

**ADMINISTRACIÓN DE DECISIONES DE ARQUITECTURA DE SOFTWARE
USANDO WIKI SEMÁNTICA**

**GABRIEL MEJÍA COELLO
DAVID PULIDO**

**UNIVERSIDAD PILOTO DE COLOMBIA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ D.C.
2016**

**ADMINISTRACIÓN DE DECISIONES DE ARQUITECTURA DE SOFTWARE
USANDO WIKI SEMÁNTICA**

**GABRIEL MEJÍA COELLO
1019022364
DAVID PULIDO
820683**

Proyecto de grado para optar al título de Ingeniero de Sistemas

**Director:
GILBERTO PEDRAZA GRACIA
Ingeniero de Sistemas y Computación**

**UNIVERSIDAD PILOTO DE COLOMBIA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ D.C.**

2016

Nota de aceptación

Firma presidente del jurado

Firma del jurado

Firma del jurado

Bogotá, D.C., Diciembre de 2016

AGRADECIMIENTOS

Agradecemos a todos los profesores de la Universidad Piloto de Colombia que hicieron parte de nuestra formación académica y que con su dedicación y esfuerzo lograron inspirarnos para seguir adelante. Un particular agradecimiento al profesor Ignacio Hernández Molina y profesor Gilberto Pedraza García por su asistencia y guía en el desarrollo del presente trabajo de grado al igual que en toda la carrera. Agradecemos a los profesores Daniel Alfonso Bojacá Torres, Giovanni Fajardo Utria, Jaime Alberto Chavarriaga Lozano, Ibo Luis Cerra Escobar, Luz Karina Sabogal Bohórquez, Juan Carlos Navarro Beltrán y Fidel Barboza Gutiérrez por ser guías y amigos.

Agradecemos a nuestras familias y amigos que nos apoyaron a lo largo de nuestra carrera, a ellos en particular se le dedica este proyecto de grado.

Agradecemos a He-Man y a los Amos del Universo su constante esfuerzo en contra de los intentos de Skeletor de dominar los misteriosos poderes del Castillo de Greyskull de manera que no pueda conquistar Eternia. Agradecemos a She-Ra que al igual que su hermano combaten por la justicia y la paz en el universo, unidos ni Skeletor, ni Hordak, ni siquiera el mismísimo Horde Prime podrán conquistar el universo. De manera similar y muy afectuosa se agradece a Zenki, Doraemon, a los Thunder Cats, los Halcones Galácticos, las Tortugas Ninjas Mutantes Adolescentes, los Street Sharks, a los Gatos Samurai, los Defensores de la Tierra, el Capitán Centella y Mazinger Z, todos a su manera hacen de este mundo un lugar mejor.

CONTENIDO

	Pág.
INTRODUCCIÓN	13
1. PLANTEAMIENTO DEL PROBLEMA	15
1.1 DESCRIPCIÓN DEL PROBLEMA	15
1.2 JUSTIFICACIÓN	¡Error! Marcador no definido.
1.3 ALCANCES	¡Error! Marcador no definido.
1.4 LÍMITES	¡Error! Marcador no definido.
1.5 OBJETIVOS	17
1.5.1 Objetivo general	17
1.5.2 Objetivos Específicos	17
2. MARCO TEÓRICO	18
2.1 ARQUITECTURA DE SOFTWARE	18
2.2 DECISIONES DE DISEÑO DE ARQUITECTURA	19
2.2.1 Decisiones de existencia (ontocríticas)	20
2.3 ADMINISTRACIÓN DEL CONOCIMIENTO	22
2.4 WEB SEMÁNTICA	24
2.5 ONTOLOGÍAS	¡Error! Marcador no definido.

3. DISEÑO METODOLÓGICO DE LA INVESTIGACIÓN	27
3.1 SISTEMA DE HIPÓTESIS	27
3.1.1 Hipótesis de trabajo	27
3.1.1.1 Hipótesis nula	27
3.2 SISTEMA DE VARIABLES	27
3.2.1 Variables independientes	27
3.2.2 Variables dependientes	27
3.2.3 Variables intervinientes	27
3.3 ANÁLISIS DEL PROBLEMA	27
3.4 TÉCNICAS DE LA RECOLECCIÓN DE LA INFORMACIÓN	28
3.5 METODOLOGÍA DE DESARROLLO	29
4. DESARROLLO METODOLÓGICO DE LA INVESTIGACIÓN	31
4.1 MODELO DE RAZÓN DE DECISIONES	31
5. IMPLEMENTACIÓN DE LA SOLUCIÓN	37
5.1 DESCRIPCIÓN DE LOS CASOS DE USO	37
5.2 TECNOLOGÍAS Y PATRONES UTILIZADOS	45
5.3 IMPLEMENTACIÓN DE LA APLICACIÓN	47
5.3.1 Implementación de los controladores y el modelo	47
5.3.2. Implementación de las vistas	51
5.4 DISEÑO DE INTERFACES	52
5.5 PRUEBAS	¡Error! Marcador no definido.

