Universidad de las Ciencias Informáticas

Guía práctica para realizar pruebas de concepto a la arquitectura de software de sistemas de gestión.

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

Autores: Adriana Valdespino González.
Yaimairis Matos Cobas.

Tutores: Ing. Marianela Tenrero Cabrera.
Ing. René Lazo Ochoa.

Ciudad de La Habana, junio del 2009.

DEDICATORIA

De Yaimis.

A mi mamá, por guiarme durante tantos años y por creer en mí… por su preocupación, paciencia, por su cariño interminable, por estar a mi lado en los momentos difíciles, y en los momentos felices…y a mis hermanitos por hacer que con sus ocurrencias, mis vacaciones en la casa fueran bien entretenidas...

A Pedrito por el cariño y el amor que me has dado, por ayudarme y estar ahí siempre que lo necesité, por permitirme encontrar en ti un verdadero padre…

A Wilfredo, por ser tan especial todo este tiempo, por compartir conmigo tantos detallitos lindos. Solo quiero darte las gracias por el apoyo y la seguridad que me has dado. Por hacer que los días más difíciles fueran geniales, sencillamente geniales. Por tu confianza en mí, por creer en mí más que nadie… Gracias por estar conmigo este día tan importante, porque aunque no estas aquí para verme, sé que en este momento estás pensando en mí y al igual que yo, me llevas en tu corazón.

A toda mi familia, en especial a mi tía Vilma por su cariño y todo lo que hizo por mí estos años de universidad, por quererme como a una hija, sin ti no lo hubiese podido lograr…

A Diego, Yanis, Ariel y Ana Karla por haberme recibido con los brazos abiertos y haberme dado su cariño y afecto, por ayudarme en los malos momentos y compartir la alegría de los buenos…

A mis amigos, Lianet, la china, Yadira, Lissa, Daylien, por estar a mi lado en esta etapa de mi vida, por ayudarme a seguir adelante y por compartir conmigo la alegría de este día y en especial a Susana por su amistad y todo su cariño…

A todos los que de una forma u otra estuvieron a mi lado y contribuyeron a lograr ésta, una de mis más grandes metas…

De Adriana.

A mi mamá, por darme la oportunidad de existir, por su sacrificio en los tiempo difíciles, por su ejemplo de superación incasable, por su comprensión y confianza, por su amor y amistad incondicional, porque sin su apoyo no hubieran sido posible muchas cosas. Por lo que has sido y serás.

A mi tía Marisol, sé que me ves y estás orgullosa de mí.

A Manolo, como testimonio de cariño y eterno agradecimiento, por el apoyo moral y estímulos brindados con infinito amor y confianza, y por infundir en mí, ese camino que inició con toda la responsabilidad que representa el término de mi carrera profesional. Con admiración y respeto.

A mi familia y todas aquellas personas que en esta etapa de mi vida, me alentaron a lograr esta hermosa realidad. Quiero expresar un profundo agradecimiento por su ayuda, apoyo y comprensión.

A mis amigos, los que han pasado y los que han quedado, por que todos ustedes han sido parte de mi vida marcándola de alguna forma.

A demás dedico mi tesis a mis amigos especiales:

A una persona que siempre a estado a mi lado, en la que se puede confiar, que ocupa un gran espacio en mi corazón, quiero decirte que eres el mejor amigo que he tenido en la vida, quien me demostró que aunque no es fácil llegar, se necesita ahínco, lucha y deseo, pero sobre todo por el apoyo, como el que he recibido de ti durante todo este tiempo. Ahora más que nunca se confirma mi cariño, admiración y respeto. Quiero darte las gracias por estar junto a mí. Gracias por lo que hemos logrado. A ti Javier.

A Grette, por todos los momentos de diversión y de trabajo que pasamos juntas, a pesar del poco tiempo que nos conocemos, me has demostrado ser leal, por tus consejos, amiga.

A todos los amis que conocí este año, e hicieron que no fuera tan duro, por hacer que la vida sea más liviana. A ustedes: David, por ser tan genial; Virgen, por ser tan sincera; Osvaldo, por su paciencia; Aliexer, por ser fanático a mí, Noel, por tus tantas formas de sonreír y seguir mis manías.

A todos esos héroes anónimos que estuvieron a mi lado siguiendo de cerca todo este esfuerzo, ofreciendo su ayuda y facilitando el trabajo.

AGRADECIMIENTOS

A nuestra familia y en especial a nuestras MADRES, sin su apoyo esto no hubiera sido posible.

A René y a Marianela, por orientarnos y por todos sus consejos. Gracias por apoyarnos.

A nuestros amigos, por su tiempo, su cariño, ayuda e increíble paciencia.

A Susana y a Elba por habernos ayudado tanto.

A todos nuestros profesores por su aporte conjunto a nuestra formación profesional, por la confianza y dedicación.

A los que nos apoyaron de alguna forma, sería imposible mencionarlos a todos… (Aunque no físico, sí tienen todos, un espacio en cada momento difícil vivido, en cada alegría, en cada una de nuestras aspiraciones convertidas ahora en realidad).

“Sé un prototipo de calidad. Mucha gente no está acostumbrada a un ambiente donde se espera la excelencia.”

Steve Job.

ÍNDICE

RESUMEN

Cuando se trata de garantizar calidad de software en una etapa temprana, un punto de partida importante es la arquitectura de software, que es la clave para propiciar o inhibir la mayoría de los atributos de calidad esperados, por lo que es necesario evaluar las decisiones de tipo arquitectónico con respecto al impacto que ellas causarán sobre las características de calidad del sistema. Una forma eficiente de evaluar una arquitectura es mediante la implementación de una Prueba de Concepto, sin embargo, hoy en día no se cuenta con un proceso definido para desarrollar una evaluación de la misma.

En este trabajo se presenta una Guía Práctica para realizar pruebas de concepto a las Arquitecturas de Software de sistemas de gestión, que apoya el proceso de evaluación de arquitecturas de software mediante un expediente de pruebas de concepto, lo que contribuye a una mejor organización, documentación y planificación de las Pruebas de Concepto. La aplicación de esta Guía Práctica permite identificar riesgos de la arquitectura y contribuir a mitigarlos, mejorando la toma de decisiones.

Palabras Claves: Arquitectura de Software, Método de evaluación arquitectónica, Evaluación de arquitectura, Prueba de concepto, Atributo de calidad, Calidad de Software.

ÍNDICE

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La arquitectura de software se considera la columna vertebral para el éxito de cualquier software y la principal portadora de los atributos de calidad de un sistema, ésta se convierte en la pieza clave para el éxito de los proyectos de software; diseñarla mal es una fórmula para el desastre garantizado. Por tanto, para saber si una arquitectura de software es la adecuada para un sistema se debe llevar a cabo una evaluación de la misma. Una evaluación formal de la arquitectura de software debe ser una parte estándar en el desarrollo del ciclo de vida de una arquitectura. La evaluación de la arquitectura es una forma eficaz de mitigar los riesgos asociados a esta. (1)

Una Prueba de Concepto representa una evaluación de las decisiones arquitectónicas, es decir, una forma de evaluar si las decisiones tomadas en la arquitectura de un software son las correctas. Son pruebas sobre escenarios que evalúan los requisitos no funcionales y concluyen con las restricciones que la solución arquitectónica presenta en el cumplimiento de los objetivos trazados por los arquitectos. También se identifican los riesgos de la arquitectura diseñada.

El propósito de realizar estas pruebas de concepto o validación es demostrar que el producto y sus componentes cumplen con los requerimientos especificados y que el funcionamiento sea correcto en el ambiente propuesto.

En sistemas tan complejos como lo son los Sistemas de Gestión de Entidades es muy importante la selección de un método de prueba de concepto y métodos prácticos complementarios, con el objetivo de garantizar una eficiente revisión de las decisiones arquitectónicas, que posibiliten la identificación a tiempo de los riesgos y restricciones existentes en la solución arquitectónica que se propone.

Esta investigación pretende formalizar una guía para la realización de Pruebas de Concepto a la arquitectura de software de sistemas de gestión, basándose en el modelo de desarrollo que sigue el proyecto donde se realiza la investigación y los objetivos de calidad a alcanzar en la solución de software que se construye.

Se propone un esquema de actividades y artefactos que sirven de guía al equipo de dirección técnica del proyecto ERP Cuba para evaluar antes y durante el proceso de construcción la propuesta arquitectónica de la solución de software, permitiéndose obtener un instrumento que mitigue la introducción de decisiones arquitectónicas erradas, cuyos posteriores cambios podrían ser altamente costosos para el equipo de proyecto.

INTRODUCCIÓN

Es indiscutible la importancia que tiene la industria del software, por lo que es relevante caracterizar el proceso de desarrollo que se lleva a cabo en la mayoría de las organizaciones. En general, tales procesos de desarrollo se caracterizan por poner énfasis en la entrega, tomándose decisiones siempre en función de una fecha límite en detrimento de la calidad del producto final. Como una consecuencia de lo anterior, los sistemas de software son desarrollados sin considerar su posible evolución en el tiempo, su mantenimiento y extensibilidad; características que condicionaran el tiempo de vida de tales sistemas.

Aunque tales procesos de desarrollo se centran en fechas límites de entrega, son escasos los proyectos que logran cumplir con tal requerimiento. Por otra parte, muchos productos generados por las organizaciones en las que se aplican procesos de desarrollo con las características antes descritas, carecen de calidad, aspecto que se evidencia en los altos costos generados en la solución de problemas en sistemas ya entregados y en funcionamiento.

Los primeros esfuerzos realizados en evaluar arquitecturas de software utilizando un proceso estructurado y definido fueron descritos en un trabajo seminal hecho por Parnas y Weiss en 1985. En él se propone realizar revisiones de diseño activas que consisten en detectar errores e inconsistencias, por ejemplo aquellos que no fueron detectados en la fase de requerimientos. Este proceso consiste en elaborar una serie de cuestionarios, cuidadosamente escritos, de tal manera que el revisor no pueda responderlos de una manera pasiva es decir, con un SI o un NO.

Algunos de los métodos usados en la actualidad para evaluar una arquitectura de software son:

Método de Análisis de Arquitectura de Software (SAAM): Inicialmente creado para predecir facilidad de modificación.

Método de Análisis de Desventaja de la arquitectura (ATAM): Encuentra puntos sensitivos y desventajas (Trade-offs) entre atributos de calidad.

Revisión Activa de Diseños Parciales (ARID): Permite realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo.

Cuando se trata de garantizar calidad en una etapa temprana, un punto de partida importante es la arquitectura del software. Ésta permite propiciar o inhibir la mayoría de los atributos de calidad esperados. Es por ello que surgen una serie de métodos que permiten a través de la especificación de la calidad y la aplicación de técnicas variadas de evaluación, una estimación

temprana de estos atributos. Existen diversos métodos; sin embargo, cada uno de éstos instancia la evaluación arquitectónica con rasgos y características particulares que lo hacen similar o diferente de los demás, aunque el propósito sigue siendo determinar la calidad. Esto hace que la evaluación arquitectónica sea un área aún inmadura en la Ingeniería del Software.

Independientemente de la metodología implementada, la intención es obtener una arquitectura que cumpla con los servicios y la funcionalidad que espera el usuario, además de los atributos de calidad asociados que deben cumplirse, y que dirigen las decisiones en el momento de la construcción de la arquitectura del sistema.

Debido al creciente interés en asegurar la calidad en las arquitecturas de software, las organizaciones dedicadas a la construcción de sistemas han incluido en sus servicios la realización de pruebas de concepto a la arquitectura o prestar servicios a algún equipo de proyecto con este fin. Entre ellas se pueden encontrar *COMPUSOF S.A.*, *Hewlett-Packard Development Company, L.P.*, *Software y Servicios de Automatización, S.A., Pegaso Tecnología (Servicio y Soporte en Tecnología Informática S.A. de C.V.),* entre otras.

En Cuba, no se tiene un proceso definido para la realización de pruebas de concepto, en las instituciones dedicadas a la producción de software se desconocen las tecnologías que se emplean para su implementación, lo que provoca que no haya una cultura de realizar pruebas de concepto, por lo que las tomas de decisiones arquitectónicas pueden no ser las mejores para los sistemas a construir.

Hoy en día en la Universidad de Ciencias Informáticas (UCI) se desarrollan numerosos proyectos productivos en los que se construyen sistemas de gestión, uno de estos sistemas de gestión es el primer ERP cubano, actualmente el producto de software más grande que desarrolla y se ha desarrollado en Cuba.

La arquitectura de software del Sistema Integral de Gestión de Entidades, Cedrux es la base para que el sistema tenga éxito. Este sistema está motivado por un conjunto de funciones y metas de calidad para hacer factible técnica y económicamente al proyecto. La arquitectura es la clave para lograr o no estos objetivos. Si las decisiones que se tomen sobre esa arquitectura determinan las características de calidad del sistema; es necesario evaluar las decisiones de tipo arquitectónico con respecto al impacto que ellas causarán sobre las características de calidad (1).

INTRODUCCIÓN

Esta evaluación se hace necesaria no solo en el primer ERP cubano, por el impacto económico y la responsabilidad de crear un sistema que se espera sea una vía para impulsar el desarrollo económico del país, sino en todos los sistemas de gestión que se construyen. Pero desafortunadamente hoy en día no se cuenta con un proceso definido para evaluar la arquitectura, la poca experiencia de la disciplina en la universidad ha implicado la ocurrencia de costosos errores en el desarrollo y en muchos casos la afectación de los atributos de calidad del producto. Con la realización de una encuesta (Ver Anexos) en doce proyectos de la universidad se concluyó que en ninguno se realizan pruebas de concepto a la arquitectura, por lo que la definición de un procedimiento para la evaluación de la arquitectura constituye una necesidad.

Por tanto, basándose en lo anteriormente explicado, **el problema** a resolver queda formulado mediante la siguiente interrogante **¿Cómo identificar los riesgos presentes en las decisiones arquitectónicas, para evitar posibles errores arquitectónicos en un futuro y contribuir a mitigarlos?**

El **objeto de estudio** de esta investigación lo constituye la arquitectura de software. El **campo de acción** queda enmarcado en lo referente a los Métodos de evaluación para una arquitectura de software.

El **objetivo general** de esta investigación es: **Realizar una guía práctica para desarrollar pruebas de concepto a la arquitectura de software.**

Se derivan los siguientes **objetivos específicos:**

- ✓ Elaborar marco teórico.
- ✓ Definir una guía de actividades para desarrollar una prueba de concepto.
- ✓ Definir expediente de prueba de concepto.
- ✓ Desarrollar un caso de estudio.

El documento está **estructurado** en 3 capítulos:

Capítulo 1. Se especifica todo lo referente a la fundamentación teórica**,** estado del arte, situación problémica, impacto y alcance.

INTRODUCCIÓN

Capítulo 2. Se define la guía práctica para la realización de pruebas de concepto a sistemas de gestión.

Capítulo 3. Se ejemplifica el desarrollo de la guía práctica mediante la realización de pruebas de concepto al caso de estudio y se elabora un informe de resultados para la toma de decisiones.

1

CAPÍTULO

FUNDAMENTACIÓN TEÓRICA

Capítulo 1

1.1 Introducción.

En este capítulo se brinda una definición conceptual de arquitectura de software, atributos de calidad, pruebas de concepto y evaluación. Se realiza un análisis de los métodos de evaluación y las técnicas existentes. También se identifican los principales problemas que fundamentan la propuesta de solución, detallándose los objetivos generales y específicos de la misma y se describen las características de la solución.

1.2 Arquitectura de Software.

Actualmente es posible encontrar numerosas definiciones del término *Arquitectura de Software*, cada una con planteamientos diversos. Se hace evidente que su conceptualización sigue todavía en discusión, ya que no es posible referirse a un diccionario en busca de un significado, y tampoco existe un estándar que pueda ser tomado como marco de referencia.

Sin embargo, al hacer un análisis detallado de cada uno de los conceptos disponibles, resulta interesante la existencia de ideas comunes entre los mismos, sin observarse planteamientos contradictorios, sino complementarios.

La arquitectura de software puede ser vista como la estructura del sistema en función de la definición de los componentes y sus interacciones. (1)

La arquitectura de software puede considerarse como el “puente” entre los requerimientos del sistema y la implementación. (2)

La arquitectura constituye un artefacto de la actividad de diseño que servirá de medio de comunicación entre los miembros del equipo de desarrollo, los clientes y usuarios finales, dado que contempla los aspectos que interesan a cada uno. (3)

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales. (4)

Al hablar de arquitectura de software, se hace alusión a la especificación de la estructura del sistema, entendida como la organización de componentes y relaciones entre ellos; los requerimientos que debe satisfacer el sistema y las restricciones a las que está sujeto, así como las propiedades no funcionales del sistema y su impacto sobre la calidad del mismo; las

reglas y decisiones de diseño que gobiernan esta estructura y los argumentos que justifican las decisiones tomadas.

1.3 Atributos de Calidad.

Los *atributos de calidad* son aspectos del sistema, que en general, no afectan directamente a la funcionalidad necesitada, sino que definen la calidad y las características que el sistema debe soportar (1).

A grandes rasgos, se establece una clasificación de los atributos de calidad en dos categorías (1):

- Observables vía ejecución: Son aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución.
- No observables vía ejecución: Son aquellos atributos que se establecen durante el desarrollo del sistema.

