey. " *Propuesta de Fábrica de Software para implementar Sistemas Integrales de Gestión de Documentos y Archivos.*" June 2009.