6. VALIDACIÓN	68
6.1 OBJETIVOS, HIPÓTESIS Y VARIABLES	68
6.2 DISEÑO EXPERIMENTAL	71
6.3 EJECUCIÓN	73
6.3.1 Preparación del experimento	73
6.3.2 Ejecución del experimento	74
7. CONCLUSIONES Y TRABAJO FUTURO	75
BIBLIOGRAFÍA	74
ANEXOS	78

LISTA DE FIGURAS

	pág.
Figura 1. Esquema del ciclo de vida de la administración del conocimiento	24
Figura 2. Modelo de decisiones	31
Figura 3. Primera parte del diseño de la ontología que representa el modelo de decisiones	33
Figura 4. Segunda parte del diseño de la ontología, se adapta la ontología para aplicar ciertas preocupaciones de seguridad.	35
Figura 5. Diagrama de casos de uso de la wiki semántica	45
Figura 6. Diagrama de componentes	48
Figura 7. Vista de información del componente Process Alternative Transaction	49
Figura 8. Vista de información del componente Process Decisión Transaction	50
Figura 9. Diagrama de secuencias general de la aplicación.	51
Figura 10. Pantalla de login	52
Figura 11. Pantalla principal de la aplicación	52
Figura 12. Resultado de una consulta general.	53
Figura 13. Módulo de consulta.	53
Figura 14. Módulo de subconsultas	54
Figura 15. Mapas de la ontología.	54
Figura 16. Diagrama de contexto del objeto de estudio	73

LISTA DE TABLAS

	pág.
Tabla 1. Modelo de desarrollo pensado para el desarrollo de la wiki semántica	29
Tabla 2. Caso de uso Consultar individuos a través de coincidencias textuales.	37
Tabla 3. Caso de uso Consultar todos los cambios realizados sobre individuos.	38
Tabla 4. Caso de uso Consultar individuo.	39
Tabla 5. Caso de uso Editar contenido de un individuo.	40
Tabla 6. Caso de uso Cargar imágenes de Disco.	41
Tabla 7. Caso de uso Cargar imágenes por URL.	42
Tabla 8. Caso de uso crear cargar imagen existente.	43
Tabla 9. Caso de uso Consultar individuo relacionado a otro.	44
Tabla 10. Caso de prueba CP-01	55
Tabla 11. Caso de prueba CP-02	56
Tabla 12. Caso de prueba CP-03	57
Tabla 13. Caso de prueba CP-04	59
Tabla 14. Caso de prueba CP-05	60
Tabla 15. Caso de prueba CP-06	62
Tabla 16. Caso de prueba CP-07	65
Tabla 17. Caso de prueba CP-08	67
Tabla 18. Diseño factorial 2x2 para uso de la aplicación.	69
Tabla 19. Diseño factorial 2x2 son uso de la aplicación.	69

GLOSARIO

ADMINISTRACIÓN DEL CONOCIMIENTO: es un modelo que describe lo que se debe hacer para comprender los conceptos o información inherente a un proceso u objeto de conocimiento, usualmente se consideran la captura, la socialización y recuperación de información como las actividades dentro de un modelo básico.¹

ARQUITECTURA DE SOFTWARE: es un conjunto de decisiones que los arquitectos de software deben tomar para satisfacer las necesidades de los stakeholders².

DECISIONES DE ARQUITECTURA DE SOFTWARE: son una explicación detallada acerca de lo que se le debe añadir o sustraer a una arquitectura de software, junto con su correspondiente justificación, reglas y restricciones que rigen su desarrollo³.

ONTOLOGÍA: es un constructo ingenieril que consiste en un esquema de conceptos y las relaciones entre los mismos para describir arduamente el dominio de un problema⁴.

REQUERIMIENTO FUNCIONAL: es una descripción de una necesidad de un stakeholder.⁵

REQUERIMIENTO NO FUNCIONAL: es una descripción de una condición o restricción que debe tomarse en cuenta en el proceso de desarrollo, inherente a los intereses del stakeholder.⁶

STAKEHOLDERS: son las personas que poseen el deseo de otorgar una solución informática a uno de sus problemas y contratan a especialistas o casas creadoras de software para ello⁷.

¹ MONOGRAFIAS. Administración del conocimiento. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: www.monografias.com › Administración y Finanzas › Recursos Humanos

² SG BUZZ. Arquitectura del software. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://sg.com.mx/revista/27/arquitectura-software>

³ BARRAZA, Fernando. Decisiones de arquitectura de software. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://prezi.com/shgc48bzwo0u/decisiones-en-el-diseno-arquitectonico/>

⁴ SIGNIFICADOS.COM. Qué significa Intología. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://www.significados.com/ontologia/>

⁵ GIRALDO, Isabel. ¿Qué son requerimientos funcionales. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://prezi.com/t7fmr4mkgwym/requerimientos-funcionales>

⁶ GIRALDO, Isabel. ¿Qué son requerimientos no funcionales. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://prezi.com/t7fmr4mkgwym/requerimientos-no-funcionales>

WEB SEMÁNTICA: es una actualización de la “web tradicional” en donde los contenidos de la misma se encuentran categorizados otorgándoles significado, siendo posible de esta manera realizar consultas a un nivel de especificación más flexible y detallado⁸.

WIKI: es un sitio web donde el contenido de la misma puede ser añadido, modificado e incluso eliminado por los usuarios utilizando su navegador⁹.

WIKI SEMÁNTICA: es una wiki en donde se implementa un enfoque de administración de conocimiento (usualmente una ontología). Wiki dotada de significado, en donde cada una de sus páginas (conceptos o entidades) se relacionan de una manera más descriptiva entre una y otra¹⁰.