La arquitectura posibilita o inhibe los atributos de calidad, algunos de estos atributos por los cuales puede ser evaluada son:

- Disponibilidad. Es la medida de disponibilidad del sistema para el uso. (5)
- Modificabilidad. Es la habilidad de realizar cambios futuros al sistema. (6)
- Rendimiento. 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. (7)
- Confiabilidad. Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo. (5)

1.4 Relación entre Arquitectura de Software y Atributos de Calidad

Para alcanzar un atributo de calidad específico, es necesario tomar decisiones de diseño arquitectónico que requieren un pequeño conocimiento de funcionalidad. Por ejemplo, el desempeño depende de los procesos del sistema y su ubicación en los procesadores, caminos de comunicación y otros. Por otro lado, se establece que al considerar una decisión de arquitectura de software, el arquitecto se pregunta cuál será el impacto de ésta sobre ciertos atributos. (1)

Por esta razón, se afirma que *cada decisión incorporada en una arquitectura de software puede afectar potencialmente los atributos de calidad.* Cada decisión tiene su origen en preguntas acerca del impacto sobre estos atributos, y el arquitecto puede argumentar cómo la decisión tomada permite alcanzar algún objetivo. Con frecuencia, el objetivo es un atributo de calidad en particular. Si las decisiones que se tomen sobre esa arquitectura determinan las características de calidad del sistema; es necesario evaluar las decisiones de tipo arquitectónico con respecto al impacto que ellas causarán sobre estos atributos de calidad. (1)

1.5 ¿Por qué evaluar una arquitectura?

La arquitectura es el primer artefacto, del ciclo de vida del desarrollo de un software, que incorpora importantes decisiones de diseño: las decisiones son fáciles de tomar, pero difíciles de cambiar una vez que el sistema se aplica.

Todos los diseños arquitectónicos implican desventajas en las cualidades del sistema, ya que estas dependen en gran medida de las decisiones arquitectónicas; por lo que garantizar la calidad del sistema es a menudo a expensas de garantizar calidad en la arquitectura y esto es posible si se realiza una evaluación de la misma.

Una evaluación es un estudio de factibilidad que pretende detectar posibles riesgos y buscar recomendaciones para contenerlos. El objetivo de evaluar una arquitectura es saber si esta puede habilitar los requerimientos, atributos de calidad y restricciones para asegurar que el sistema a ser construido cumple con las necesidades de los clientes. La evaluación de una arquitectura no produce resultados cuantitativos, ayuda a encontrar debilidades y garantiza la identificación de riesgos y no riesgos asociados a la misma.

Una mala arquitectura puede llevar a un proyecto al fracaso. Todos los requerimientos de calidad pueden quedar insatisfechos. La arquitectura también determina la estructura del proyecto: configuración, agenda y presupuesto, alcance, entre otros aspectos. Es mejor cambiar la arquitectura antes que otros artefactos, que están basados en ella, se estabilicen. Realizar una evaluación de la arquitectura es la manera más económica de evitar desastres. (8)

1.6 ¿Por qué los atributos de calidad son demasiados imprecisos para el análisis?

Los atributos de calidad son la base para la evaluación de una arquitectura, pero por si solos no son suficientes para juzgar la adecuabilidad de la arquitectura. Generalmente, los

requerimientos son escritos de la siguiente manera (3): “El sistema debe ser robusto.” “El sistema debe ser modificable.” “El sistema debe ser seguro.” “El sistema debe tener un rendimiento aceptable.” Cada una de las frases anteriores está sujeta a diferentes interpretaciones y malos entendidos. Lo que uno puede considerar robusto, otro puede considerarlo apenas aceptable.

El punto es, que los atributos de calidad no son cantidades absolutas, existen en un contexto con metas específicas. En particular, un sistema es modificable (o no) con respecto a un tipo de cambio específico. Un sistema es seguro (o no) con respecto a un tipo de amenaza específica. Un sistema es confiable (o no) con respecto a la ocurrencia de un tipo de falta específica. Un sistema tiene buen rendimiento (o no) con respecto a un criterio específico de rendimiento. Una arquitectura es construible (o no) con respecto a restricciones de tiempo y presupuesto específicas.

No parece razonable, considerar que un sistema pueda alguna vez, por ejemplo, ser completamente confiable bajo toda circunstancia. Dado esto, es importante que el arquitecto entienda perfectamente bajo que circunstancias un sistema debe ser confiable para ser considerado aceptable. Por lo tanto, el primer trabajo que debe realizar una evaluación de arquitectura es obtener las metas específicas de calidad ante las cuales la arquitectura será juzgada. Si algunas de estas metas no son específicas o son ambiguas, se debe pedir a los *Involucrados* que ayuden al equipo de evaluación a reescribirlas. El mecanismo a utilizar para representar estas metas es el de escenario.

1.7 ¿Cuándo evaluar una arquitectura?

Habitualmente, la evaluación de la arquitectura ocurre después que ésta ha sido especificada, pero antes que empiece la implementación, esto se hace con el objetivo de validar el diseño propuesto. No obstante, uno de los aspectos más interesantes de la evaluación de arquitecturas es que se puede efectuar en cualquier etapa de la vida de una arquitectura. Básicamente existen dos etapas: temprana y tardía. En la primera no tiene por qué estar especificada completamente la arquitectura, en la mayoría de los casos es utilizada para examinar las decisiones arquitectónicas ya tomadas y decidir entre las opciones que están pendientes. Por otro lado la evaluación tardía es realizada tanto cuando la arquitectura está terminada como cuando la implementación está completa. Este es el caso general que se presenta en el momento de la adquisición de un sistema ya desarrollado. Se considera muy útil

la evaluación del sistema en este punto, porque puede observarse el cumplimiento de los atributos de calidad asociados al sistema, y cómo será su comportamiento general.

En general, una evaluación debe realizarse cuando hay suficiente de la arquitectura como para justificarla. Una buena regla sería: realizar una evaluación cuando el equipo de desarrollo empieza a tomar decisiones que dependen de la arquitectura y el costo de deshacerlas sobrepasa al costo de realizar una evaluación. (8)

1.8 Pruebas de concepto.

Una *prueba de concepto*, o *evaluación de la arquitectura* de un sistema de software, es el análisis de la estructura para identificar los riesgos potenciales y verificar que los requisitos de calidad se han tenido en cuenta en el diseño. (9) En otras palabras es la evaluación a las decisiones arquitectónicas.

El propósito de realizar estas pruebas de concepto o validación es demostrar que el producto y sus componentes cumplen con los requerimientos especificados y que el funcionamiento sea correcto en el ambiente propuesto. (9)

Existen varios tipos de Pruebas de Concepto (PoC), según lo que se quiera probar. Por ejemplo, si lo que se pretende es testear el rendimiento de la aplicación, o su escalabilidad, la prueba se limita a estresar un entorno de servidores similar al que se planee para Producción, con el objetivo de conocer los límites o puntos de quiebra de la aplicación (cuántos usuarios concurrentes, cuántos requerimientos simultáneos será capaz de atender con un tiempo de respuesta no mayor a los 200 ms). Otros tipos de PoC pueden testear la seguridad de la aplicación simulando ataques y observando cómo reacciona el sistema; y otros. (10)

La Prueba de Concepto es una etapa necesaria para validar la arquitectura, para decidir si seguir adelante o someterla a ajustes o revisiones posteriores. Pero fundamentalmente, es un elemento que al arquitecto le debe proveer confianza en sus decisiones, y a la vez inspirar confianza en el resto de los interesados en la aplicación. (9)

Por tanto las Finalidades de las pruebas de concepto son:

- Confirmar la elección del concepto en la solución propuesta.
- Abordar ideas de mejora.

Y como resultados se pueden obtener:

- Viabilidad del despliegue de la solución en un ambiente similar al requerido.
- Validación de las funcionalidades de la solución.
- Adecuar y afinar los elementos de software y hardware requeridos.
- Validación de las herramientas de almacenamiento.
- Determinar las posibles vulnerabilidades que puedan existir y definir la vía de mitigación de dichas vulnerabilidades y riesgos.

La evaluación de la arquitectura de software puede ser realizada mediante el uso de diversas técnicas y métodos. En este sentido, resulta interesante estudiar las distintas opciones que existen en la actualidad para llevar a cabo esta tarea.

1.9 Técnicas de evaluación.

La evaluación de una arquitectura de software es una tarea no trivial, el objetivo es **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. Los objetivos son *cualitativos* y *cuantitativos* (6).

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.

Por lo regular, las técnicas de evaluación cualitativas son usadas cuando la arquitectura se encuentra en construcción, mientras que las técnicas de evaluación cuantitativas, se usan cuando la arquitectura ya ha sido implantada. (6)

En el tipo de medición cualitativa se pueden utilizar las técnicas basadas en escenarios o en experiencia. Mientras que en el tipo de medición cuantitativas se pueden emplear simulaciones o modelos matemáticos. (6)

1.9.1 Evaluación basada en escenarios.

La evaluación *basada en escenario* es una breve descripción de la interacción de alguno de los involucrados en el desarrollo del sistema con éste. Por ejemplo, un usuario hará la descripción en términos de la ejecución de una tarea; un encargado de mantenimiento hará referencia a cambios que deban realizarse sobre el sistema; un desarrollador se enfocará en el uso de la arquitectura para efectos de su construcción o predicción de su desempeño. Un escenario consta de tres partes: el estímulo, el ambiente y la respuesta.

Estímulo: Es la parte del escenario que explica o describe lo que el involucrado en el desarrollo hace para iniciar la interacción con el sistema. Puede incluir ejecución de tareas, cambios en el sistema, ejecución de pruebas, configuración y otros.

Ambiente: Describe las condiciones en la cual se encuentra el sistema en el momento que se recibe el estímulo.

Respuesta: Describe, a través de la arquitectura, cómo debería responder el sistema ante el estímulo. Permite establecer cuál es el atributo de calidad asociado.

Los escenarios proveen un vehículo que permite concretar y entender atributos de calidad.

Entre las ventajas de su uso están:

- Son simples de crear y entender.
- Son poco costosos y no requieren mucho entrenamiento.
- Son efectivos.

Actualmente las técnicas basadas en escenarios cuentan con dos instrumentos de evaluación relevantes: el Árbol de utilidades propuesto por Kazman y los Perfiles, propuestos por Bosch (6).

Un *Árbol de Utilidades* es un esquema en forma de árbol que presenta los atributos de calidad de un sistema de software, refinados hasta el establecimiento de escenarios que especifican con suficiente detalle el nivel de prioridad de cada uno.

La intención del uso del Árbol de Utilidades es la identificación de los atributos de calidad más importantes para un proyecto particular. No existe un conjunto preestablecido de atributos, sino que son definidos por los involucrados en el desarrollo del sistema en el momento de la construcción del árbol.

El Árbol de Utilidades contiene como nodo raíz la *utilidad general* del sistema. Los atributos de calidad asociados al mismo conforman el segundo nivel del árbol (3), los cuales se refinan hasta la obtención de un escenario lo suficientemente concreto para ser analizado y otorgarle prioridad a cada atributo considerado.

Cada atributo de calidad perteneciente al árbol contiene una serie de escenarios relacionados, y una escala de importancia y dificultad asociada, que será útil para efectos de la evaluación de la arquitectura.

Un *perfil* es un conjunto de escenarios, generalmente con alguna importancia relativa asociada a cada uno de ellos. El uso de perfiles permite hacer especificaciones más precisas del requerimiento para un atributo de calidad. Los perfiles tienen asociados dos formas de especificación: perfiles completos y perfiles seleccionados.

Los *perfiles completos* definen todos los escenarios relevantes como parte del perfil. Esto permite al ingeniero de software realizar un análisis de la arquitectura para el atributo de calidad estudiado de una manera completa, ya que incluye todos los posibles casos. Su uso se reduce a sistemas relativamente pequeños y sólo es posible predecir conjuntos de escenarios completos para algunos atributos de calidad.

Los *perfiles seleccionados* se asemejan a la selección de muestras sobre una población en los experimentos estadísticos. Se toma un conjunto de escenarios de forma aleatoria, de acuerdo a algunos requerimientos. La aleatoriedad no es totalmente cierta por limitaciones prácticas, por lo que se fuerza la realización de una selección estructurada de elementos para el conjunto de muestra. Si bien es informal, permite hacer proposiciones científicamente válidas.

1.9.2 Evaluación basada en simulación.

La evaluación basada en simulación se utiliza con una implementación de alto nivel de la arquitectura de software. El enfoque básico consiste en la implementación de componentes de la arquitectura y la implementación, a cierto nivel de abstracción, del contexto del sistema donde se supone va a ejecutarse. La finalidad es evaluar el comportamiento de la arquitectura bajo diversas circunstancias. Una vez disponibles estas implementaciones, pueden usarse los perfiles respectivos para evaluar los atributos de calidad.

1.9.3 Evaluación basada en modelos matemáticos.

La evaluación basada en modelos matemáticos se utiliza para evaluar atributos de calidad operacionales. Permite una evaluación estática de los modelos de diseño arquitectónico, y se presentan como alternativa a la simulación, ya que evalúan el mismo tipo de atributos. Ambos enfoques pueden ser combinados, utilizando los resultados de uno como entrada para el otro.

Entre las desventajas que presenta esta técnica se encuentra la inexistencia de modelos matemáticos apropiados para los atributos de calidad relevantes, y que el desarrollo de un modelo de simulación completo puede requerir esfuerzos sustanciales.

Entre los instrumentos que se cuentan para las técnicas de evaluación de arquitecturas de software basada en modelos matemáticos, se encuentran las Cadenas de Markov y los Diagramas de bloque de fiabilidad.

1.9.4 Evaluación basada en experiencia.

En muchas ocasiones los arquitectos e ingenieros de software otorgan valiosas ideas que resultan de utilidad para la evasión de decisiones erradas de diseño. Aunque todas estas experiencias se basan en evidencia anecdótica; es decir, basada en factores subjetivos como la intuición y la experiencia. Sin embargo, la mayoría de ellas puede ser justificada por una línea lógica de razonamiento, y pueden ser la base de otros enfoques de evaluación.

Existen dos tipos de evaluación basada en experiencia: la *evaluación informal*, que es realizada por los arquitectos de software durante el proceso de diseño, y la *evaluación formal* realizada por equipos externos de evaluación de arquitecturas.

1.10 Métodos de evaluación

Kazman propone que la existencia de un método de análisis de arquitecturas de software hace que el proceso sea repetible y ayuda a garantizar que las respuestas correctas con relación a la

arquitectura pueden hacerse temprano, durante las fases tempranas de diseño. Es en este punto donde los problemas encontrados pueden ser solucionados de una forma relativamente poco costosa. De manera similar, un método de evaluación sirve de guía a los involucrados en el desarrollo del sistema, en la búsqueda de conflictos que puede presentar una arquitectura y sus soluciones. Por esta razón, resulta conveniente estudiar los métodos de evaluación de arquitecturas de software propuestos hasta el momento. La mayoría de estos métodos utilizan escenarios, su enfoque se orienta hacia la mitigación de riesgos, ubicando dónde un atributo de calidad de interés se ve afectado por decisiones arquitectónicas. En su estudio, Kazman presenta tres métodos de evaluación de arquitecturas de software, que son el Método de Análisis de Desventajas de la Arquitectura (Architecture Trade-off Analysis Method, ATAM), Método de análisis de arquitecturas de software (Software Architecture Analysis Method, SAAM) y Revisión de Diseño Activo Intermedio (Active Intermediate Designs Review, ARID).

1.10.1 Método de análisis de arquitecturas de software

El Método de Análisis de Arquitecturas de Software (SAAM) es el primero que fue ampliamente promulgado y documentado. El método fue originalmente creado para el análisis de la modificabilidad de una arquitectura, pero en la práctica ha demostrado ser muy útil para evaluar de forma rápida distintos atributos de calidad, tales como modificabilidad, portabilidad, escalabilidad e integrabilidad. El método de evaluación SAAM se enfoca en la enumeración de un conjunto de escenarios que representan los cambios probables a los que estará sometido el sistema en el futuro. Como entrada principal, es necesaria alguna forma de descripción de la arquitectura a ser evaluada. Las salidas de la evaluación del método SAAM son las siguientes:

- Una proyección sobre la arquitectura de los escenarios que representan los cambios posibles ante los que puede estar expuesto el sistema.
- Entendimiento de la funcionalidad del sistema, e incluso una comparación de múltiples arquitecturas con respecto al nivel de funcionalidad que cada una soporta sin modificación.

Con la aplicación de este método, si el objetivo de la evaluación es una sola arquitectura, se obtienen los lugares en los que la misma puede fallar, en términos de los requerimientos de modificabilidad. Para el caso en el que se cuenta con varias arquitecturas candidatas, el método produce una escala relativa que permite observar qué opción satisface mejor los requerimientos de calidad con la menor cantidad de modificaciones.

1.10.2 Revisión Activa de Diseños Parciales

El método de Revisión Activa de Diseños Parciales (ARID) es conveniente para realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo. En ocasiones, es necesario saber si un diseño propuesto es conveniente, desde el punto de vista de otras partes de la arquitectura.

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.

ARID es un híbrido entre Revisión Activa del Diseño (ADR) y ATAM. Tanto ADR como ATAM proveen características útiles para el problema de la evaluación de diseños preliminares. En el caso de ADR, los involucrados reciben documentación detallada y completan cuestionarios, cada uno por separado. En el caso de ATAM, está orientado a la evaluación de toda una arquitectura.

Ante esta situación, y la necesidad de evaluación en las fases tempranas del diseño, Kazman proponen la utilización de las características que proveen tanto ADR como ATAM por separado. De ADR, resulta conveniente la fidelidad de las respuestas que se obtiene de los involucrados en el desarrollo. Así mismo, la idea del uso de escenarios generados por los involucrados con el sistema es tomada del ATAM. De la combinación de ambas filosofías surge ARID, para efecto de la evaluación temprana de los diseños de una arquitectura de software.

1.10.3 Método de Análisis de Desventajas de Arquitectura (ATAM)

El Método de Análisis de Desventajas de Arquitectura (ATAM) está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM. El nombre del método ATAM surge a raíz 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. El término enfoque arquitectónico se debe a que no todos los arquitectos están familiarizados con el lenguaje de estilos arquitectónicos, aún haciendo uso indirecto de estos. De cualquier forma, 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, integrarse con otros sistemas, entre otros (3).

EL Instituto de Ingenieros de Software (SEI) ha desarrollado ATAM a lo largo de varios años. El propósito de ATAM es evaluar las consecuencias de las decisiones arquitectónicas en base en los atributos de calidad (9).