⁷ GUIOTECA. ¿Qué son los stakeholders?. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://www.guioteca.com/rse/que-son-los-stakeholders/>

⁸ W3C.ES. ¿Qué es WEB Semántica? [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: www.w3c.es › Documentos y Guías › Guías Breves

⁹ PÉREZ, Isabel. ¿Qué es Wiki?. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: www.isabelperez.com/taller1/wiki.htm

¹⁰ DIALNET. ¿Qué es wiki semántica? ?. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://dialnet.unirioja.es/descarga/articulo/2924515.pdf>

RESUMEN

El presente documento tiene como propósito presentar una propuesta de modelo conocimiento para decisiones de arquitecturas de software, adaptar una ontología que se ajuste al modelo de conocimiento propuesto y una wiki funcional que servirá de interfaz entre el usuario y la ontología en sí misma a fin de brindar una alternativa a los métodos tradicionales de recuperación, representación y difusión de las decisiones tomadas, una alternativa dedicada dejando de lado la ideas de las páginas en las wiki convencionales y de las anotaciones que brindan semántica en extensiones como semantic wiki.

Palabras claves: modelo de conocimiento. Arquitecturas de software, ontología

INTRODUCCIÓN

En un proceso de creación de alguna arquitectura de software usualmente lo que es considerado importante son el conjunto de artefactos a desarrollar y sus vistas¹¹, pero esto no es lo único importante. Una interrogante que muchos arquitectos, o diseñadores de software en general que desean continuar con el trabajo de otros, se formulan es ¿por qué se tomaron esas decisiones? Es entonces cuando estos arquitectos o diseñadores no conocen los trasfondos de rendimiento, funcionalidad, de reutilización, de inteligibilidad, económicos y tecnológicos¹² que su predecesor consideró para tomar esas decisiones¹³. Entonces debe uno preocuparse hasta qué punto se vería comprometida la eficiencia y la eficacia de la mantenibilidad de una arquitectura de software si únicamente se conoce su resultado y no lo que la produjo.

Pues conocer la existencia de la relación de causa y efecto entre lo diseñado o producido en la arquitectura y aquello que lo impulsó puede ser definitorio en la toma de nuevas decisiones sobre la arquitectura, lo anterior nos lleva a considerar las decisiones de arquitectura de software como conocimiento, producto de la experiencia y la experticia¹⁴, lo que lleva a la suposición de que la correcta administración de dicho conocimiento puede llegar a ser el motor la productividad y la eficacia¹⁵ en el proceso, además de ayudar, incluso, a los diseñadores o arquitectos menos experimentados a tomar decisiones de calidad.

Por todo lo anterior el presente trabajo de grado tiene como propósito presentar una propuesta de modelo conocimiento para decisiones de arquitecturas de software, adaptar una ontología que se ajuste al modelo de conocimiento propuesto y una wiki funcional que servirá de interfaz entre el usuario y la ontología en sí misma a fin de brindar una alternativa a los métodos tradicionales de recuperación, representación y difusión de las decisiones tomadas, una

¹¹ KRUCHTEN, . Philippe . An Ontology of Architectural Design Decisions in software-Intensive System. 2004, Octubre, [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: <https://philippe.kruchten.com/architecture/>

¹² Ibid ., p.2

¹³ TYREE, Jeff y AKERMAN Art. Architecture Decisions: Demystifying Architecture. 20 p. , [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: https://www.researchgate.net/.../245352502_An_Ontology_of_Architectural_Design_

¹⁴ WILEY, John y SONS, The Complete Guide to Knowledge Management A Strategic Plan to Leverage Your Company's Intellectual Capital, p. 10-17. [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: [apollon1.alba.edu.gr/OKLC2002/Proceedings/pdf.../ID138 .pdf](http://apollon1.alba.edu.gr/OKLC2002/Proceedings/pdf.../ID138.pdf)

¹⁵ DALAIR, Knowledge Management in Theory and Practice, CA: Elsevier Inc, pp. 10-25. [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: <https://mitpress.mit.edu/.../knowledge-management-theory-and-pr>.

alternativa dedicada dejando de lado la ideas de las páginas en las wiki convencionales y de las anotaciones que brindan semántica en extensiones como semantic wiki.

1. PLANTEAMIENTO DEL PROBLEMA

1.1 DESCRIPCIÓN DEL PROBLEMA

El enfoque actual de arquitectura de software la define como un conjunto de decisiones que se toman con el fin de balancear las necesidades de los stakeholders (interesados). Los aquellos involucrados en el desarrollo de una arquitectura de software, casi que por regla generalizada suelen documentar las decisiones de arquitectura tomadas de una manera tradicional (escrita, impresa), algunos otros disponen de ciertas herramientas como wikis para registrar dichas decisiones, wikis que utilizan una ontología base generalizada para la captura y representación de las mismas. Lo anterior puede causar que en etapas de mantenimiento se pueda perder el hilo de comprensión de las decisiones de arquitectura de software, en especial cuando el que intenta comprender dichas decisiones no estuvo involucrado en el desarrollo de la arquitectura como tal (nuevos arquitectos en una nómina, por ejemplo), lo cual podría implicar que no se tomen decisiones de calidad sobre lo cual se está analizando o comprendiendo.

1.2 JUSTIFICACIÓN

Las empresas que ofrecen servicios orientados a tecnologías (específicamente las empresas productoras de software), tienen que lidiar con necesidades varias del cliente, como son las aplicaciones móviles que representan amplias oportunidades para el desarrollo de soluciones corporativas¹⁶. Por lo tanto, la agilización en la producción es un detalle del cual no se debe obviar.