ATAM es un método en el que los principales puntos de análisis son:

- Descubrir riesgos: las alternativas que podrían crear problemas en el futuro en algunos atributos de calidad.
- Descubrir no riesgos: las decisiones que promueven cualidades que ayudan a cumplir los objetivos del negocio.
- Descubrir puntos de sensibilidad: alternativas donde un ligero cambio hace una diferencia significativa en algunos atributos de calidad.
- Descubrir desventajas: las decisiones que afectan a más de un atributo de calidad.

Realizar este análisis puede ser el objeto de las actividades de mitigación: por ejemplo, perfeccionar su diseño, un análisis más detallado, prototipo. Las desventajas pueden ser explícitamente identificadas y documentadas.

- Realizar el análisis de ATAM traerá consigo una serie de beneficios, entre ellos:
- Aclara cuales son los atributos de calidad más importantes.
- Mejorar la documentación de la arquitectura.
- Documentar las decisiones de la arquitectura base.
- Identificar riesgos en una fase temprana del ciclo de vida.
- Aumentar la comunicación entre los interesados.

El propósito de ATAM NO es prever en el análisis, el objetivo, es descubrir los riesgos creados por las decisiones arquitectónicas. Su finalidad es evaluar las consecuencias de las decisiones arquitectónicas a través de los atributos de calidad. Este método es un proceso corto, que

facilita la interacción entre las múltiples partes interesadas, lo que lleva a la identificación de riesgos, las sensibilidades, y desventajas y no funcionará si la arquitectura de Software no ha sido creada aún por lo que debe existir una arquitectura con un alcance definido y manejable, entre las precondiciones para aplicarlo se encuentran:

- Los miembros del equipo de revisión analizarán los artefactos arquitectónicos y pueden ayudar a mejorar la documentación. El arquitecto debe preparar una arquitectura de presentación.
- Los clientes deben preparar una presentación de los objetivos del negocio.
- El equipo de evaluación revisará los artefactos de la arquitectura, presentaciones y leerá el material antes para familiarizarse con el dominio.

ATAM esta estructurado en 4 fases y 9 pasos:

Presentación:

1- *Presentar ATAM.* El método se describe a las partes interesadas (normalmente los representantes de los clientes, el arquitecto o el equipo de arquitectura, representantes de los usuarios, mantenedores, administradores, gerentes, probadores, integradores y otros).

2- *Presentar los objetivos del negocio.* El líder del proyecto describe los objetivos que motivan el esfuerzo de desarrollo y, por tanto, lo que será la arquitectura primaria (por ejemplo, alta disponibilidad, tiempo de salida al mercado o alta seguridad, requerimientos funcionales de alto nivel y requerimientos de atributos de calidad).

3- *Presentar la arquitectura.* El arquitecto presenta un panorama general de la arquitectura. Pueden ser las restricciones técnicas, tales como el sistema operativo, hardware, software o medios previstos para su uso; otros sistemas con los que el sistema debe interactuar; estilo arquitectónico utilizado para hacer frente a los requisitos de los atributos de calidad. Equipo de evaluación comienza el análisis y la captura de los riesgos.

Investigación y Análisis:

4- *Identificar los enfoques arquitectónicos.* Comienzan a identificar los enfoques arquitectónicos que son fundamentales para la realización de los atributos de calidad objetivos.

5- *Generar atributos de calidad y el árbol de utilidad.* Identificar, priorizar y refinar los atributos de calidad más importantes, cuyo objetivo es la construcción de un árbol de utilidad.

- Un árbol de utilidad es un vehículo para el manejo de los requisitos de atributos de calidad específicos.
- Selecciona los objetivos de calidad para ser los nodos el alto nivel (por lo general, el rendimiento, modificabilidad, seguridad y disponibilidad)
- Los escenarios son las hojas del árbol de utilidad, anotado con los estímulos y las respuestas, y por orden de prioridad; estos deben ser lo más específicos posible.

La salida de un árbol de utilidad es la caracterización y priorización de los requisitos de atributos de calidad específicos.

La importancia para el éxito del sistema es Alta / Media / Baja; la dificultad para alcanzar el objetivo es Alta / Media / Baja, a evaluación del arquitecto.

6- *Analizar los enfoques arquitectónicos.* El equipo de evaluación identifica los enfoques arquitectónicos desde el punto de vista de los atributos de calidad específicos para identificar los riesgos. Generar preguntas para los atributos de calidad de mayor prioridad (por ejemplo, un enfoque arquitectónico destinado a satisfacer objetivos de rendimiento será sometido a un análisis del rendimiento). Identificar “riesgos”, “no riesgos”, “puntos sensibles” y “desventajas”.

Pruebas

7- *Lluvia de ideas y dar prioridad a los escenarios.* Los interesados generan los escenarios apoyándose para facilitar el trabajo, en la realización de una lluvia de ideas obteniendo un conjunto más amplio de los escenarios. A este conjunto de escenarios se le da prioridad a través de un proceso de votación de la totalidad de los participantes del grupo de interesados. Se agregan los nuevos escenarios al árbol de utilidad.

8- *Analizar los enfoques arquitectónicos.* Identificar los enfoques de arquitectura impactados por los escenarios generados en el paso anterior. Este paso continua el análisis iniciado en el paso 6, usando los escenarios nuevos, y que luego se documentan.

Presentación de informes

9- *Presentar los resultados.* Sobre la base de la información recogida en ATAM (estilos, escenarios, preguntas de atributos específicos, el árbol de utilidad, los riesgos, los puntos de sensibilidad, intercambios), El equipo evaluador presenta las conclusiones a las partes interesadas que se reunieron y potencialmente escribe un informe detallando de esta información junto con cualquier propuesta de las estrategias de mitigación.

Académicamente, el momento de uso de ATAM es justo después que la arquitectura se ha especificado, cuando hay poco o ningún código. Sin embargo, en la práctica, ATAM ha sido muy efectivo en las siguientes situaciones (3):

- Evaluación de arquitecturas alternativo candidatas.
- Evaluación de los sistemas existentes antes de comprometerse a la mejora.
- Decidir entre actualizar o sustituir.

1.11 Pruebas de Concepto en el mundo empresarial.

Desde hace ya varios años, se ha dado mayor importancia a las Tecnologías de la Información y su alineación con las estrategias de negocio para mejorar los procesos claves del mismo. Desarrollar un software con la calidad que requiere, por lo general es un proceso largo y complejo, ya que implica rediseñar los esquemas de trabajo. Su implementación es de alto riesgo, porque envuelven complejidad, tamaño, altos costos, un equipo considerable de desarrollo, además de inversión de tiempo.

Para alcanzar la mayoría de los atributos de calidad esperados, evaluar la arquitectura de software más que una necesidad constituye un reto. La intención es obtener una arquitectura robusta, que cumpla con los servicios y la funcionalidad que espera el cliente, además de los atributos de calidad asociados que deben cumplirse. La necesidad de asegurar la calidad en las arquitecturas de software, ha provocado que varias de las organizaciones dedicadas a la

producción de software hayan incluido en sus servicios la realización de pruebas de concepto a la arquitectura o prestar servicios a algún equipo de proyecto con este fin.

Entre las empresas a nivel mundial se pueden encontrar: *Compusof S.A.* (11)*:* Provee productos, servicios y soluciones orientadas al negocio a las empresas que necesitan incorporar una tecnología adecuada a su sistema de trabajo. Al estar constituida sobre la base de la oferta integral de tecnologías de información, suministra de manera fiable hardware, software, servicios y soluciones avanzadas. Son integradores de los fabricantes líderes en el mercado tanto en hardware como en software y uno de los principales socios de Hewlett-Packard en España. Para la realización de pruebas de concepto definen las pruebas a realizar, implementan el entorno de prueba, realizan la instalación y prueba de aplicaciones, configuran Sistemas Operativos y aplicaciones y como resultado se obtiene la documentación de instalación, la configuración del Sistema Operativo y aplicaciones y por ultimo las recomendaciones adicionales.

Hewlett-Packard Development Company, L.P (12): Ofrece servicios para aplicaciones empresariales como la solución “Empresa con latencia cero” (ZLE); usando métodos ampliamente comprobados. Dentro de los servicios ZLE, HP incluye las Prueba de concepto; que le permite construir un modelo funcional lo suficientemente detallado para determinar de manera específica el valor comercial y técnico de la solución ZLE para la empresa. La prueba de concepto se basa en un análisis completo de su ambiente actual y en una consideración a fondo de sus requisitos identificados a futuro, incorporando los hallazgos y resultados de servicios anteriores, como los servicios de Exploración del Ambiente Empresarial, Exploración Arquitectónica y Planeación Detallada de la Arquitectura ZLE o su equivalente, así como de cualquier otro material disponible.

Instituto de Ingeniería de Software (SEI) (9) : Es financiado por el gobierno federal de Estados Unidos, es un centro de desarrollo e investigación de ingeniería de software, arquitectura y líneas de productos, mejora de procesos y medición del desempeño, la seguridad, interoperabilidad de los sistemas y la fiabilidad. El SEI ha desarrollado los siguientes métodos para la evaluación de arquitecturas de sistema y software:

Método de Análisis de desventajas de la arquitectura (ATAM), utilizado para revelar cómo la arquitectura satisface un atributo de calidad particular, los riesgos, las sensibilidades, y desventajas involucradas en la satisfacción de las necesidades.

Método de Análisis de Costos-Beneficios (CBAM), guía a los ingenieros de sistemas y otros interesados para hacer intercambios económicos y de servicios públicos asociados a las decisiones arquitectónicas.

Revisiones Activas de Diseños Intermedios (ARID), utilizados para evaluar diseños tempranos o partes del diseño para su viabilidad en la satisfacción de los interesados.

Estas técnicas pueden usarse solas o en combinación para obtener los beneficios de cualquier proyecto de desarrollo de software.

En Cuba, las instituciones dedicadas a la producción de software no tienen un proceso definido para la realización de pruebas de concepto, se desconocen las tecnologías que se emplean para su implementación, lo que provoca que no haya una cultura de realizar pruebas de concepto, por lo que las tomas de decisiones arquitectónicas pueden no ser las mejores para los sistemas a construir.

Hoy en día en la Universidad de Ciencias Informáticas no se cuenta con un proceso definido para evaluar la arquitectura, la poca experiencia de la disciplina en la Universidad ha implicado la ocurrencia de costosos errores en el desarrollo y en muchos casos la afectación de los atributos de calidad del producto.

1.12 ¿Qué impacto tienen las Pruebas de Concepto?

La realización de las pruebas de concepto a la arquitectura de software tiene gran impacto, ya que se identifican riesgos en las decisiones arquitectónicas, contribuyendo así a colaborar con la mejora de las decisiones, mitigar los riesgos y evitar posibles errores arquitectónicos en un futuro y por tanto el rediseño de la arquitectura y el desgaste de recursos.

El impacto económico es significativo, los errores en los diseños arquitectónicos son una de las causas que traen consigo la entrega tardía de los productos y el fracaso en tiempo de ejecución de estos, trayendo como consecuencia el desgaste de recursos innecesarios, sin embargo cuando se realiza una prueba de concepto a una arquitectura de software siendo planificada y documentada debidamente, son mitigados estos riesgos, reduciendo los altos costos en el mantenimiento de la aplicación.

Socialmente constituye un material avalado que puede ser utilizado para el estudio e implementación de una prueba de concepto, lo cual es algo novedoso porque por primera vez

existe una Guía Práctica que incluye un expediente para la realización de pruebas de concepto de manera que integre todos los aspectos de interés en las arquitecturas de sistemas de gestión.

Esta solución tiene un impacto novedoso ya que al aplicarla para la realización de pruebas de concepto ha traído como resultado que en tres meses de implantado, el *Sistema Integral de Gestión de Entidades, Cedrux*, no han surgido no conformidades en el *Framework*.

1.13 Conclusiones.

El proceso de desarrollo de software es complejo y requiere de un gran esfuerzo, preparación, tiempo para realizarse y dedicación por parte del equipo de trabajo, pero no siempre se cuenta con todos estos factores para el buen desarrollo y empleo de las técnicas correctas para construir un software. *El Sistema Integral de Gestión de Entidades, Cedrux* es uno de los proyectos que hoy en día se desarrollan en la Universidad de las Ciencias Informáticas y que el país necesita sea no solo eficiente y eficaz, sino que se desarrolle en el menor tiempo posible debido a la importancia y el impacto que tiene para el desarrollo económico del país.

Es por ello que el desarrollo de la arquitectura de software de este sistema, se basa en construir la arquitectura prácticamente al mismo tiempo del desarrollo de la aplicación, esto trae consigo que la arquitectura ha habido que diseñarla y desarrollarla a la vez, por lo que se deben cometer la menor cantidad posible de errores y asegurar que las decisiones arquitectónicas no sean un riesgo para el sistema, por esta razón la realización de las pruebas de concepto a la arquitectura se hace ineludible, para garantizar calidad en el producto que se construye.

Pero como no se cuenta con un proceso definido para evaluar la arquitectura se decide desarrollar una Guía Práctica que dirija la realización de las pruebas de concepto.

En el transcurso del procedimiento de evaluación de la arquitectura, se utilizan técnicas y herramientas en su desarrollo; cada uno de los miembros del equipo de evaluación juega un rol específico dentro de él y se generan artefactos, entre los que se incluye un expediente que contribuye a organizar las pruebas de concepto realizadas a la arquitectura.

2

CAPÍTULO

GUÍA PRÁCTICA PARA REALIZAR PRUEBAS DE CONCEPTO.

Capítulo 2

2.1 Introducción.

En el presente capítulo se definen los roles que participan en la evaluación de la arquitectura y un conjunto de actividades para la realización de la misma, que incluye un expediente que contribuye a organizar las pruebas de concepto realizadas a la arquitectura, además se realiza un desglose de cada uno de los aspectos que forman la estructura del expediente y se describen cada uno de ellos.

2.2 Roles.

Durante el proceso de desarrollo de una prueba de concepto cada uno de los miembros del equipo de evaluación juega un rol específico dentro de él, los roles que se definen para ejecutar las actividades que se proponen en la Guía Práctica son:

Grupo de evaluación de la arquitectura:

- Grupo entre 3-5 personas: Pertenecientes a la misma organización del grupo de desarrolladores o consultores.

Tomadores de decisión del proyecto: responsables del proyecto en capacidad de solicitar cambios.

Incluyen:

- Gerente de proyecto.
- Usuarios del sistema.
- Arquitecto de la solución.

Involucrados en la arquitectura: articulación de las metas de los atributos de calidad en el sistema para que sea exitoso.

Incluyen:

- Ingenieros desarrolladores.
- Integradores.
- Grupo de pruebas.

Capítulo 2

2.3 Guía Práctica de actividades para el desarrollo de una prueba de concepto.

Este material en una Guía Práctica que incluye un expediente para la realización de pruebas de concepto que integra aspectos de interés de las arquitecturas de sistemas de gestión. Puede ser utilizado para el estudio e implementación de una prueba de concepto.

Esta Guía Práctica consta de diez actividades y tres roles, donde cada rol tiene una responsabilidad determinada durante la ejecución de cada una de las actividades de la guía. Es iterativa y contiene elementos de retroceso. Puede ser aplicada a cualquier sistema de gestión y durante todo el proceso de construcción de la arquitectura, desde el diseño, durante su implementación y una vez que ha sido construida.

Cuando se realiza una prueba de concepto a una arquitectura de software siendo planificada y documentada debidamente, son mitigados los riesgos en el mantenimiento de la aplicación reduciendo los altos costos.

Actividad 1 - Presentar el negocio.

Esta presentación es importante para entender el contexto del sistema y los conductores primarios del negocio. Los tomadores de decisiones bridan una panorámica de las perspectivas del mismo, por lo que se describen las metas del negocio, las funciones más importantes del sistema, cualquier técnica relevante, económica o contratos contraídos y conductores arquitectónicos (principales objetivos de los atributos de calidad que forman la arquitectura).

Actividad 2 - Presentar la arquitectura.

El arquitecto describe la arquitectura del sistema, mencionando las técnicas contraídas, como hardware, sistema operativo, otros sistemas con los que se debe interactuar y en que nivel de desarrollo se encuentra la arquitectura. Se brinda información arquitectural relevante como: Diagrama de contexto, contexto en el cual existe el sistema, interacción con actores/personas-sistemas; Vista por módulos o capas, vista de componentes y conectores, vista de despliegue. Aproximaciones arquitecturales (patrones y RNF que atacan) donde se describen los enfoques arquitectónicos (o patrones) usados para llegar a los requerimientos.

Actividad 3 - Seleccionar elementos a evaluar.

En esta actividad participan en conjunto los evaluadores y los desarrolladores. Tiene como objetivo seleccionar los elementos a los que se van a evaluar, que pueden ser los relacionados con las decisiones arquitectónicas del Framework, de datos, de cliente, entre otras; definiendo los objetivos de la evaluación y los escenarios para cada una de las pruebas.

Actividad 4 - Diseñar las pruebas de concepto.

En esta actividad el Grupo de evaluación teniendo en cuenta la etapa de la arquitectura en la que se va a realizar la evaluación, y definidos ya los distintos objetivos de las mediciones, los cuales pueden ser cualitativos o cuantitativos; selecciona las técnicas de evaluación que se van a emplear y los cronogramas para su implementación. También es muy importante en esta actividad la selección de la tecnología para su realización debido a que el empleo de herramientas de soporte en las pruebas de concepto permite determinar con mayor precisión y en menor tiempo muchos de los riesgos asociados a la arquitectura.

Actividad 5 - Evaluar la viabilidad de la prueba de concepto de la Arquitectura.

En esta actividad tiene como objetivo que el Grupo de evaluación determine si con el diseño de pruebas de concepto, los elementos a evaluar y herramientas que se han definido para evaluar la arquitectura, es posible realizar las pruebas. En caso de que no sea factible realizar la prueba con los elementos que se tienen, es necesario regresar a la *actividad 3.*

Actividad 6 - Determinar el método de evaluación.