Las organizaciones son sistemas de actividades conscientes coordinadas, es decir son sistemas cooperativos, no son productos mecánicos del trabajo de ingenieros de eficiencia¹⁷. En estas circunstancias la comunicación juega un papel determinante. En el caso de las empresas diseñadoras de software la comunicación no solo se da verbalmente, los modelos que se generan dentro de todo el proceso de desarrollo que se sigue, desde que se conoce el cliente hasta que se entrega el producto terminado y posteriormente se mantiene, son un canal comunicativo importante, ya que estos representan gráficamente los requisitos del usuario y son el producto de una actividad de toma de decisiones. Cada modelo, cada alternativa y en general todo aquello que se documenta en un proceso de desarrollo de software constituye información importante que debe ser gestionada adecuadamente, es el registro histórico de los aciertos y los fracasos en este

¹⁶ PROEXPORT. Soluciones corporativas. 2009. [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: www.talent.upc.edu/media/SolucionesCorporatives_esp.pdf

¹⁷ BARNARD. Sociología de las organizaciones. 1938, [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: datateca.unad.edu.co/contenidos/.../libro_libre_Sociologia_de_las_organizaciones.pdf

proceso y recordarlos es vital para evitar cometer errores futuros y mejorar por ende la productividad.

Las empresas desarrolladoras de software que realizan un trabajo “más o menos serio” en sus productos aplican diferentes técnicas y metodologías en sus procesos de desarrollo y se esfuerzan en mantener la correcta ejecución de los mismos. En ese tipo de empresas no es extraña la aparición de ciertos grupos de personas que se encargan de balancear las necesidades de los clientes (stakeholders) en el producto por desarrollar, personas que se dedican a identificar que es, a rasgos macros (omiten detalles técnicos y de implementación, por ejemplo), lo importante en el producto solicitado¹⁸. Los arquitectos en última instancia deciden que es lo que se realizará y cómo (a grandes rasgos) se realizará teniendo en cuenta diferentes variables como el presupuesto, tiempo, complejidad, entre otras. La totalidad de las decisiones tomadas constituye la arquitectura de software.

Las decisiones reflejan los concerns de los stakeholders, reflejan las restricciones y condiciones planteadas porque se tomaron por alguna razón. Las razones de las decisiones representan aquella información que permite identificar las variables involucradas en la elección de la decisión permitiendo pasar de lo individual a lo global y así entender cómo estas interactúan con su entorno (concerns, atributos de calidad considerados, responsables de la decisión, etc). Lo anterior no poco ya que permite a nuevos arquitectos que no estuvieron involucrados en las etapas iniciales de desarrollo, comprender mejor lo que se hizo y porqué se hizo para poder posteriormente tomar decisiones de mejor calidad sobre la misma arquitectura, reduciendo tiempos y costos a la empresa.

1.3 ALCANCES

Desarrollar un modelo de conocimiento, una ontología y wiki funcional sobre las decisiones de arquitectura de software el cual se validará con un equipo de arquitectos de un proyecto.

1.4 LÍMITES

Se aplicará la perspectiva de seguridad de Rozanski en el desarrollo de la wiki, más específicamente los conceptos de autorización, trazabilidad y recuperación frente a fallos. La Wiki solo servirá como medio que servirá para la representación, recuperación y difusión de decisiones de arquitectura de software, no se incluirá el proceso de captura.

¹⁸ FOWLER Martin. ¿Who Needs an Architect?, pp 2-4. [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: martinfowler.com/ieeeSoftware/whoNeedsArchitect.p

1.5 OBJETIVOS

1.5.1 Objetivo general. Desarrollar una estrategia para recuperar, representar, difundir y reutilizar las decisiones que toman los equipos de arquitectura en el diseño de la misma.

1.5.2 Objetivos Específicos

- Diseñar un modelo de conocimiento que represente las decisiones de arquitecturas de software y las variables más importantes que se ven involucradas en ellas.
- Diseñar una ontología para representar decisiones de arquitectura y que se ajuste al modelo de conocimiento que se propondrá.
- Construir una wiki para recuperar, representar, difundir y reutilizar las decisiones de arquitectura.
- Aplicar la wiki con el caso de estudio: PUA.
- Validar la wiki con un conjunto de arquitectos.

2. MARCO TEÓRICO

2.1 ARQUITECTURA DE SOFTWARE

Según el estándar IEEE 1471¹⁹, una arquitectura de software es una organización fundamental de un sistema de software formado por sus componentes, relaciones entre ellos y su entorno y unos principios que rigen su diseño y evolución. Sin embargo, para desarrollar una arquitectura de software es necesario tomar en cuenta las decisiones tomadas y el proceso que se llevó a cabo para llegar a ellas. Es en este contexto entonces que se considerará la definición dada por RUP (Rational Unified Process)²⁰ que dice que una arquitectura de software es un conjunto de decisiones de diseño acerca de la organización de un sistema de software. Una arquitectura de software también incluye la funcionabilidad, el rendimiento, la reutilización, la inteligibilidad, las condiciones económicas y tecnológicas e intereses estéticos²¹.

Existen dos formas básicas de describir arquitecturas de software:

- Usando vistas²², representaciones gráficas bajo un estándar dado. Uno de los estándares difundidos y utilizados en el desarrollo de las vistas el IEEE 1471²³.
- Usando un lenguaje de descripción de arquitecturas.

Las arquitecturas no son creadas en un proceso solitario, es un trabajo colaborativo y dinámico entre los stakeholders y los arquitectos, en donde los requerimientos asumen un papel no trivial en la construcción del conocimiento de la arquitectura²⁴, siendo esto último lo documentable, lo que se necesita conocer y entender.

¹⁹ KRUCHTEN, Philippe. IEEE ORG.1471-2000 . IEEE Recommended Practice for Architectural Description for Software-Intensive Systems: An Ontology of Architectural Design Decisions in software-Intensive Systems. p.1. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: <http://standards.ieee.org/findstds/standard/1471-2000.html>

²⁰ IBM citado por Kruchten Philippe. IBM.Rational Unified Process: Best Practices for Software Development Teams: An Ontology of Architectural Design Decisions in software-Intensive Systems. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

²¹ KRUCHTEN, Philippe. Op. cit. p. 13

²² Ibid., p. 1.

²³ Ibid., p. 1.

²⁴ VLIET Hans y BOER, Remco. Experiences with Semantic Wikis for Architectural Knowledge Management. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: ieeexplore.ieee.org/abstract/document/5959696/?section=abstract

Como se considerará una arquitectura de software como un conjunto de decisiones acerca de la organización de un sistema de software, estas son las constituyentes primordiales del conocimiento de dicha arquitectura, y por tanto es requerido saber que es una decisión de diseño de arquitectura, que tipos de decisiones podemos encontrar y qué consideraciones debemos tener en cuenta al momento de tomar una decisión.

2.2 DECISIONES DE DISEÑO DE ARQUITECTURA

Las decisiones de arquitectura resuelven problemas particulares que se encuentran en un proceso de creación de arquitectura²⁵, estas implican en cierta manera adiciones, sustracciones y modificaciones a una arquitectura de software, lo que lleva a un proceso de racionalización e identificación de reglas y restricciones de diseño y los requerimientos adicionales²⁶.

La racionalización hace referencia a las razones que están detrás de una decisión de diseño de arquitectura, estas explican porque se ha hecho un cambio a una arquitectura de software. Las reglas de diseño describen que está permitido realizarse cuando se esté desarrollando el diseño. Las restricciones son las contrarias a las reglas, estas nos dicen que no podemos realizar en un futuro, es decir prohíben ciertos comportamientos. Por último, cuando se esté en proceso de toma de decisiones, usualmente se presentan requerimientos adicionales que deben ser satisfechos y añadidos²⁷.

Se supondrá que un arquitecto entra en la nómina de una empresa, con la labor de continuar el proceso de diseño de una arquitectura de software para un sistema contable. En estas circunstancias deberá familiarizarse con el trabajo que su par dejó. Es muy común que suceda que el nuevo arquitecto vea, con cierta dificultad e incluso incredulidad, las razones de por qué se decidieron en construir un API intermedio para el control del proceso de transacciones entre el cliente y la empresa (siendo esto una suposición) cuando podría llegar a ser más económico y con mayor soporte la compra a una casa generadora. Resulta que las decisiones tomadas son circunstanciales, es decir, únicamente tienen sentido en las condiciones en que se toman²⁸.

²⁵ PEDRAZA-GARCIA, Gilberto, Astudillo, Hernán and Correal, Darío. Modeling Software Architecture Process with a Decision-Making Approach. 2014. 33rd International Conference of the Chilean Computer Science Society (SCCC), Talca, 2014, pp. 1-6. doi: 10.1109/SCCC.2014.27.

²⁶ BOSCH Jan, JANSEN Anton. Software Architecture as a Set of Architectural Design Decisions. p.2. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: www.ics.uci.edu/~andre/ics223w2006/jansenbosch.pdf

²⁷ Ibid., p.2

²⁸ TYREE Jeff, AKERMAN Art. Architecture Decisions: Demystifying Architecture. 20 p. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: https://www.utdallas.edu/.../zz-impreso-architecture_decisions-tyr..

Sin embargo, no todos los diseñadores ven a la arquitectura de software como un conjunto de decisiones que se deben tomar para satisfacer las necesidades de los stakeholders, como se puede notar en el estándar IEEE 1471²⁹. En este caso la arquitectura es vista, más bien como un producto y cuando esto ocurre, y no se le otorga la suficiente importancia a la fundamentación del conjunto de decisiones tomadas en el proceso de desarrollo de dicho producto, se pueden presentar problemas como los que Bosch identificó:

Violación de las decisiones tomadas en etapas previas en el proceso de desarrollo. En la etapa de evolución del software (especialmente en esta), suelen violar reglas y restricciones de diseño que se consideraron cuando se seguía la etapa de desarrollo. La violación de estas reglas y restricciones acarrea problemas (costo de mantenimiento), asumiendo el hecho de que tanto las reglas como las restricciones de diseño especifican lineamientos, tanto como para el presente como para el futuro del proceso de diseño.

- Algunas decisiones obsoletas pueden no ser removidas. Esto puede causar que el sistema se “corroa” más rápidamente.
- Toma de diversas decisiones “excesivamente relacionadas”. En el proceso de diseño de decisiones pueden existir algunas que presenten una relación íntima. Pero cuando la relación se lleva fuera de “proporciones normales” se presentarán situaciones en donde fuere difícil encontrar cambios en el diseño³⁰.
- Los tipos de decisiones de diseño pueden variar de autor en autor, no existe una aceptación generalizada para cada una de ellas. Pero en orden de entender porque es necesario conocerlas se considerará las identificadas por Bosh:

2.2.1 Decisiones de existencia (ontocríticas). Son decisiones de estructura y de comportamiento. La primera lidera la creación de subsistemas, capas, particiones y componentes en una vista de la arquitectura. Las decisiones de comportamiento son las relacionadas a la forma en cómo los elementos de la arquitectura interactúan entre sí, para proveer una funcionalidad que satisface un requerimiento funcional y se limiten según los requerimientos no funcionales. Por ejemplo:

- La vista lógica está organizada en tres capas: Capa de Datos, Capa de inteligencia de Negocios y la Capa de Interfaz de Usuario.