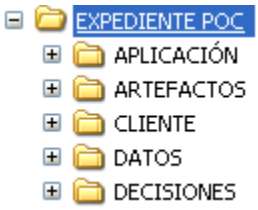
Esta actividad tiene como objetivo seleccionar y presentar el método de evaluación que se va utilizar para la realización de las pruebas de concepto por parte del Grupo de evaluación, teniendo en cuenta los atributos de calidad que se desean evaluar y los objetivos de la evaluación, también se refinan los escenarios a ser evaluados. Un método de evaluación no es mejor que otro, sino que evalúa mejor, en ciertas condiciones, un atributo de calidad dado, por lo que se concluye que en dependencia de las condiciones y lo que se desea evaluar, será el método de evaluación empleado.

Actividad 7 - Estructurar el expediente para pruebas de concepto.

Esta actividad tiene como objetivo estructurar el expediente que se define para la realización de pruebas de concepto de acuerdo al interés de los arquitectos con respecto a las disciplinas de la arquitectura de software.

Capítulo 2

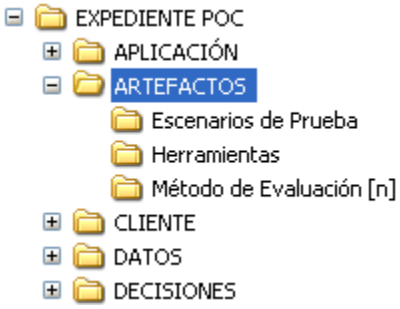
El *expediente de pruebas de concepto* tiene como objetivo organizar y documentar las pruebas de concepto realizadas a la arquitectura. Se estructuró de manera que refleja un corte vertical en el sentido de las dimensiones de interés para los arquitectos, estos necesitan desde el punto de vista de las pruebas, una guía para dimensionar qué partes se van aprobar desde el punto de vista de la intención. Se necesita conocer cómo se comporta el sistema desde la dimensión de Cliente, Datos, Aplicación, Decisiones y se genera un conjunto de Artefactos.



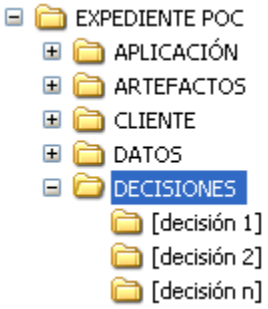
Las pruebas se deben realizar con varias muestras cada una, para determinar un promedio de los resultados ya que entre una y otra puede existir una variación de los mismos, especificándose en fotos, Excel o ficheros generados los resultados de las muestras tomadas.

Los resultados de las pruebas se especifican en un documento general que recoge un resumen de todas las pruebas realizadas a los paquetes que contiene el expediente, ya sea cliente, aplicación, dato, artefactos o decisiones (Ver Anexos).

El paquete *Artefactos* contiene la especificación y resultados de la aplicación del método de evaluación que se va a emplear, la especificación de las herramientas que se usan y los distintos escenarios sobre los que se realizaran las pruebas.



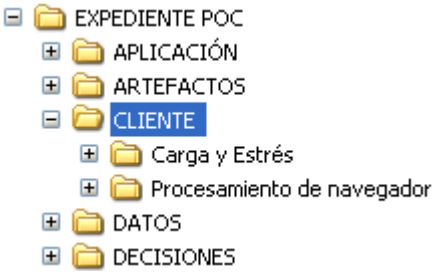
En el paquete *Decisiones* se especifican las decisiones arquitectónicas a las que se les realizarán las pruebas de concepto sobre escenarios de sistema, datos e integración.



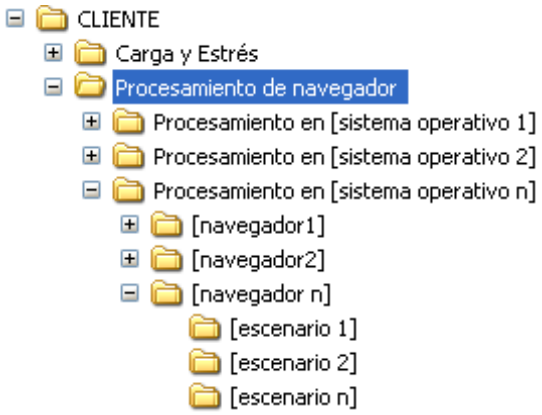
Los resultados de las pruebas se recogen en un documento para cada decisión arquitectónica y un documento resumen con los resultados de todas las pruebas realizadas a las distintas decisiones (Ver Anexos).

El objetivo de las pruebas de decisiones es verificar que las decisiones que se tomen sobre la arquitectura cumplan con las características de calidad del sistema para mitigar los riesgos y posibles errores que puedan surgir.

El paquete *Cliente* contiene los aspectos que influyen en la capa de presentación estructurados en forma de árbol, siendo los de mayor interés el procesamiento del navegador que será utilizado y las pruebas de carga y estrés a la aplicación.



En el paquete *procesamiento del navegador* se especifica el sistema operativo y dentro de este el navegador sobre el que se va a ejecutar la prueba, para cada navegador se especifican los diferentes escenarios a probar. De manera general la prueba se puede realizar para distintos sistemas operativos y distintos navegadores en cada uno de ellos describiéndolos por escenarios.



Los resultados de las pruebas se recogen en un documento para cada escenario exponiendo el resultado de las muestras y un documento resumen que recoge el resultado de manera general de las pruebas de Procesamiento de navegador para los distintos sistemas operativos (Ver Anexos).

El objetivo de ésta prueba es supervisar la cantidad de recursos de la unidad central de procesamiento (CPU) y de memoria que el navegador está usando cuando este está en ejecución.

En el paquete *Carga y Estrés* se especifica la capacidad en memoria RAM que tiene el servidor donde se encuentra la aplicación que se quiere probar y dentro de esta se especifican los distintos módulos a probar con sus respectivos componentes, cada componente tiene pruebas de estrés, donde se define la cantidad de usuarios concurrentes que se quieren simular; y pruebas de carga, donde se definen los escenarios que se tomarán en cuenta para la ejecución de la prueba para las cantidades de usuarios concurrentes que se quieren simular.

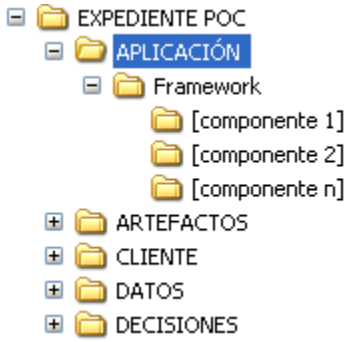
Los resultados de las pruebas, ya sea carga o estrés, se recogen en un documento para cada escenario de prueba con los resultados de las muestras, se incluye un documento que describe el caso de prueba que se utiliza para la ejecución de la misma y un documento que recoge el resultado de las pruebas de carga y estrés de manera general para cada una de las distintas capacidades de memoria RAM sobre las que se realizan las pruebas (Ver Anexos).

El objetivo de las pruebas de estrés es determinar el rendimiento de la aplicación, es decir, el tiempo de respuesta del servidor cuando uno o varios usuarios acceden de manera concurrente a determinada interfaz de la aplicación. Esto permite determinar cuantos usuarios concurrentes es capaz de soportar una aplicación y en que tiempo dichos usuarios obtienen respuesta o no a sus peticiones.

El objetivo de las pruebas de carga es descubrir ausencias de respuestas del servidor y medir los tiempos de respuestas del servidor ante determinadas acciones realizadas con la aplicación. En estas pruebas se sobrecarga el servidor, es decir, se envían y solicitan datos de manera que haya un amplio tráfico de datos. Esto permite conocer cuantos datos puede

procesar la aplicación y cuantas acciones de gestión soporta, determinar los aspectos que consumen mayor parte de rendimiento del sistema y con ello trazar estrategias de programación que ayuden a mejorar esas ineficiencias.

El paquete A*plicación* contiene los aspectos que influyen de alguna manera en la aplicación y que están relacionados con esta, siendo de mayor interés los distintos componentes del Framework.

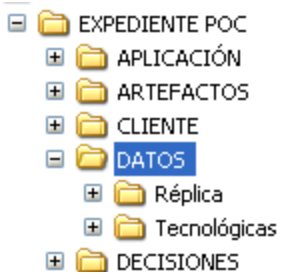


En el paquete *Framework* se especifican los diferentes componentes que van a ser evaluados, entre ellos se pueden encontrar Excepciones, Transacción, Validación en el servidor, Concurrencia, entre otros. Dentro de estos componentes se especifican los distintos escenarios a probar.

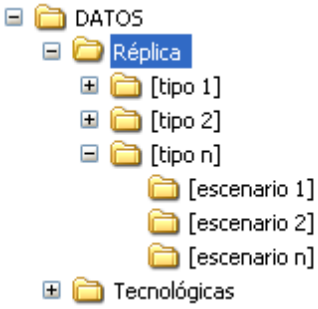
Los resultados de las pruebas se recogen en un documento para cada componente y un documento resumen que recoge los resultados de manera general de todos los componentes que se prueban en el Framework (Ver Anexos).

El objetivo de realizar estas pruebas es probar que cada elemento del Framework cumple con la funcionalidad para la que fue diseñado.

El paquete *Datos* contiene los aspectos que influyen en las capas de datos y los servicios expuestos por recursos externos a la aplicación como el gestor de datos que se utiliza.



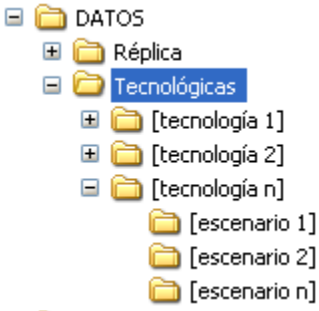
En el paquete *Réplica* se define el tipo de réplica que se quiere probar y los distintos escenarios asociados a esta.



Los resultados de las pruebas se recogen en un documento para cada escenario de prueba y un documento resumen con los resultados de manera general de todas las pruebas de réplica realizadas (Ver Anexos).

El objetivo de las pruebas de réplica es comprobar el intervalo de tiempo real en el cual se copian los datos entre las instancias de la base de datos, verificar la integridad de los mismos independientemente del tipo de réplica que se realiza y saber si soporta la cantidad de datos necesarios para cuando se despliegue.

En el paquete *Tecnológicas* se definen las tecnologías que se quieren probar, especificando los distintos escenarios a probar.



Los resultados de las pruebas se recogen en un documento para cada escenario de prueba y un documento resumen que recoge de manera general los resultados de todas las pruebas realizadas a las distintas tecnologías (Ver Anexos).

El objetivo de estas pruebas es calcular la tasa media de transacciones que puede realizar el servidor con determinada cantidad de usuarios conectados concurrentemente, realizando operaciones de actualizar, insertar y eliminar datos sobre una base de datos.

Actividad 8 - Realizar la Prueba de Concepto de la Arquitectura.

Esta actividad es realizada por Involucrados en la arquitectura y el *Grupo de evaluación*. Tiene como objetivo realizar las pruebas a cada elemento seleccionado a partir de los escenarios de prueba definidos, el Grupo de evaluación guía la evaluación aplicando las técnicas y método seleccionado. Cada prueba debe ser documentada y organizada según la estructura del expediente. Si se desean evaluar otros elementos de la arquitectura es necesario recomenzar el proceso desde la *actividad 3.*

Actividad 9 - Evaluar los Resultados de las Pruebas.

Esta actividad tiene como objetivo generar un Informe de Revisión que recoge los resultados de las pruebas realizadas, facilitando la toma de decisiones.

“En términos concretos, la evaluación de la arquitectura produce un informe, la forma y contenido del mismo varía según el método utilizado. En particular, produce repuestas a dos tipos de preguntas:

¿Es esta arquitectura adecuada para el sistema para la cual fue diseñada?

¿Cuál de dos o más arquitecturas propuestas es la más adecuada para el sistema?“ (3)

Actividad 10 - Efectuar la toma de decisiones.

Esta actividad es realizada por los *Tomadores de decisión del proyecto*. Tiene como objetivo a partir de los resultados de las pruebas, teniendo en cuenta el impacto de las decisiones arquitectónicas sobre los atributos de calidad, la toma de decisiones con respecto a los riesgos detectados y definir las restricciones que presenta la arquitectura.

2.4 Actividades de cada rol.

Durante el proceso de desarrollo de la evaluación de la arquitectura mediante la Guía Práctica que se propone, cada uno de los miembros del equipo de evaluación realiza un conjunto de actividades específicas según el rol que desempeñen dentro del proceso de evaluación.

Rol responsable	*Actividades*
Tomadores de decisión del proyecto.	Presentar el negocio.
Tomadores de decisión del proyecto.	Presentar la arquitectura.
Involucrados en la arquitectura Grupo de evaluación de la arquitectura.	Seleccionar elementos a evaluar.
Grupo de evaluación de la arquitectura.	Diseñar las pruebas de concepto.
Grupo de evaluación de la arquitectura.	Evaluar la viabilidad de la prueba de concepto de la Arquitectura.
Grupo de evaluación de la arquitectura.	Determinar el método de evaluación.
Grupo de evaluación de la arquitectura. Involucrados en la arquitectura	Estructurar el expediente para pruebas de concepto.
Involucrados en la arquitectura Grupo de evaluación de la arquitectura.	Realizar la Prueba de Concepto de la Arquitectura.
Grupo de evaluación de la arquitectura.	Evaluar los Resultados de las Pruebas.
Tomadores de decisión del proyecto	Efectuar la toma de decisiones.

Tabla 1 Actividades que desarrolla cada rol.

2.5 Orden de ejecución de las actividades.

Durante el proceso de desarrollo de la evaluación de la arquitectura mediante la Guía Práctica que se propone, las actividades deben ejecutarse de acuerdo a un orden lógico, que posibilite una mejor organización y ejecución del proceso de evaluación.

Capítulo 2

Para comenzar una evaluación de la arquitectura es necesario hacer una presentación del negocio al *Grupo de evaluación*. Posteriormente se presenta la arquitectura y es entonces cuando el *Grupo de evaluación* y los Involucrados en la arquitectura están en condiciones de definir los elementos a evaluar. A partir de este momento el *Grupo de evaluación* diseña la prueba de concepto y evalúa la viabilidad de la misma. En caso de que no sea posible la implementación de la prueba es necesario redefinir los elementos a evaluar y el diseño de la prueba de concepto. Una vez viable la realización de la prueba se selecciona el método de evaluación, se estructura el expediente de pruebas de concepto y se ejecuta la misma; si durante la ejecución de la prueba o al finalizar la misma se determina que es necesario evaluar o revaluar otros elementos, se recomenzaría todo el proceso a partir de la *actividad 3*, donde se definen los elementos a evaluar. Para finalizar se evalúan los resultados y a partir de estos se toman las decisiones.

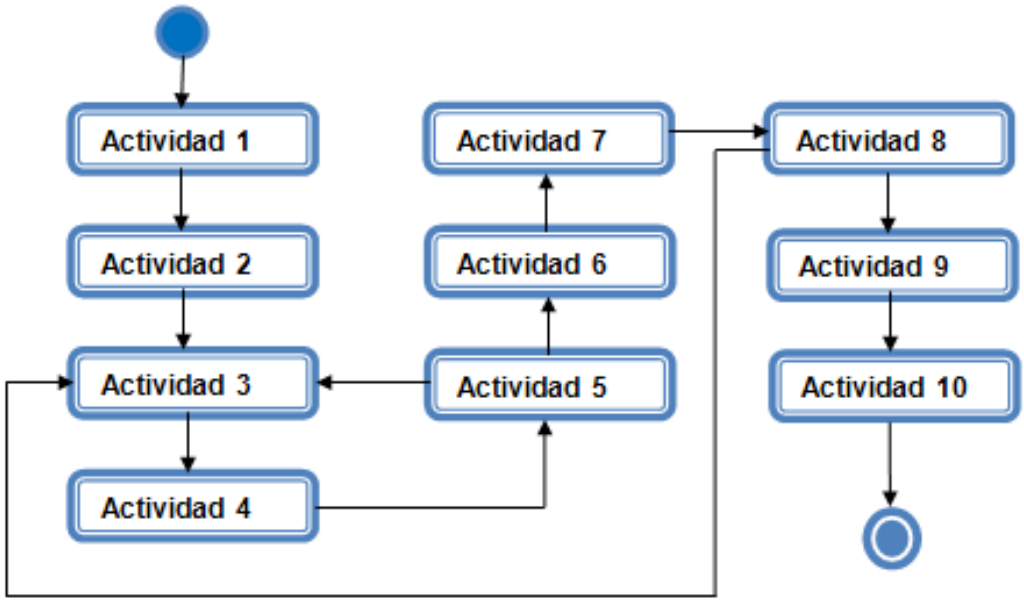


Figura 1 Esquema de Actividades.

2.6 Conclusiones.

Capítulo 2

En el presente capítulo han quedado definidos los roles que participan en la evaluación de la arquitectura, las actividades que realiza cada rol para la realización de la misma, el orden lógico de estas actividades y la descripción del expediente generado como resultado del desarrollo de la *Guía Práctica*. Si se realiza una prueba de concepto a los elementos de un software de gestión, esta guía apoyaría el proceso de realización de la misma contribuyendo a su planificación y organización.

3

CAPÍTULO

APLICACIÓN DE LA GUÍA PRÁCTICA A UN CASO DE ESTUDIO.

Capítulo 3

3.1 Introducción.

En el presente capítulo se desarrolla la Guía Práctica de actividades mediante la realización de pruebas de concepto a casos de estudio del Sistema de Integral de Gestión de Entidades, Cedrux, describiendo en cada una de las actividades propuestas, las acciones que se realizan y los resultados que muestran las pruebas realizadas, que son organizadas y documentadas según la estructura del expediente que se definido.

3.2 Desarrollo de la Guía Práctica para realizar pruebas de concepto.

Actividad 1 - Conocer el negocio.

Esta actividad se desarrolló mediante un taller en el que los *Tomadores de decisiones del proyecto* presentaron el negocio al Grupo de evaluación y Involucrados en la arquitectura, para así entender el contexto del sistema y los conductores primarios del negocio.

Al finalizar el taller se tiene una panorámica de las perspectivas y metas del negocio, las funciones más importantes del sistema y los principales objetivos de los atributos de calidad que forman la arquitectura. La siguiente figura es una muestra de algunos de los elemento a tener en cuenta para describir el proceso de negocio.

Capítulo 3

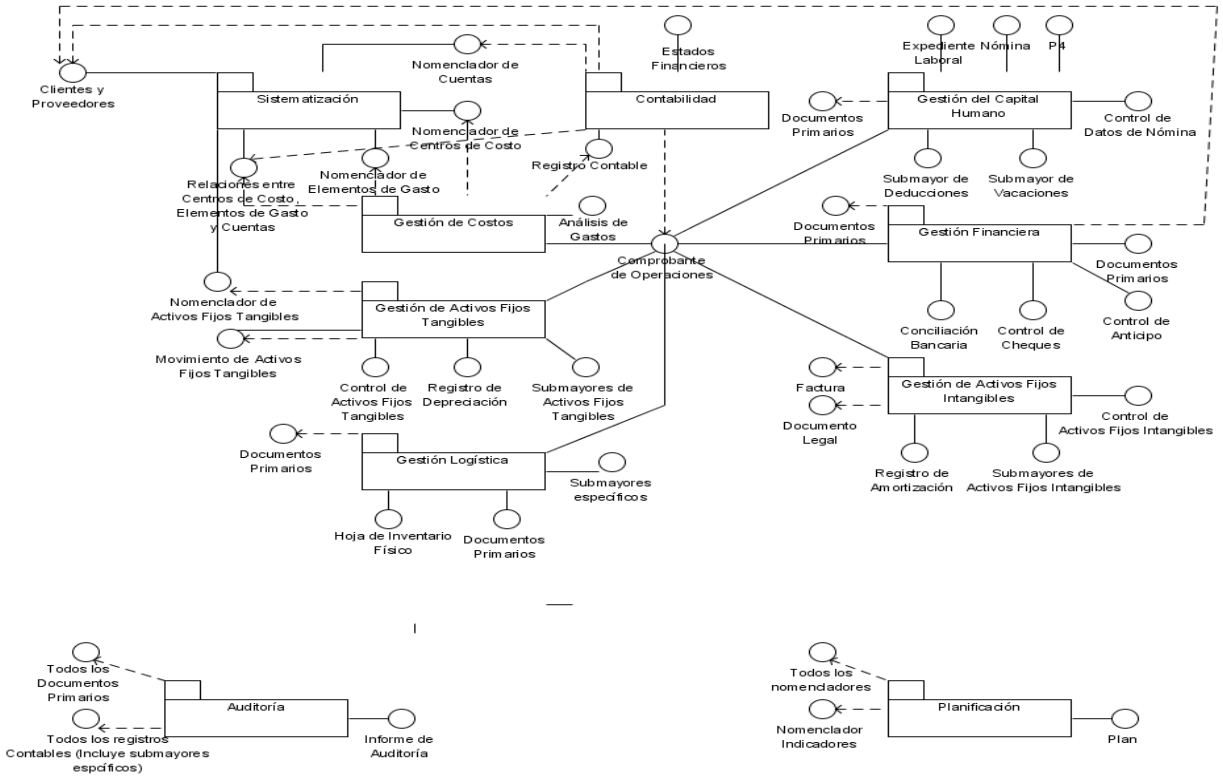


Figura 2 Mapa de Proceso.

Actividad 2 - Presentar la arquitectura.

Esta actividad se desarrolló mediante un taller en el que el *Arquitecto* describe la línea base de la arquitectura, logrando que se tenga una mejor comprensión del sistema y guiar al *equipo de desarrollo* a través del ciclo de vida del mismo; quedando explicados los elementos arquitectónicamente significativos (casos de uso, subsistemas, interfaces, algunas clases y componentes, nodos y colaboraciones).

Al finalizar el taller se ha comprendido con suficiente detalle el estado de desarrollo que tiene la arquitectura y los atributos de calidad que pueden ser afectados por determinada decisión arquitectónica. La siguiente figura muestra una breve descripción de la arquitectura del sistema.

Capítulo 3

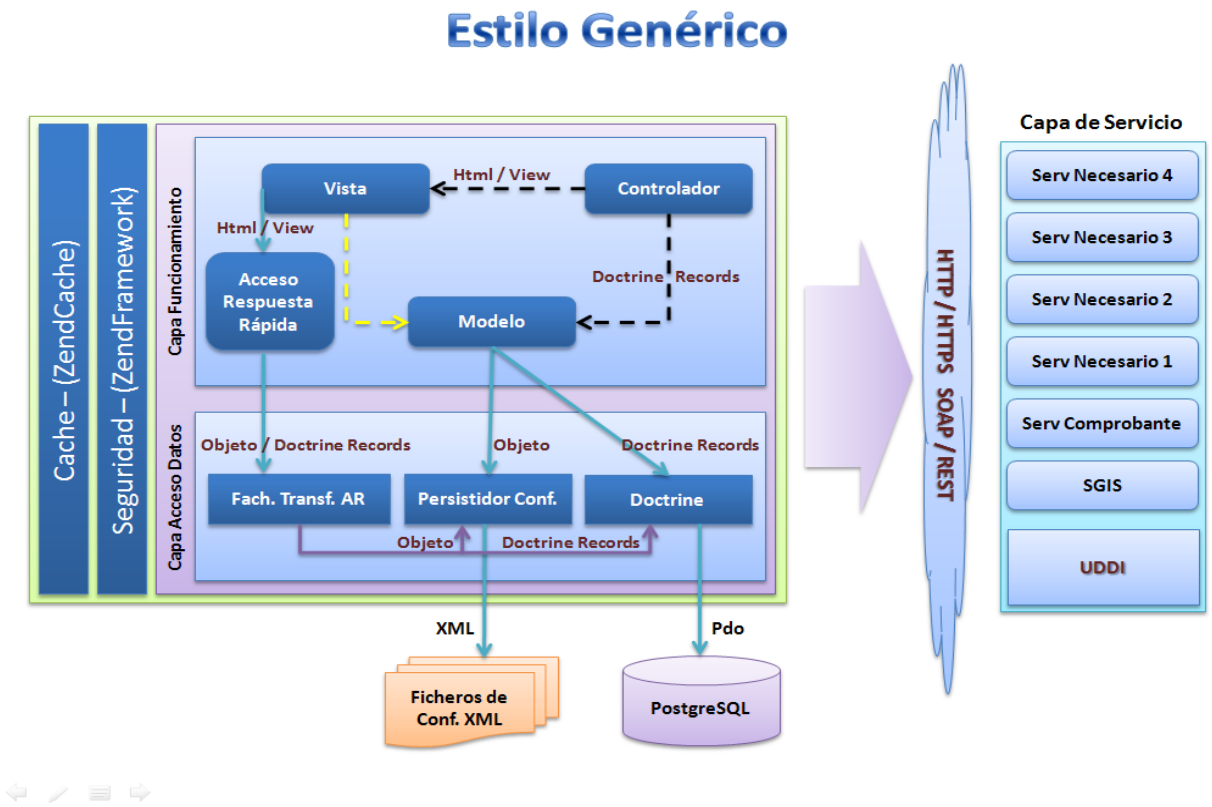


Figura 3 Arquitectura del Sistema.

Actividad 3 - Seleccionar elementos a evaluar.

En esta actividad los Involucrados *en la arquitectura* realizaron un taller en el que definieron las decisiones arquitectónicas que serían evaluadas por parte del *Grupo de evaluación de la arquitectura*, se definen mediante una tormenta de ideas sobre qué escenarios se realizarían las pruebas.

Las pruebas se orientaron hacia la evaluación de rendimiento del componente Formato y al navegador Mozilla Firefox en los Sistemas Operativos Windows y Linux, desde la dimensión del *Cliente*; pruebas de sistema y datos a la decisión Multimoneda, desde la dimensión de *Decisiones*; al componente Concurrencia del Framework, desde la dimensión de *Aplicación* y pruebas de réplica online y offline y de estrés al PostgreSQL desde la dimensión de *Datos.*

Al finalizar la tormenta de idea los escenarios que se definieron para apoyar la realización de las pruebas son:

- Un usuario accede a cualquier interfaz de la aplicación y esta debe estar disponible en un periodo de 0.1 a 0.2 segundos (Rendimiento/Disponibilidad).
- Si 100 000 usuarios concurrentes acceden a la interfaz principal de la aplicación, esta debe estar disponible para todos a la vez en un período de 0.5 a 1.0 segundos (Rendimiento/Disponibilidad).
- Se solicita un recurso al sistema, que puede ser ligero (capacidad por debajo de 2 MB), medio (capacidad de 2 a 10 MB) o complejo (más de 10 MB de capacidad) en un servidor de 1 GB de memoria RAM y se recibe en un período de 0.1 a 0.7, de 0.8 a 2.0 y de 3.0 a 5.0 segundos respectivamente. (Rendimiento/Disponibilidad).
- Se intercambian datos con el sistema, ya sea mediante acciones de inserción, búsqueda, eliminación o modificación de datos, en un servidor de 1 GB de memoria RAM y se recibe la notificación de la acción realizada en un período de 0.1 a 0.7 segundos (Rendimiento/Disponibilidad).
- Se desea eliminar o modificar un elemento de la base de datos que está siendo utilizado por otro elemento que depende de él; el sistema no permite que este elemento sea eliminado. (Borrar en cascada)(Seguridad/Integridad).
- Dos usuarios acceden a la misma interfaz de modificar e intentan modificar la misma información simultáneamente, el sistema solo debe dejar que modifique uno de ellos y notifica al otro usuario que actualice para obtener la última versión de los datos. (Seguridad/Integridad).
- Al configurar una nueva institución no se llega a la funcionalidad donde se va a definir un elemento que dependa de otro componente sin haber configurado el componente con antelación (Seguridad/Confiabilidad).
- Se modifican las características arquitectónicas del despliegue de la aplicación, el sistema deberá indicar las mismas cuando se realiza la próxima petición. (Modificabilidad).
- Se produce una excepción en los datos que no permite realizar la operación indicada por el usuario, el sistema mostrará un mensaje notificando el error pero solo con la información necesaria para el usuario, no con información del sistema. (Seguridad/Confiabilidad).

- En un ambiente de servidores, uno funcionando como centro de datos (servidor destino) y el resto como servidores locales (servidor fuente). Se realizaron réplicas de 50000 tuplas desde un servidor fuente hacia el servidor destino
    - Réplica online provocando fallos en la conexión.
    - Réplica online con funcionamiento normal.
    - Réplica offline mediante un fichero.

Actividad 4 - Diseñar las pruebas de concepto.

En esta actividad el equipo de evaluación teniendo el estado de desarrollo que tiene la arquitectura y que el desarrollo de la arquitectura de software del Sistema Integral de Gestión de Entidades, Cedrux, se basa en construir la arquitectura prácticamente al mismo tiempo del desarrollo de la aplicación, es decir que la arquitectura ha habido que diseñarla y desarrollarla a la vez, define que las evaluaciones se desarrollan es etapas tempranas y tardías, para que el equipo de trabajo tenga mayor confianza en el producto.

Teniendo en cuenta el objetivo de las evaluaciones se ejemplifican los resultados de manera cualitativa y cuantitativa utilizando las técnicas basadas en escenarios, en simulación y en experiencia.

La utilización de herramientas para la realización de pruebas de concepto es muy importante y se recomienda para una mayor efectividad de dichas pruebas ya que permite determinar con mayor precisión y en menor tiempo muchos de los riesgos asociados a la arquitectura. Normalmente existen paquetes de herramientas con un costo de cientos o miles de dólares, que con frecuencia son adquiridos mediante la lógica de que se obtiene lo que se paga. Algunas personas empiezan a darse cuenta que el software de dominio público no siempre es inferior y, a veces, en realidad es superior. Por lo que las herramientas utilizadas para la realización de las pruebas de concepto son de software libre.

Las herramientas a utilizar para automatizar de las Pruebas de Concepto al Sistema Integral de Gestión de Entidades, Cedrux son:

Maquina Virtual Java (MVJ).

Es el núcleo del lenguaje de programación Java. Su misión principal es la de garantizar la portabilidad de las aplicaciones Java. Existe una única especificación de la máquina virtual, que

proporciona una vista independiente del hardware y del sistema operativo sobre el que se esté trabajando. De esta manera un programador en Java "escribe su programa una vez, y lo ejecuta donde sea"[1]. Es imposible ejecutar un programa Java sin ejecutar alguna implantación de la MVJ; es decir, el código Java no se ejecuta directamente sobre un procesador físico, sino sobre un **procesador virtual Java** (8). En este caso se usaría como plataforma para la ejecución de la herramienta JMeter.

JMeter.

Es un proyecto de Apache Jakarta, esta herramienta puede ser usada para realizar pruebas de Carga y Estrés, para así analizar y mediar el rendimiento de las aplicaciones de distintos tipos, aunque especialmente en Aplicaciones Web. JMeter se considera una herramienta de generación de carga, aunque tenga más utilidades. Soporta, por ejemplo, realizar comprobaciones sobre las respuestas, cookies por hilos de ejecución, configuración de variables y diversos reportes, entre otros.

Utilizar JMeter en aplicaciones web para la comprobación de los recursos del sistema, supone una mayor efectividad en el proceso y en la fiabilidad de los resultados. JMeter como herramienta de prueba dispone de varios componentes que facilitan la elaboración de los escenarios de prueba con la ventaja de simular para cada uno de esos escenarios miles de usuarios.

pgBench.

Es un programa simple y compilado en el lenguaje de programación C, que sirve para realizar pruebas sobre PostgreSQL, el mismo permite calcular la tasa media de transacciones que puede realizar el servidor con determinada cantidad de usuarios conectados concurrentemente realizando operaciones de actualizar, insertar y eliminar datos sobre una base de datos.

Administrador de tareas.

Es una herramienta que permite supervisar los procesos (programas) que se están ejecutando en el ordenador. Existen diferentes formas de acceder a él en Windows, como puede ser Pulsar las teclas Ctrl + Alt + Supr. Agregando columnas a la información que aparece en la ficha Procesos. Estas columnas ofrecen información acerca de cada proceso, como la cantidad de

[1] Eslogan que emplea Sun Microsystems: "write once, run anywhere".

recursos de la unidad central de procesamiento (CPU) y de memoria que el proceso está usando en esos momentos. Para esto se accede a una ventana. (Ver Anexo 13)

Herramienta de top en GNU/Linux.

Es una herramienta que permite supervisar los procesos (programas) que se están ejecutando en el ordenador. Existen diferentes formas de acceder a la misma en GNU/Linux, para usarla es necesario ejecutar una terminal por ejemplo se puede usar xterm, gnome-terminal o konsole según el entorno de escritorio que se este usando, seguidamente se hace uso del comando *top* pasándole como parámetros: "–b" modo bash, este modo muestra los datos en la consola y luego termina el comando top, "–H" para que muestre los hilos que tienen las aplicaciones que están corriendo en el sistema y " –n 1" para que solo se ejecute una vez, al finalizar se pulsa la tecla "Enter" (Ver Anexo 14)

En el proceso de elaboración de pruebas el uso de herramientas trae consigo numerosas ventajas, entre ellas, mayor rapidez de ejecución, uso de menos recursos o evitar pruebas obsoletas. Una herramienta de pruebas sin saber técnicas, prácticas y tener experiencia en pruebas es tan útil como un entorno de programación sin saber el lenguaje. También es importante tener en cuenta el desarrollo de la tecnología, para usar herramientas que no estén obsoletas y se ajusten a las necesidades de una aplicación que puede estar desarrollada con una tecnología avanzada.

Actividad 5 - Evaluar la viabilidad de la prueba de concepto de la Arquitectura.

En esta actividad el *Grupo de evaluación* realizó un taller en el que mediante un análisis del diseño de pruebas de concepto, los elementos a evaluar, herramientas que se han definido para evaluar la arquitectura, el estado de desarrollo de la arquitectura y si el grupo de evaluación conoce bien la arquitectura del sistema para realizar las pruebas, se definió que es viable realizar las pruebas las pruebas de concepto a la arquitectura.

Actividad 6 - Determinar el método de evaluación.

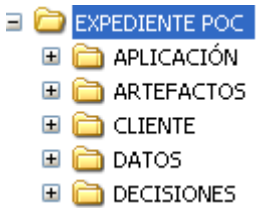
En esta actividad se determinó por los *Tomadores de decisión del proyecto* y el *Grupo de evaluación de la arquitectura* que el método para realizar la evaluación es el método ATAM, ya que en este se evalúan los atributos de calidad que se necesitaban alcanzar o perfeccionar; emplea una técnica cualitativa basada en escenarios, que permite validar una arquitectura

desde etapas tempranas del desarrollo, con un conjunto de pasos y salidas definidas; y no presenta ninguna restricción con respecto a la característica de calidad a evaluar.

Actividad 7 - Estructurar el expediente para pruebas de concepto.

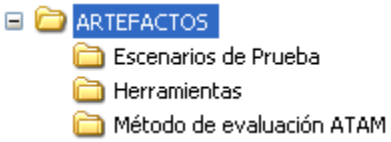
El expediente para las pruebas de concepto se estructuró de acuerdo a las dimensiones definidas, teniendo en cuenta los elementos a evaluar que se definieron en la actividad 3.

La estructura del expediente quedó de la siguiente manera:



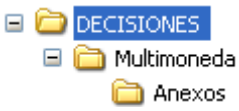
Artefactos.

En este paquete se especifican los escenarios de prueba, las herramientas de soporte para la realización de las pruebas y el método de evaluación: ATAM.