²⁹ KRUCHTEN, Op.Cit., p.16

³⁰ BOSCH y JANSEN, Op.Cit, p. 17

- La comunicación entre las clases usa RMI (Remote Method Invocation).

Este tipo de decisiones no son tan importantes de capturar ya que ellas están usualmente visibles en el diseño o implementación del sistema, y las razones por las cuales se tomaron estas decisiones se encontrarán descritas en la documentación desarrollada.

– Decisiones de Propiedad (diacríticas). Son decisiones relacionadas a las reglas de diseño y restricciones de diseño. Las decisiones de propiedad son difíciles de encontrar, ya que estas suelen afectar a muchos elementos del sistema y otras pueden ser implícitas siendo estas olvidadas fácilmente.

– Decisiones ejecutivas (pericríticas). Son decisiones relacionadas con el ambiente de negocio y afecta el proceso de desarrollo (metodologías), las personas (educación y entrenamiento), la organización y a una larga extensión de elecciones de tecnologías y herramientas. Por ejemplo:

- Todos los cambios en subsistemas deben ser aprobados por el equipo de arquitectura.
- El sistema es desarrollado usando J2EE.
- El sistema es desarrollado en Java.
- El sistema es desarrollado usando Eclipse.

Este tipo de decisiones están íntimamente relacionadas con todos los aspectos políticos, personales, culturales, financieros y tecnológicos que generan grandes cantidades de restricciones y, todas las decisiones asociadas difícilmente (por regla general) se capturan o documentan³¹.

Es razonable o lógico pensar que cuanto más grande y complejo sea el sistema informático a construir, un mayor número de decisiones se deberán tomar, y cuando mayor sea el número de restricciones y condiciones que se tengan, más *críticas* se vuelven estas decisiones. Es aquí en dónde radica la importancia de la buena captura y comprensión de estas para evitar problemas futuros de mantenimiento y escala. Estas decisiones deben ser socializadas no solo con el equipo de desarrollo o entre los arquitectos, sino que también entre los stakeholders. Estos últimos son los más importantes en vista de que son los inversionistas del trabajo desarrollado. Entre los métodos más utilizados para la

³¹ KRUCHTEN, Op.Cit., p.2.

exposición o socialización de las decisiones tomadas, pero no implementadas aun, se puede nombrar la clásica documentación y las presentaciones en Microsoft Power Point, sin embargo trabajos como los de Kruchten o los de Tyree y Akerman, exponen, entre otras cosas, las importantes ventajas que las ontologías pueden llegar a tener en la administración del conocimiento de las arquitecturas, aunque las webs semánticas, por su naturaleza, puede proveer de un entorno apropiado para la captura, socialización, recuperación y por tanto comprensión de los conocimientos que podamos obtener de una arquitectura de software determinada.

2.3 ADMINISTRACIÓN DEL CONOCIMIENTO

El conocimiento es algo que para cualquier empresa le es de gran utilidad, el conocimiento es experiencia, experticia que al unirse con la interpretación, razonamiento y, en general, procesamiento de información³² pueden generar ganancia, capital intelectual³³. Entonces el conocimiento es un flujo de experiencias, valores de facto, información contextualizada y visión de expertos que ofrece un espacio de trabajo para la evaluación y la incorporación de nuevas experiencias e información³⁴. Pare efectos del presente proyecto de grado se considerará como conocimiento las razones que justifican el diseño de software y permiten tomar decisiones correctas.

El conocimiento como generador de capital intelectual necesita recuperarse, reutilizarse, no debe ser cosa de una o dos veces. El conocimiento es la memoria de las experiencias pasadas que se traen al presente con el objeto de mejorar la eficiencia de la empresa³⁵. La administración del conocimiento ha sido expuesta en un principio como una aproximación sistemática para la captura, recuperación, representación y difusión del conocimiento como lo muestra la figura 1, todo en aras de la eficiencia, el reuso de las mejores prácticas y la disminución del trabajo repetitivo entre proyecto y proyecto (Organizational Learning)³⁶ enfocándose en la generación de capital intelectual³⁷.

³² GONZÁLEZ, Oscar et al. MONO+KM: Knowledge Management in Collaborative Project Development. Ingeniería y Universidad: Engineering for Development, [S.I.], v. 20, n. 2, p. 267-302, jun. 2016. ISSN 2011-2769. Available at: <<http://revistas.javeriana.edu.co/index.php/iyu/article/view/14862>>. Date accessed: 15 Dec. 2016. doi:<http://dx.doi.org/10.11144/Javeriana.iyu20-2.mkkm>.

³³ RONEN, P. Edna., The Complete Guide to Knowledge Management A Strategic Plan to Leverage Your Company's Intellectual Capital, USA: John Wiley & Sons, pp. 10-17.

³⁴ MURRAY. J. Knowledge Management in Modern Organizations. San Diego State University, USA .. pp 2.

³⁵ *Ibíd.*, p 3.

³⁶ DALKIR, J. Knowledge Management in Theory and Practice, CA: Elsevier Inc, p. 10. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: [25.https://dianabarbosa.files.wordpress.com/.../knowledge-management-kimiz-dalkir.pdf](https://dianabarbosa.files.wordpress.com/.../knowledge-management-kimiz-dalkir.pdf)

³⁷ *Ibíd.*, p 20.