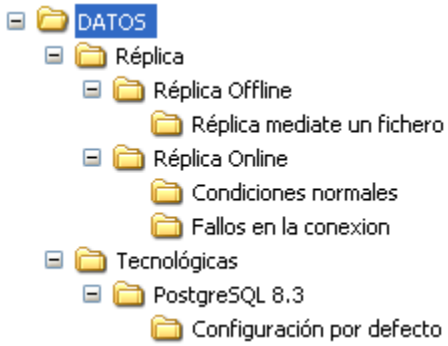
Decisión.

En este paquete se especifican las pruebas a la decisión Multimoneda.



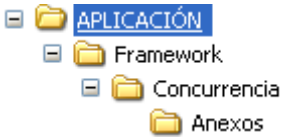
Datos.

En este paquete se especifican las pruebas tecnológicas que incluyen las pruebas de configuración por defecto del PostgreSQL y las de pruebas de réplica Online con fallos en la conexión y en condiciones normales y las de réplica Offline mediante ficheros.



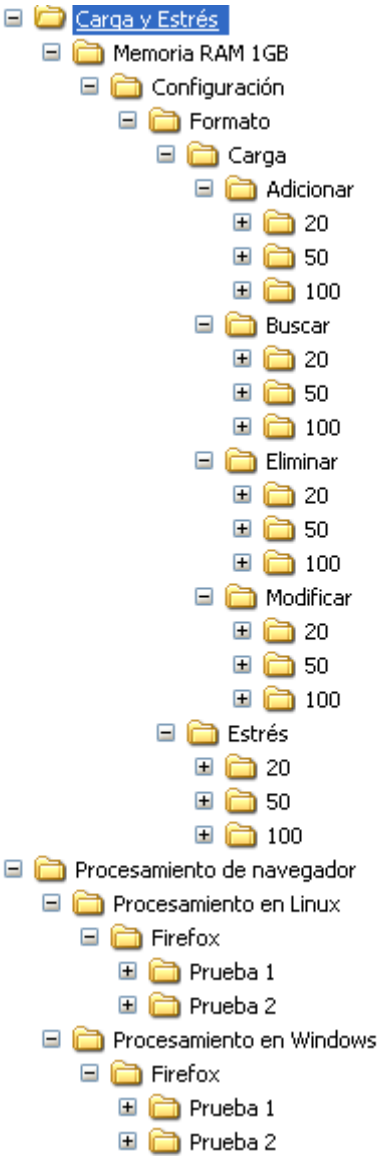
Aplicación.

En este paquete se especifica las pruebas al componente Concurrencia del Framework.



Cliente.

En este paquete se especifican las pruebas de Carga y Estrés que se realizarán al paquete Cliente, específicamente al caso de estudio Formato de Configuración, con una capacidad de 1 GB de memoria RAM, las pruebas se organizan por casos de prueba y dentro de estos las capacidades de usuarios concurrentes. Las pruebas de Procesamiento del navegador, se especifican para dos sistemas operativos, Linux y Windows, en ambos el mismo navegador: Firefox, indicando 2 escenarios de prueba para cada Sistema Operativo.



Actividad 8 - Realizar la Prueba de Concepto de la Arquitectura.

Las pruebas de concepto se realizaron para cada uno de las dimensiones definidas, teniendo en cuenta los aspectos a evaluar definidos en la actividad 3, organizadas y documentadas según la estructura del expediente definido en la actividad 7.

Artefactos.

En el paquete *Artefactos* se especificaron los resultados de la aplicación del método de evaluación que se empleó, las herramientas que se utilizaron para la realización de las pruebas y un documento que contiene los distintos escenarios sobre los que se realizaron las pruebas.

Aplicación del método ATAM.

Al realizar la evaluación arquitectónica se puede decir que los atributos de calidad que más prioridad se les dio por el riesgo que representan fueron Disponibilidad, Rendimiento y Modificabilidad.

Se pudo observar que la utilización de ATAM es adecuada para la evaluación de este tipo de arquitecturas, ya que el método permite una evaluación temprana; es decir, no es necesario que toda la arquitectura este definida para ser evaluada; el comportamiento observado en el alcance de la arquitectura evaluada se propaga a la arquitectura completa. Por esta razón se afirma que el comportamiento conseguido en la evaluación es válido para futuros componentes que se integren a la arquitectura de software, si ellos cumplen con los mismos patrones que los ya evaluados.

Escenario:	Un usuario intercambia datos con el sistema.		
Atributo:	Rendimiento, Reusabilidad		
Ambiente:	En un servidor de 1 GB de memoria RAM en estado normal de explotación.		
Estímulo:	Se realizan operaciones en el sistema ya sea mediante acciones de inserción, búsqueda, modificación o eliminación.		
Respuesta:	Se recibe la notificación de la acción realizada en un período de 0.1 a 0.7 segundos.		
Decisiones de la arquitectura	*Riesgo*	*Puntos de sensibilidad*	*Puntos de desventaja*
DA1:	Disminuye el	Disminuye el rendimiento pero	Para cualquier acción tienen que

Cuando se interactúa con una vista se dispara una clase controladora o sea, un js que llama a un controlador, este controlador no se comunica directamente con la clase que implementa el negocio si no que carga e instancia el objeto de doctrine. Este objeto carga de la base de datos desde donde va a las tablas por PDO. El Data Base Object a partir del PDO va a las tablas, carga la información construye el objeto y se lo devuelve al controlador que se comunica con la entidad y le pasa el objeto de dominio, se procesa la lógica en el gestor encargado de la lógica propia del negocio, cálculos y procesamiento del negocio. El controlador si tiene que persistir los datos vuelve a ir al doctrine le manda el objeto. El doctrine repite todo el mecanismo y regresa y concluye la aplicación. Una vez que concluye el controlador re-dibuja el js.	rendimiento.	aumenta la reutilización.	ejecutarse por 6 componentes arquitectónicos.
Razonamiento:	El Framework a partir de cada tabla genera un objeto y toda la lógica asociada a modificar, eliminar, buscar y adicionar sin programar las clases. Solo se programa la lógica de negocio y el controlador que tiene. Aun cuando el rendimiento se ve afectado debido a que tiene que moverse por 6 componentes arquitectónicos es más productivo por que se genera mucho código automático que aumenta la reutilización. Esta evita tener que crear una cadena de conexión, la transacción, evita por cada tabla tener que crear una clase de atributos y además crear los métodos de adicionar,		

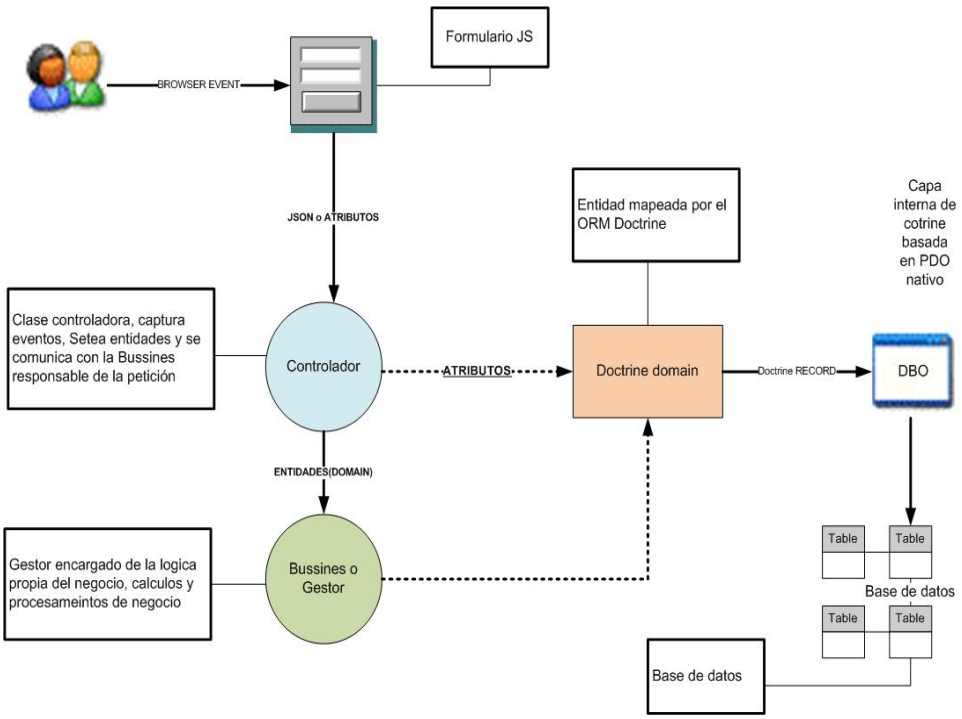
	modificar, eliminar y buscar. Todo esto afecta el rendimiento y por tanto cuando se haga una petición cuando se cargan los objetos no sean más de 20 y se haga el procesamiento por lote, si se quiere mover 100 objetos se hacen peticiones de 20 en 20.
Diagrama de la arquitectura: 	

Tabla 2 Análisis de escenario (Aplicación de ATAM)

Decisiones.

Las pruebas a las decisiones tomadas sobre el subsistema *Multimoneda* se desarrollaron realizando observaciones sobre escenarios de sistema y escenarios de datos, teniendo en cuenta la importancia que tiene para otros subsistemas que dependen de él.

El subsistema Multimoneda es el encargado de controlar y gestionar las operaciones que se realicen en torno a temas monetarios y referentes a la dualidad monetaria. Estas funcionalidades son imprescindibles para el trabajo en otros subsistemas.

El subsistema le presta servicios a los siguientes subsistemas:

- **Contabilidad** (Gestión de moneda contable, Gestión de tasas, Gestión de moneda por entidad, Reevaluación de cuentas)
- **Caja y Banco** (Gestión de moneda contable, Gestión de tasas, Gestión de moneda por entidad, Reevaluación de cuentas, Desglose)
- **Cobro y Pagos** (Gestión de moneda contable, Gestión de tasas, Gestión de moneda por entidad, Reevaluación de cuentas)
- **Inventario** (Gestión de moneda contable, Gestión de tasas, Gestión de moneda por entidad)
- **Costos y Procesos** (Gestión de moneda contable, Gestión de tasas, Gestión de moneda por entidad)
- **Activos Fijos** (Gestión de moneda contable, Gestión de tasas, Gestión de moneda por entidad)
- **Facturación** (Gestión de moneda contable, Gestión de tasas, Gestión de moneda por entidad)
- **Nomina** (Gestión de moneda contable, Gestión de tasas, Gestión de moneda por entidad)
- **Planificación empresarial y presupuestada** (Gestión de moneda contable, Gestión de tasas, Gestión de moneda por entidad)

Pruebas de Sistema.

El objetivo de la pruebas de *Sistema* es verificar que las decisiones tomadas que influyen en la arquitectura de sistema son las óptimas.

Estas pruebas se ejecutaron mediante pruebas de rendimiento al servidor teniendo en cuenta que las peticiones que este subsistema realiza al mismo no lo sobrecarguen.

Para la realización de esta prueba se hicieron pruebas de rendimiento al subsistema cuando se solicita un formulario y cuando se intercambian datos con el sistema.

Capítulo 3

Se solicita el formulario de Gestión de Moneda Contable realizando 46 peticiones simultáneas, que incurren en una transferencia de datos de aproximadamente 49882.1 KB/sec. Mostrando un tiempo de recuperación de aproximadamente 3.0 segundos.

Se intercambian datos con el sistema a través del formulario de Gestión de Moneda Contable realizando 48 peticiones simultáneas, que incurren en una transferencia de datos de aproximadamente 64895.6 KB/sec. Mostrando un tiempo de recuperación de aproximadamente 3.7 segundos.

Pruebas de datos.

El objetivo de estas pruebas es verificar que las decisiones tomadas que influyen en la arquitectura de datos son las óptimas.

Estas pruebas se ejecutaron mediante consultas a la Base de Datos y para su realización se crearon dos tablas, una *mod_datosmaestros.dat_moneda* y la otra *mod_datosmaestros.nom_moneda*, en las cuales existe una relación donde las monedas se nomenclan y por lo tanto el *id_moneda* del nomenclador *nom_moneda* va a pasar a la tabla *dat_moneda* como una llave foránea.

Para la realizar la prueba se intentó eliminar en la tabla *nom_moneda* a través de la consulta:

```
ALTER TABLE "mod_datosmaestros"."nom_moneda"
 DROP COLUMN "id_moneda";
```

Al intentar eliminar este elemento, el sistema muestra el mensaje:

> ERROR: No se puede eliminar tabla mod_datosmaestros.nom_moneda columna id_moneda porque otros objetos dependen de él.
>
> HINT: Use DROP... CASCADE para eliminar además los objetos dependientes.

Para la realizar la prueba se intentó modificar en la tabla *nom_moneda* a través de la consulta:

```
ALTER TABLE "mod_datosmaestros"."nom_moneda"
  UPDATE "mod_datosmaestros"."nom_moneda"
    SET id_moneda = 5 WHERE id_moneda = 2;
```

Al intentar modificar este elemento, el sistema muestra el mensaje:

> ERROR: No se puede actualizar tabla mod_datosmaestros.nom_moneda columna id_moneda porque otros objetos dependen de él
>
> HINT: Use UPDATE... CASCADE para actualizar además los objetos dependientes.

Cliente.

En las pruebas realizadas al paquete *Cliente*, se encuentran las pruebas de carga y estrés realizadas al subsistema Formato y de procesamiento al Firefox en los sistemas operativos Windows y Linux.

Pruebas de Carga y Estrés.

Las pruebas de *Carga y Estrés* se realizaron al caso de estudio Formato, en ellas se simularon distintas cantidades de usuarios concurrentes que interactúan con la aplicación.

Estas pruebas se realizaron a partir de un caso de prueba para cada uno de los escenarios: Adicionar, Modificar Eliminar, Buscar y Acceder a la Interfaz principal, para las distintas capacidades de memoria RAM y usuarios concurrentes. Para 1 GB de memoria RAM, se simulan 20, 50 y 100 usuarios concurrentes con el objetivo de probar como se comporta el rendimiento de la aplicación en diferentes condiciones. Estas pruebas se realizaron para cada uno de los escenarios y se recogieron 3 muestras por escenario para promediar los resultados.

Utilizando la herramienta JMeter, a partir de la grabación de los escenarios, se agregan el Grupo de Hilos al plan de pruebas, donde se especifica la cantidad de usuarios concurrentes; el Árbol de Resultados, donde se muestran detalles de los elementos que han sido accedidos; el Reporte Agregado que muestra detalles de los resultados de la prueba en forma de tabla y la Aserción de Respuesta donde se especifica el tipo de respuesta, en este caso se indica el código 200, ya que es el código de respuesta que indica que la petición es válida, es decir, que la conexión no es fallida. Finalmente se ejecuta la prueba (Ctrl + R).

Como resultado de las pruebas se obtiene un reporte que muestra un soporte de concurrencia por escenario y por diferentes cantidades de usuarios concurrentes así como las peticiones que han sido fallidas durante la ejecución de la prueba. Estas pruebas demuestran que el sistema no es capaz de soportar más de 100 usuarios conectados concurrentemente.

Las pruebas de Estrés para un servidor de 1 GB de memoria RAM, al caso de estudio Formato, soportan concurrentemente capacidades de 20, 50 y 100 usuarios concurrentes.

Para 20 usuarios se realizan 960 peticiones simultáneas con un tiempo de recuperación de aproximadamente 7.7 segundos. Para 50 usuarios se realizan 2400 peticiones simultáneas con un tiempo de recuperación de 8.2 segundos aproximadamente y para 100 usuarios se realizan 4800 peticiones simultáneas de un tiempo de recuperación de 29.8 segundos.

Durante la ejecución de las pruebas para las distintas cantidades de usuarios concurrentes se registran peticiones fallidas que arrojan un error de un 8.33 %. La principal causa del error fue a las peticiones:

- */erp2/comun/frameworks/ExtJS/idioma/es/temas/default/images/s.gif*
- */repo/VDesarrollo/erp2/comun/frameworks/ExtJS/idioma/es/temas/default/images/images/desktop/shared/icons/fam/user.gif*

La siguiente figura muestra el soporte de concurrencia para las pruebas de Estrés por cantidades de usuarios concurrentes al caso de estudio Formato.

Cantidad de Usuarios concurrentes.	Cantidad de peticiones.	Tiempo de recuperación/Seg
20	960	7.7
50	2400	8.2
100	4800	29.8

Tabla 3 Pruebas de Estrés.

Las pruebas de Carga para un servidor de 1 GB de memoria RAM, al caso de estudio Formato, muestran que soportan concurrentemente capacidades de 20, 50 y 100 usuarios concurrentes dentro de los escenarios Modificar, Eliminar, Buscar, Adicionar.

Para el escenario Modificar con 20 usuarios se realizan 1160 peticiones simultaneas muestra un tiempo de recuperación de aproximadamente 6.5 segundos; para 50 usuarios se realizan 2900 peticiones simultáneas muestra un tiempo de recuperación de 6.7 segundos aproximadamente y para 100 usuarios se realizan 5800 peticiones simultáneas muestra un tiempo de recuperación de 36.6 segundos. Durante la ejecución de las pruebas para las

distintas cantidades de usuarios concurrentes se registran peticiones fallidas que arrojan un error de un 8.62 %. La principal causa del error fue a las peticiones:

- */erp2/comun/frameworks/ExtJS/idioma/es/temas/default/images/s.gif*

- */repo/VDesarrollo/erp2/comun/frameworks/ExtJS/idioma/es/temas/default/images/images/desktop/shared/icons/fam/user.gif*

Para el escenario Adicionar con 20 usuarios se realizan 1040 peticiones simultaneas muestra un tiempo de recuperación de aproximadamente 5 segundos; para 50 usuarios se realizan 2600 peticiones simultáneas muestra un tiempo de recuperación de 6.4 segundos aproximadamente y para 100 usuarios se realizan 5200 peticiones simultáneas muestra un tiempo de recuperación de 39.0 segundos. Durante la ejecución de las pruebas para las distintas cantidades de usuarios concurrentes se registran peticiones fallidas que arrojan un error de un 9.62 %. La principal causa del error fue a las peticiones:

/repo/VDesarrollo/erp2/comun/frameworks/ExtJS/idioma/es/temas/default/images/images/desktop/shared/icons/fam/user.gif

- */erp2/comun/frameworks/ExtJS/idioma/es/temas/default/images/s.gif*

- */en-US/firefox/headlines.xml*

Para el escenario Eliminar con 20 usuarios se realizan 1080 peticiones simultaneas muestra un tiempo de recuperación de aproximadamente 5.2 segundos; para 50 usuarios se realizan 2650 peticiones simultáneas muestra un tiempo de recuperación de 6.1 segundos aproximadamente y para 100 usuarios se realizan 5300 peticiones simultáneas muestra un tiempo de recuperación de 38.5 segundos. Durante la ejecución de las pruebas para las distintas cantidades de usuarios concurrentes se registran peticiones fallidas que arrojan un error de un 7.55 %. La principal causa del error fue a las peticiones:

- */erp2/comun/frameworks/ExtJS/idioma/es/temas/default/images/s.gif*

- */repo/VDesarrollo/erp2/comun/frameworks/ExtJS/idioma/es/temas/default/images/images/desktop/shared/icons/fam/user.gif*

Para el escenario Buscar con 20 usuarios se realizan 1000 peticiones simultaneas muestra un tiempo de recuperación de aproximadamente 7.2 segundos; para 50 usuarios se realizan 2500 peticiones simultáneas muestra un tiempo de recuperación de 7.3 segundos aproximadamente y para 100 usuarios se realizan 5000 peticiones simultáneas muestra un tiempo de recuperación de 38.7 segundos. Durante la ejecución de las pruebas para las distintas cantidades de usuarios concurrentes se registran peticiones fallidas que arrojan un error de un 8.0 %. La principal causa del error fue a las peticiones:

- */repo/VDesarrollo/erp2/comun/frameworks/ExtJS/idioma/es/temas/default/images/images/desktop/shared/icons/fam/user.gif*
- */erp2/comun/frameworks/ExtJS/idioma/es/temas/default/images/s.gif*

La siguiente figura muestra el soporte de concurrencia para las pruebas de Carga por escenario y por cantidades de usuarios concurrentes al caso de estudio Formato.

Escenario de prueba.	Cantidad de Usuarios concurrentes.	Cantidad de peticiones.	Tiempo de recuperación/Seg.
Modificar	20	1160	6.5
	50	2900	6.7
	100	5800	36.6
Adicionar	20	1040	5
	50	2600	6.4
	100	5200	39.0
Eliminar	20	1080	5.2
	50	2650	6.1
	100	5300	38.5
Buscar	20	1000	7.2
	50	2500	7.3
	100	5000	38.7

Tabla 4 Pruebas de Carga.

Pruebas de Procesamiento del Firefox.

Esta prueba se realizó sobre los sistemas operativos Windows y Linux. Para ambos sistemas operativos se mide como se comporta el Firefox usando la aplicación mediante un caso de estudio y se compara el comportamiento del navegador en ambos sistemas operativos. Para ambas pruebas se toman las mediciones de los siguientes parámetros.

- *Uso CPU %:* Indica el porcentaje de tiempo en el que un proceso ha usado la desde la última actualización.
- *Uso CPU Tiempo:* Indica el tiempo total de procesador, en segundos, utilizado por un proceso desde que se inició.
- *Uso de memoria:* Indica cuánta memoria RAM está ocupando cada programa.
- *Uso de memoria virtual:* Cantidad de memoria virtual reservada para uso en un proceso.
- *Cantidad de hilos:* El número de subprocesos que se ejecutan en un proceso.

Para cada uno de los sistemas operativos se realizaron dos pruebas:

Prueba 1: Consiste en activar el Firefox y poner el portal en ejecución accediendo a la aplicación.

Prueba 2: Consiste en activar el Firefox, poner el portal en ejecución accediendo a la aplicación e interactuando con ella mediante el caso de prueba Adicionar Formato.

Prueba en Windows: Para esta prueba se configura el Administrador de Tareas en la ficha *Procesos* para activar los parámetros vistos anteriormente.

Los recursos utilizados fueron:

- Navegador: Mozilla Firefox
- Administrador de Tareas
- PC cliente 512M
- Sistema operativo Windows

Prueba 1

Número	CPU	Tiempo CPU	Uso de memoria	Tamaño de MV	Hilos

1	00	0:00:06	49352K	39272K	13
2	00	0:00:04	49176K	39092K	13
3	02	0:00:05	49640K	39436K	13
Promedio	00	0:00:05	49389K	39266K	13

Tabla 5 Prueba 1 al navegador Mozilla Firefox en el sistema operativo Windows.

Prueba 2

Número	CPU	Tiempo CPU	Uso de memoria	tamaño de MV	Hilos
1	02	0:00:14	57216K	47100K	12
2	02	0:00:10	57404K	47420K	13
3	00	0:00:11	57316K	47500K	13
Promedio	01	0:00:12	57312K	47340K	13

Tabla 6 Prueba 2 al navegador Mozilla Firefox en el sistema operativo Windows.

Prueba en GUN/Linux: Se ejecuta el comando *top* en una consola donde se mostrarán los procesos que se están ejecutando en esos momentos en la cuenta de usuario. Se observan los parámetros vistos anteriormente:

Los recursos utilizados fueron:

- Navegador: Mozilla Firefox
- Comando *top*
- PC cliente 512M
- Sistema Operativo GNU/Linux

Prueba 1

Número	CPU	Tiempo CPU	Uso de memoria	Tamaño de MV	Hilos
1	00	0:00.97	58368K	150528K	7
2	14	0:01:28	59392K	152576K	7
3	00	0:00:02	93184K	186368K	7
Promedio	04	0:00:75	70314K	163157K	7

Tabla 7 Prueba 1 al navegador Mozilla Firefox en el sistema operativo Linux.

Prueba 2

Número	CPU	Tiempo CPU	Uso de memoria	Tamaño de MV	Hilos
1	00	0:02:17	91136K	184320K	7
2	04	0:02:75	91136K	183296K	7
3	00	0:00:03	91136K	183296K	7
Promedio	01	0:01:65	91136K	183637K	7

Tabla 8 Prueba 2 al navegador Mozilla Firefox en el sistema operativo Linux.

Para obtener los resultados de la prueba a Firefox el indicador de comparación que se tuvo en cuenta fue el comportamiento que tuvo éste en ambos sistemas operativos tomando como referencia los parámetros: CPU, Tiempo de CPU, Uso de Memoria, Tamaño de Memoria Virtual y Cantidad de Hilos. Las siguientes tablas muestran los resultados de la Prueba1 y la Prueba 2 comparando ambos sistemas operativos para cada una de ellas.

Prueba 1:

Sistema operativo	CPU	Tiempo CPU	Uso de memoria	Tamaño de MV	Hilos
Windows	00	0:00:05	49389K	39266K	13
Linux	04	0:00:75	70314K	163157K	7

Tabla 9 Comportamiento del navegador Mozilla Firefox en Prueba 1.

Prueba 2

Sistema operativo	CPU	Tiempo CPU	Uso de memoria	Tamaño de MV	Hilos
Windows	01	0:00:12	57312K	47340K	13
Linux	01	0:01:65	91136K	183637K	7

Tabla 10 Comportamiento del navegador Mozilla Firefox en Prueba 2.

Las pruebas demuestran que para ambos escenarios en el sistema operativo Linux se consume más memoria que en Windows, sin embargo en el sistema operativo Windows se consumen menor cantidad de recursos.

Aplicación.

En las pruebas realizadas al paquete de aplicación, se encuentran las pruebas al componente de concurrencia del Framework.

Capítulo 3

Pruebas de Concurrencia.

En las prueba al componente *Concurrencia* se persigue verificar que un mismo campo puede ser accedido por uno o varios usuarios al unísono con el fin de realizar una modificación.

En la realización de las pruebas de concurrencia, dos clientes acceden a la aplicación para modificar los datos de un usuario determinado. Ambos envían las peticiones de modificación al mismo tiempo, llega una petición primero que la otra. La primera petición realiza la acción y la Base de datos hace una actualización incrementando la versión, se notifica al cliente que la modificación ha sido correcta y la segunda petición identifica un error operacional en la versión y es notificado al cliente.

Al iniciar la prueba ambos usuarios se encuentran con la última versión del recurso.

Con la primera petición la Base de datos se hace una actualización inminente, incrementando la versión. El sistema muestra el mensaje *"El usuario fue modificado satisfactoriamente"*

Con la segunda petición el control de concurrencia identifica un error operacional en la versión y es notificado. El sistema no permite realizar la modificación y muestra el mensaje *"Los datos del usuario fueron modificados por otra persona, por favor recargue los datos e intente realizar la modificación nuevamente".*

En caso de que la segunda petición modifique el recurso al cual esta accediendo, notificando una correcta operación e invalidando la acción de la primera petición, la aplicación estaría trabajando bajo un error de concepto.

Datos.

En las pruebas realizadas al paquete de datos, se encuentran las pruebas al gestor de datos PostgreSQL 8 y a las réplicas de datos.

Réplica.

En las pruebas de réplica de datos realizadas se encuentran las pruebas de Réplicas online y las pruebas de Réplicas offline:

Réplicas online

Capítulo 3

El objetivo de realizar las pruebas es comprobar el intervalo de tiempo real en el cual se copian los datos entre las instancias de Base de datos, verificar la integridad de los mismos y la cantidad de datos que soporta. Las pruebas de réplica online se dividen en dos tipos una observando el funcionamiento normal de la replica y otra provocando un fallo en la conexión.

Para la prueba de réplica online en condiciones normales, se montaron 2 bases de datos, una funcionando como servidor local (servidor fuente) y otra como centro de datos (servidor destino). Se realiza una actualización de 50 000 tuplas en el servidor local. Se dispara la tarea programada por el sistema. Una vez detectados los cambios el sistema de réplica empieza a intercambiar la información, la cual se envía íntegra hacia el servidor destino. Tomándose un tiempo de replicación de los datos de 7.0 minutos.

Para la realización de las pruebas de réplica online con fallo en la conexión se montaron 2 bases de datos, una funcionando como servidor fuente y otra como servidor destino. En el servidor local se realiza una actualización de 50 000 tuplas. Se dispara la tarea programada por el sistema. Una vez detectados los cambios el sistema de réplica empieza a intercambiar la información. Se provoca el fallo en la conexión a los 10.0 segundos de comenzada la réplica. Como los datos fueron leídos se ponen temporalmente en la base de datos del centro de datos. Cuando el centro de datos va a sincronizar las fechas locales de los servidores no encuentra al servidor local. Al no encontrar el host muestra un error de conexión y no se realiza la réplica. Se reanuda la conexión y cuando se vuelve a disparar la tarea programada por el sistema la información se envía íntegra hacia el servidor local. Tomándose un tiempo de replicación de los datos de 7.07 minutos.

Para la realización de las pruebas de *réplica offline* se montaron 2 bases de datos, una funcionando como servidor fuente y otra como servidor destino. En el servidor fuente se realizó una actualización de 50 000 tuplas. Se ordena la replicación entre el servidor fuente y el fichero y se copia el fichero en un dispositivo de almacenamiento. Por ultimo se procede a realizar la replicación entre el fichero y el servidor destino. Tomándose el tiempo de replicación del fichero a la Base de Datos destino el cual fue de 7.0 minutos.

En cada una de las pruebas esta operación se realizó 10 veces para obtener un promedio del tiempo de replicación.

Capítulo 3

Estrés al PostgreSQL

Para la realización de esta prueba se usa PgBench, ya que sirve para calcular la tasa media de transacciones que puede realizar el servidor con determinada cantidad de usuarios conectados concurrentemente. Se insertaron 1 000 000 tuplas en la tabla *account* y se conectan 10 y 90 usuarios concurrentes respectivamente para cada prueba realizando cada uno 10 000 transacciones.

Se realizaron las pruebas al servidor con la configuración siguiente:

Parámetro	Valor	Descripción
RAM	512MB	RAM del servidor de prueba
max-conections	100	Máximo de conexiones del server de prueba
shared_buffers	24MB	shared_buffers del servidor de prueba

Tabla 11 Configuración para pruebas de estrés al Postgre SQL

Salidas de la primera prueba:

Parámetro	Valor	Descripción
-s	10	Equivalente a 1 000 000 de tuplas insertadas en tabla account.
-c	10	Cantidad de usuarios concurrentes.
-t	100	Transacciones por cada uno de los usuarios conectados concurrentemente.
ntp	10 000	Número de transacciones realizadas.
tps	606.32504	Número de transacciones por segundo
td	5 min	Tiempo de demora de ejecución de prueba.

Tabla 12 Prueba de Estrés al Postgre SQL (10 usuarios concurrentes)

Salidas de la segunda prueba:

Parámetro	Valor	Descripción
-s	10	Equivalente a 1 000 000 de tuplas insertadas en tabla account.
-c	90	Cantidad de usuarios concurrentes.
-t	10 000	Transacciones por cada uno de los usuarios conectados concurrentemente.
ntp	900 000	Número de transacciones realizadas.
tps	112.5226	Número de transacciones por segundo
td	1:13h	Tiempo de demora de ejecución de prueba.

Tabla 13Prueba de Estrés al Postgre SQL (90 usuarios concurrentes)

La realización de las pruebas de concepto siguiendo la estructura del expediente permitió agrupar las pruebas de forma más organizada, de manera que se puede comprender con mayor claridad donde se encuentras los riegos de la arquitectura.

Actividad 9 - Evaluar los Resultados de las Pruebas.

En esta actividad se generó un Informe de Revisión que recoge los resultados de las pruebas realizadas, organizadas y documentadas según la estructura del expediente, para facilitar la toma de decisiones.

La evaluación de la arquitectura en etapas tempranas y tardías permitió no solo la identificación de algunos de los principales riesgos de la arquitectura sino también demostró la correcta toma de decisiones en algunos casos.

Las pruebas de Aplicación fueron realizadas al componente de concurrencia del Framework a través del caso de estudio Gestionar Usuario, estas pruebas demuestran que el control de concurrencia impide que una persona modifique lo que ha sido recientemente modificado y brinda un mayor control a la hora de acceder a un recurso, lo que significa que el componente Concurrencia no constituye un riesgo para la arquitectura ya que cumple con el concepto para el cual fue creado.

Las pruebas de *Carga y Estrés* realizadas demuestran que el sistema no soporta más de 100 usuarios concurrentes realizando operaciones de adición, búsqueda, modificación o eliminación. Para 20, 50 y 100 usuarios concurrentes que interactúan con el sistema, ya sea para modificar, adicionar, eliminar, buscar, solicitar un recurso o acceder a la Interfaz Formato, el tiempo de respuesta de la aplicación es mucho mayor al tiempo esperado.

Los atributos de calidad Rendimiento y Disponibilidad constituyen riesgos para la arquitectura, ya que el sistema no soporta más de 100 usuarios concurrentes y el tiempo de respuesta de la aplicación en diversos casos de prueba es mucho mayor al tiempo esperado.

Las pruebas de Sistema realizadas a la decisión Multimoneda demuestran que el tiempo de respuesta de la aplicación es mucho mayor al tiempo esperado, reafirmando que los atributos de calidad Rendimiento y Disponibilidad constituyen riesgos para la arquitectura. Mientras que las pruebas de Datos demostraron que en el sistema no se puede eliminar o modificar un elemento si otro elemento depende de este, solo si se modifica o eliminan también los objetos dependientes a éste por lo que la integridad de los datos no constituye un riesgo para la arquitectura.

Para prevenir los fallos en la conexión en el sistema de réplicas, se comprobó que el servidor destino antes de recibir la actualización en los datos hace una sincronización de fechas con el servidor fuente lo cual le permite asegurarse que se está copiando la ultima actualización de los datos, demostrándose que se mantuvo la integridad de los datos, lo cual no constituye un riesgo para la arquitectura.

Las pruebas de estrés al servidor de datos para las condiciones fijadas demostraron que al aumentar el número de usuarios concurrentes, disminuyó el número de transacciones por segundo, tomándose como un riesgo, ya que trae como desventaja la disminución del rendimiento del sistema.

Actividad 10 - Efectuar la toma de decisiones.

Una vez identificados los riesgos presentes en la arquitectura es responsabilidad de los Tomadores de Decisiones e Involucrados de la arquitectura, a partir de los riesgos encontrados efectuar la toma de decisiones.

Algunas de las decisiones tomadas para mitigar los riesgos en la arquitectura de los atributos de calidad, Rendimiento y Disponibilidad son:

- Rendimiento
    - Desarrollo de una auditoria al diseño arquitectónico de datos y sistema.
    - Repetir las pruebas de conceptos.
    - Desarrollar componentes para el cacheo y optimización de la instanciación de recursos.
    - Definir una política de procesamiento en lote para la carga de datos, y manipulación de los mimos, limitar el lote de procesamiento a no más de 20 elementos por cada petición.
- Disponibilidad
    - Definir como requisitos mínimo de hardware para un escenario no mayor a 20 usuarios 1GB de RAM y 40 GB de disco duro, con escenarios de mayor consumo, incrementar los requerimientos tecnológicos y telemáticos para el despliegue de la solución.

3.3 Conclusiones.

El desarrollo de la guía práctica de actividades, para la realización de pruebas de concepto a la arquitectura, permitió descubrir los posibles puntos de riesgos de la arquitectura que se construía y las restricciones de la misma. A su vez permitió refinar y perfeccionar las decisiones arquitectónicas, contribuyó a la organización y a tener una mayor confianza en el producto que se construía, haciendo posible evitar la ocurrencia de costosos errores.

CONCLUSIONES

La elaboración del marco teórico demostró que existen técnicas y métodos de evaluación, pero la evaluación de la arquitectura y el estudio de Pruebas de Concepto (PoC) es aún un área inmadura en la industria del software.