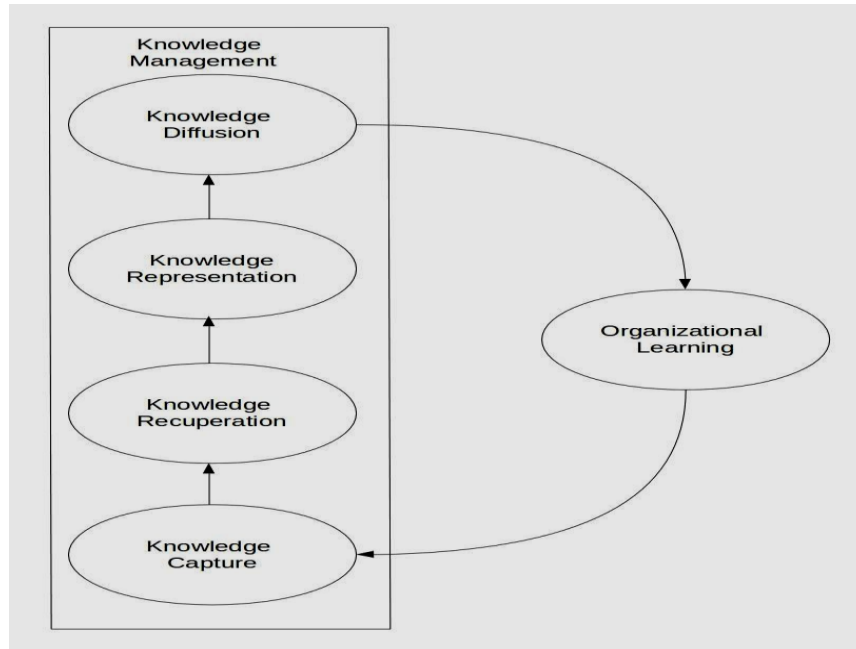
Si bien el capital intelectual es dependiente de la estrategia y enfoque de negocio de la empresa³⁸, en una, desarrolladora de software, sería el producto software desarrollado y entregado a los clientes sería el pilar o uno de los pilares de la misma. Usualmente las empresas pequeñas y microempresas (desarrolladoras) no invierten, de momento, muchos recursos en el proceso de diseño detallado del software por la carencia de personal, principalmente, estas desean optar por otras estrategias o metodologías de desarrollo ágiles como SCRUM o XP (hechas para equipos pequeños de desarrollo) para realizar sus procesos de creación y modificación del producto. En estas circunstancias no es común ver un gran número de vistas³⁹ (las de información son las más comunes), ni un proceso formal de diseño de la arquitectura de lo que se va a desarrollar, a la larga los procesos ágiles son los que tienen más problemas con las incidencias (errores de software) reportadas.

Para las empresas desarrolladoras de software grandes (certificadas) es importante la etapa de diseño, estas disponen de arquitectos de software que se encargan de esta actividad, el problema está cuando se desea realizar el mantenimiento en la arquitectura, realizar un cambio a la misma no es sencillo si no se comprende la relación de causalidad que genera las decisiones, si no se comprende que la razón de la decisión no se comprende apropiadamente la arquitectura por ende existe la amenaza de realizar cambios indebidos a lo diseñado que de implementarse después afectaría negativamente a todo el sistema. Es por esta razón que las decisiones deben considerarse como conocimiento que debe administrarse, una base sólida de diseño es el paso inicial para que el producto quede bien desarrollado, si esta fase se demora la producción también, si esta queda mal, el producto final también quedará mal.

³⁸ R. Johan, Capital intelectual: el valor intangible de la empresa, CA: México: Paidós, p 9-18.

³⁹ KRUCHTEN, Philippe. Op. cit. p. 19

Figura 1. Esquema del ciclo de vida de la administración del conocimiento



Fuente: autores.

2.4 WEB SEMÁNTICA

Según la W3C la web semántica es una web extendida, dotada de mayor significado en la que cualquier usuario en el internet podría encontrar respuestas a sus preguntas de forma más sencilla gracias a una información mejor definida. Al dotar la web de mayor significado se pueden obtener soluciones a inconvenientes habituales en la búsqueda de información gracias a la utilización de una infraestructura común, resolviendo problemas ocasionados por una web carente de semántica que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante⁴⁰.

La web semántica busca dotar de estructura y anotar los recursos con semántica explícita procesable por las máquinas; es decir, darle mayor autonomía de decisión a las máquinas intermediarias. Por ejemplo, un software puede aplicar ciertos algoritmos de decisión basados en modelos estadísticos para definir la página destino en un direccionamiento, pero los equipos son “ignorantes” acerca de lo que pasa, las aplicaciones son las que se encargan de ello y estas pueden

⁴⁰ W3C. Guía Breve de Web Semántica [En línea], [consultado el 8 de octubre de 2015]. Disponible en: <http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica>.

diferir entre sí.

Al extenderse más allá de las webs semánticas, otorgando más libertad en edición y refinamiento de contenido, estas webs extendidas se conocerán como wiki semántica⁴¹. Esta wiki se diferencia de una wiki tradicional en el sentido que la primera posee todas las bondades descritas asociadas a una web semántica, como lo son la descripción del contenido web y la existencia de significado en este para generar una relación explícita entre sus componentes hipervinculitos (páginas webs - vistas- conectadas por hipervínculos).