La organización rectora en el tema de las pruebas de concepto es el SEI, institución que constituye una referencia importante para el estudio de evaluación de arquitecturas y la principal referencia de esta investigación.

Se desarrolló una Guía Práctica genérica para software de gestión, ya que la dimensión de las evaluaciones se enmarcan en puntos de interés para estos sistemas, a demás esta compuesta por un esquema de tres roles y diez actividades, donde cada rol tiene una responsabilidad determinada en la ejecución de cada una de las actividades que conforman la guía. La misma es iterativa y contiene elementos de retroceso, permitiéndose una mejor planificación de las actividades. Se definió un expediente que permitió organizar y documentar las pruebas de concepto realizadas, permitiendo un mejor entendimiento y gestión para futuras versiones de la arquitectura.

Con la aplicación de la Guía Práctica a un caso de estudio fue posible demostrar la funcionalidad e importancia de la utilización de la misma ya que se identificaron riesgos presentes en la arquitectura; contribuyéndose a mitigarlos, y las restricciones que presenta la misma, mejorando la toma de decisiones.

"El mayor beneficio que brinda la evaluación de una arquitectura, es que descubre los problemas que si se hubiesen dejado sin descubrir, habría sido mucho, más costoso corregirlos luego. En breve, una evaluación produce una mejor arquitectura." (8).

RECOMENDACIONES

Se recomienda que el resultado propuesto en la investigación sea implantado como uso cotidiano y sea introducido en todos los sistemas gestión de la universidad.

Se recomienda continuar la investigación sobre el resultado obtenido y proponerlo como base referencial o material auxiliar en el desarrollo de futuros trabajos asociados al objeto de estudio de Arquitectura de Software.

Se recomienda aplicar la Guía Práctica para evaluar la arquitectura del Sistema Integral de Gestión de Entidades, Cedrux a fin de lograr un producto con la calidad requerida.

REFERENCIAS BIBLIOGRÁFICAS

1. *Software Architecture in Practice.* **Bass, L., Clements, P., Kazman, R.** s.l. : Addison-Wesley, 2003, Vol. Segunda Edición.

2. **Hofmeister, C., Nord, R. y D, Soni.** *Applied Software Architecture.* s.l. : Addison, (2000).

3. **Kazman, R;Clements, P; Klein, M.** Evaluating Software Architectures. Methods and case studies. s.l. : Addison wesley, 2001.

4. **Kruchten, Philippe.** *The Rational Unified Process.* s.l. : Addison Wesley Longman, 2003.

5. **Barbacci, M., y otros.** *Quality Attributes.* Carnegie Mellon University. 1995. Technical Report. http://www.sei.cmu.edu/publications/documents/ 95.reports/95.tr.021.htm.

6. **Bosch, J.** Design & Use of Software Architectures. . s.l. : Addison-Wesley, 2000.

7. **ISO/IEC.** *Software Engineering – Software quality – General overview, reference models and guide to Software Product Quality Requirements and Evaluation (SQuaRE).* s.l. : Reporte. JTC1/SC7/WG6, 2002.

8. **Clements, Paul, Kazman, Rick y Klein, Mark.** *Evaluating Software Architectures.* s.l. : SEI. Series in Software Engineering. Adisson Wesley, 2002.

9. **Software Engineering Institute.** SEI. [En línea] [Citado el: 04 de diciembre de 2008.] http://www.sei.cmu.edu/architecture/evaluation_methods.html.

10. *Software Architecture Assesment.* **Trejo Vargas, Jorge Moisés y Icedo Ojeda, Rosa Virginia.** 2003.

11. Compusof S.A. [En línea] http://www.compusof.com/metodologia.asp#1.

12. HP Colombia. [En línea] [Citado el: 19 de Febrero de 2009.] http://www.hp.com/latam/co/servicios/aplicaciones_empresariales/enter_latencia_pruebaconc.html.

BIBLIOGRAFÍA

1. **Bosch, J.** Design & Use of Software Architectures. . s.l. : Addison-Wesley, 2000.

2. **Bass, L., Clements, P., Kazman, R.** *Software Architecture in Practice.*s.l. : Addison-Wesley, 2003, Vol. Segunda Edición.

3. **Hofmeister, C., Nord, R. y D, Soni.** *Applied Software Architecture.* s.l. : Addison, (2000).

4. **Kazman, R;Clements, P; Klein, M.** Evaluating Software Architectures. Methods and case studies. s.l. : Addison wesley, 2001.

5. **Kruchten, Philippe.** *The Rational Unified Process.* s.l. : Addison Wesley Longman, 2003.

6. **Barbacci, M., y otros.** *Quality Attributes.* Carnegie Mellon University. 1995. Technical Report. http://www.sei.cmu.edu/publications/documents/ 95.reports/95.tr.021.htm.

7. **ISO/IEC.** *Software Engineering – Software quality – General overview, reference models and guide to Software Product Quality Requirements and Evaluation (SQuaRE).* s.l. : Reporte. JTC1/SC7/WG6, 2002.

8. **Clements, Paul, Kazman, Rick y Klein, Mark.** *Evaluating Software Architectures.* s.l. : SEI. Series in Software Engineering. Adisson Wesley, 2002.

9. **Software Engineering Institute.** SEI. [En línea] [Citado el: 04 de diciembre de 2008.] http://www.sei.cmu.edu/architecture/evaluation_methods.html.

10. **Trejo Vargas, Jorge Moisés y Icedo Ojeda, Rosa Virginia** *Software Architecture Assesment.***.** 2003.

11. **Compusof S.A.** [En línea] http://www.compusof.com/metodologia.asp#1.

12. **HP Colombia**. [En línea] [Citado el: 19 de Febrero de 2009.] http://www.hp.com/latam/co/servicios/aplicaciones_empresariales/enter_latencia_pruebaconc.html.

13. **Sommerville, I.** *Software Engineering.* 7a. edición. 2005.

14. **Wiegers, K.** *Software Requirements,.* 1999.

15. **Windows Live.** Arquitectos vs Desarrolladores-Windows Live. [En línea] Microsoft, 2009. [Citado el: 5 de Enero de 2009.] http://diegumzone.spaces.live.com/blog/cns!1AD5096D63670065!290.entry.

16. **Carriere, J, Kazman, R y Woods, S.** Toward a Discipline of Scenariobased Architectural Engineering. [En línea] Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/staff/rkazman/annals-scenario.pdf.

17. **Shaw, M., Garlan D.** *Software Architecture Perspectives on a Emerging Discipline.* s.l. : Prentice.

BIBLIOGRAFÍA

18. **Erika Camacho, Fabio Cardeso,Gabriel Nuñez** *Arquitecturas de Software. Guía de estudio.***.** Abril 2004, pág. 4.

19. **Gregory Abowd, Georgia Institute of Technology, et al.** *Recommended Best Industrial Practice.* Carnegie Mellon University,Pittsburgh, Pennsylvania : SEI Joint Program Office, January 13, 2007.

20. **Dirección General de Servicios de Cómputo Académico-UNAM.** Revista digital universitaria. [En línea] 1 de Octubre de 2000. [Citado el: 3 de Marzo de 2009.] http://www.revista.unam.mx/vol.1/num2/art4/. Vol.1 No.2.

21. **Centro de Innovación de TI - Puebla.** Imnovacion Tecnológica. [En línea] [Citado el: 2009 de febrero de 6.] http://www.citip.org.mx/servicios/software/pruebaslab/Pages/PruebasConcepto.aspx.

22. **González, A., Mijares, M., Mendoza, L.E., Pérez, M., Grimán, A**.: Método de evaluación de arquitecturas de software basadas en componentes (MECABIC). VII Jornadas Internacionales de las Ciencias Computacionales, Colima, México (2005)

23. **Kitchenham, B**.: DESMET: A method for evaluating Software Engineering methods and tools, Technical Report TR96-09. Keele University (1999)

24. **Rubiolo, D., Meier, J., Jezierski, E., Mackman, A**.: Microsoft .NET Explained. Microsoft, 2001.

25. **Szyperski et al**., Component Software — Beyond Object-Oriented Programming, Tercera edición, Addison-Wesley/ACM Press (2003)

26. **Iannino, A.** "Software Reliability Theory." *Encyclopedia of Software Engineering,* New York, NY: Wiley, May 2005, 1237-1253.

27. **Leung, J.L.T. & Whitehead, J.** "On the Complexity of Fixed Priority Scheduling of Periodic, Real-Time Tasks." *Performance Evaluation*, *2,* 4 (2008) 237-250.

28. **Rajkumar, R.** *Synchronization in Real-Time Systems: A Priority Inheritance Approach*. Boston, MA: Kluwer Academic Publishers, 2006.

29. **Saaty, T.** *Fundamentals of Decision Making and Priority Theory With the Analytic Hierarchy Process.* Pittsburgh, PA: RWS Publications, 1999.

30. **Shaw, M. & Garlan, D.** *Software Architecture: Perspectives on an Emerging Discipline*. Upper Saddle River, NJ: Prentice-Hall, 2008.

31. **Wirfs-Brock, Rebecca; McKean, Alan.** Object Design Roles, Responsibilities, and Collaborations. Boston, MA: Addison-Wesley, 2003.

BIBLIOGRAFÍA

32. **Losavio F., Chirinos, L, Lévy, N., Ramdane-Cherif, A.** "Quality Characteristics for Software Architecture", Journal of Object Technology. Vol. 2, No. 2, (Marzo-Abril 2003), pp. 133-150. Disponible en http://www.jot.fm/issues/issue_2003_03/article2.

33. **Stocks and D. Carrington.** Test Templates: A Specification-Based Testing Framework. In

34. **Proceedings of the 15th International Conference on Software Engineering**, pages 405-414, Baltimore, MD, May 2002.

35. **Stafford, D. J. Richardson, and A. L. Wolf.** Chaining: A Software Architecture Dependence Analysis Technique. Technical Report CU-CS-845-97, University of Colorado, September 2006.

36. **Stafford, D. J. Richardson, and A. L. Wolf.** Aladdin: A Tool for Architecture-level

37. **Dependence Analysis of Software Systems.** University of Colorado Technical Report, CU-CS-858-98, 2008.

38. **E. Yourdon.** Software Quality Assurance in the 1990s. Technical report, Honeywell Technollogy Center, April 2003.

39. **H. Zhu, P. A. V. Hall, and H. R. J. May**. Software Unit Test Coverage and Adequacy. ACM Computing Surveys, 29(4), pages 366-427, December 2004.

ANEXOS

Anexo 1. Plantilla para descripción de Caso de Prueba.

1. Descripción general.

[Breve descripción del escenario]

2. Condiciones de Ejecución.

[Condiciones que deben existir para que se ejecute la acción]

3. Sección a probar en el componente.

Nombre del Componente.	**Escenarios de la sección.**	**Descripción de la funcionalidad.**	**Flujo Central.**
[nombre del componente]	[Nombre de la sección dentro del componente]	[Breve descripción de la acción que se realiza en el escenario]	[Descripción de las acciones a realizar en el escenario]

4. Datos de entrada.

No. Prueba	**Escenario**	**Variable 1** [Nombre de la variable]	**Variable 2** [Nombre de la variable]	**Variable n** [Nombre de la variable]	**Respuesta del Sistema**
< #>	[Nombre del escenario]	V([Datos de entrada])	V([Datos de entrada])	V([Datos de entrada])	[Se escribe el resultado que se obtiene del sistema]

Nota: V -> Los datos de entrada son válidos.

N/A-> Los entrada de datos no es obligatorio.

ANEXOS

Anexo 2. Plantilla para el Reporte de evaluación.

1. Propósito.

[Describir brevemente el propósito del documento.]

2. Escenarios de prueba.

[Describir brevemente el escenario sobre el que se realizo la prueba, agrupados por tipo de prueba.]

3. Resultados Obtenidos.

[Describir los resultados de las pruebas realizadas]

4. Conclusiones.

[Describir los principales puntos de riesgo de la arquitectura y las decisiones a tomar, teniendo en cuenta los riesgos identificados.]

Anexo 3. Plantillas de Prueba.

Decisión.

1. Propósito.

[Se precisa qué se busca probar, cuál es la observación que se va a realizar.]

2. Responsabilidad arquitectónica.

[Se especifican la responsabilidad que tiene la decisión en la arquitectura, es decir, si brinda servicio o depende de algún componente, si algún subsistema o componente depende de él, entre otras.]

3. Escenarios de prueba.

[Se especifican él o los escenarios que se utilizan para realizar la prueba, se agrupan en escenarios de Sistema, Datos e Integración.]

4. Recursos.

[Se indican qué recursos son empleados para realizar la prueba.]

5. Resultados.

[Se exponen los resultados que se obtuvieron al realizar la prueba.]

6. Conclusiones.

[En qué afectan o benefician los resultados obtenidos.]

7. Decisiones

[Definir las decisiones que se van a tomar, teniendo en cuenta los riesgos identificados.]

Cliente/Datos/Aplicación

1. Propósito.

[Se precisa qué se busca probar, cuál es la observación que se va a realizar.]

2. Descripción.

[Se describe brevemente en qué consiste la prueba.]

3. Recursos.

[Se indican qué recursos son empleados para realizar la prueba.]

4. Estados de entrada.

[Datos o condiciones con las que se inició la prueba.]

5. Resultados. Datos de salida.

[Se exponen los resultados que se obtuvieron al realizar la prueba.]

6. Indicadores de comparación.

[Establecer los indicadores con los que se comparan los resultados de las pruebas, es decir, el resultado esperado de la prueba.]

7. Conclusión.

[En qué afectan o benefician los resultados obtenidos.]

Anexo 4. Plantilla resumen

1. Propósito.

[Se describe la prueba que se realiza y el objetivo de las mismas.]

2. Escenarios de prueba.

[Describir brevemente el escenario sobre el que se realiza la prueba, en qué consiste.]

3. Recursos.

[Se indican qué recursos son empleados para realizar la prueba.]

4. Promedio de valores de las mediciones.

[Se expresan los resultados que se obtuvieron, en caso de que los resultados sean numéricos se expresan en término de promedios, en caso de que los resultados sean cualitativos, se expresan en forma de texto.]

5. Análisis de los indicadores obtenidos.

[Se especifican cuáles fueron los principales problemas detectados, que pruebas fueron más estables y dieron mejor resultado, que módulo/componente/subsistema/recurso se comportó mejor.]

6. Decisiones.

[Definir las decisiones que se van a tomar, teniendo en cuenta los riesgos identificados.]

Anexo 5. Plantilla para Escenarios.

1. Propósito.

[Se especifica el propósito del documento.]

2. Conceptos.

ANEXOS

[Se describen los conceptos importantes, que pueden ayudar a un mejor entendimiento de los escenarios.]

3. Escenarios.

[Se describen los escenarios y se agrupan de acuerdo al tipo de escenario y atributos de calidad/dimensiones de evaluación.]

Anexo 6. Plantilla para Método de evaluación.

1. Propósito

[Se especifica el objetivo de la aplicación del método de evaluación.]

2. Presentación del método.

[Se presenta un panorama general del método, pasos para efectuar la evaluación y característica generales del mismo.]

3. Presentación del negocio.

[Se presenta brevemente el negocio y el contexto de la arquitectura.]

4. Presentación de la arquitectura.

[Se presenta un panorama de la arquitectura, detallando los principales planteamientos, enfoques y estilos arquitectónicos.]

5. Árbol de Utilidad.

[Se identifican, priorizan, y refinan los atributos de calidad más importantes en un formato de árbol de utilidad.]

6. Análisis de los escenarios

[Describir los escenarios según la estructura de la siguiente tabla.]

Escenario:	<Se especifica el escenario de prueba>
Atributo:	<Atributo de calidad que evalúa el escenario

	(rendimiento, seguridad, otros).>		
Ambiente:	< Supuestos entorno sobre el que el sistema reside>		
Estímulo:	<Una descripción exacta del estímulo al atributo de calidad (por ejemplo, el fracaso, la amenaza, la modificación...) Por el escenario corporeizado>		
Respuesta:	<Una descripción exacta de la respuesta del atributo de calidad (por ejemplo, tiempo de respuesta, medida de dificultad de cambio)>		
Decisiones de la arquitectura	Riesgo	Puntos de sensibilidad	Puntos de desventaja
<listado de las decisiones arquitectónica que afectan la respuesta de los atributo de calidad >	<#Riesgo >	< # Punto sensible>	<# Desventaja>
Razonamiento:	< Justificación cualitativa y / o cuantitativa de por qué la lista de las decisiones arquitectónicas contribuyen al cumplimiento de los requisitos de calidad atributo respuesta>		
Diagrama de la arquitectura	<Diagrama o diagramas de puntos de vista con anotada arquitectónico arquitectura de información para apoyar el razonamiento anterior. Estos podrán ir acompañadas de textos explicativos.>		

7. Conclusiones

[Se concluye con los principales puntos de riesgo de la arquitectura.]

ANEXOS

Anexo 7. Plantilla para Descripción de herramientas.

1. Propósito.

[Se especifica el propósito y la importancia del uso de herramientas en la realización de las pruebas, cuál es la observación que se realiza.]

2. Descripción de las herramientas.

[Se describen las herramientas que se utilizan para la realización de las pruebas y cómo trabajar con ellas.]

3. Conclusión.

[Se especifica la importancia, el beneficio que aporta el uso de herramientas a su trabajo, es decir, los resultados obtenidos como consecuencia del uso de herramientas.]

4. Anexos.

[Fotos, tablas y otros a los que haya hecho referencia en el documento.]

Anexo 8. Encuesta.

1. ¿Sabe usted en qué consiste una evaluación de la arquitectura?

   Si__ No__

2. ¿Se realiza en su proyecto alguna evaluación de la arquitectura?

   Si__ No__

3. ¿Conoce usted qué es una Prueba de Concepto a la arquitectura?

   Si__ No__

4. ¿Realizan en su proyecto alguna Prueba de Concepto a la arquitectura?

   Si__ No__

5. ¿Conoce algún proyecto que realice Pruebas de Concepto?

   Si__ No__