Una wiki es un sistema que se basa en el contenido web (puede llegar a ser visto como un sitio web) que basa su atractivo en la actualización y modificación de su contenido (conocimiento explícito), en texto y multimedia, de una manera fácil y flexible. En este contexto no difiere mucho, en cuanto a propiedades se trata, la web tradicional, más que es la característica clave ya expuesta. La wiki semántica ofrece una forma más sencilla de incluir anotaciones semánticas a conceptos ontológicos reflejados como cada uno de los artículos que se encuentran en ella, así en un artículo de la wiki semántica estas anotaciones, que toman la forma de hipervínculos, nos permiten navegar entre artículos (conceptos ontológicos) relacionados con el que actualmente se está observando.

En este contexto la idea de la web semántica es que exista una red de nodos tipificados e interconectados mediante clases, y relaciones definidas por una ontología compartida por sus distintos autores. Por ejemplo, una vez creada una ontología sobre cuadros y pinturas, un museo virtual puede organizar sus contenidos definiendo instancias de pintores, cuadros, etcétera, inter relacionándolas y publicándolas en la web semántica.

2.5 ONTOLOGÍAS

En el sentido filosófico una ontología es un sistema particular de categorías para una cierta visión del mundo. Por otro lado, en el uso más prevaeciente en el campo de la inteligencia artificial, una ontología hace referencia a un “constructo ingenieril”, constituido por un vocabulario específico usado para describir cierta realidad, además de un conjunto de especulaciones independientes del significado de las palabras que componen dicho vocabulario⁴². No obstante dentro del área de las ciencias computacionales una ontología representa el esfuerzo por formular un

⁴¹ REUTELSHOFER Jochen; BAUMEISTER Joachim; PUPPE Frank y . Knowwe A Semantic Wiki for Knowledge Semántica Engineering. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: <https://pdfs.semanticscholar.org/.../c09d67bfd152629aed183bbe>

⁴² GUARINO N. Formal Ontology in Information Systems. 4p. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: uosis.mif.vu.lt/.../Guarino98-Formal%20Ontology%20and%20Inf..

esquema conceptual formal y muy riguroso de un dominio dado, en general puede ser una estructura de datos jerarquizada que contenga todos los elementos relevantes, junto con las relaciones y las reglas que los rigen⁴³. En este orden de ideas una ontología puede proveer de un lenguaje común para favorecer la comprensión de las ideas generadas en procesos de análisis, procesos que se ven bastante arraigados a las etapas de un ciclo de vida de un software (levantamiento, diseño, implementación, pruebas y mantenimiento).

Aunque las ontologías ven sus inicios en la filosofía, ciertamente uno de sus principales beneficios es la interrelación de conceptos o ideas que diversos autores han sabido explotar. Autores como Bench-Capon proponen la utilización de ontologías para corroborar la coherencia de los conocimientos, proveer un medio para estructurar pruebas y para sugerir propuestas de acción cuando se presente una falla⁴⁴. Por regla general (e inclusive sentido común) para mantener un software es necesario tener una comprensión profunda del mismo, no solo a nivel funcional sino también a un nivel interactivo, como cada uno de los componentes que componen el software se integran para satisfacer los requerimientos establecidos. Inclusive en la etapa de levantamiento de requerimientos se pueden presentar vacíos comunicativos entre los interesados (stakeholders) y los analistas, Benaroch en este contexto plantea un método para capturar los requerimientos e incluye en él una ontología local que luego se traducirá en una base de datos relacional⁴⁵.

Otros trabajos como AutoBayes, una aplicación Open Source (código abierto) de la NASA que permite la “síntesis” de algoritmos personalizados a partir de unas especificaciones declarativas y compactas en el dominio del análisis, utilizan algoritmos basados en modelos estadísticos y de predicción de estados que resultaron ser difíciles de mantener por su alta complejidad. Sin embargo, los desarrolladores de esta aplicación están explorando la posibilidad de usar ontologías, ya que estos creen que facilitará la extensión del dominio de análisis, la escritura de los esquemas, la validación en salida de estos y la generación de artefactos adicionales, entre otras ventajas.

⁴³ WONGTHONGTHAM Pornpit, KHADEER Farookh, CHANG Elizabeth, DILLON Tharam. Multi-site Software Engineering Ontology Instantiations Management Using Reputation Based Decision Making. 2 p.

⁴⁴ BENCH-CAPON, Citado por: ZAPATA Carlos, GIRALDO Gloria, URREGO Germán. “The Role of Ontologies in the Verification and Validation of Knowledge Based Systems: Las ontologías en la ingeniería de *software*: un acercamiento de dos grandes áreas del conocimiento. Usa: 9th International Workshop on Database and Expert Systems Applications (DEXA), 1998. P. 20

⁴⁵ BENAROCH, M. citado por: ZAPATA Carlos, GIRALDO Gloria y URREGO Germán, “Specifying Local Ontologies in Support of Semantic Interoperability of Distributed Inter-organizational Applications,” en Proc. of the 5th Intl. Workshop on Next Generation Inf. Techn. and Systems, Caesarea, Israel, 2002, pp. 90-106. En: Rev. ing. univ. Medellín. No.16 (Jun, 2010). ISSN 1692-3324.

3. DISEÑO METODOLÓGICO DE LA INVESTIGACIÓN

3.1 SISTEMA DE HIPÓTESIS

3.1.1 Hipótesis de trabajo. La utilización de una wiki semántica con enfoque en administración de conocimiento para la representación, divulgación y recuperación de decisiones de arquitecturas de software reducirá los tiempos requeridos para la comprensión y toma correcta de decisiones de arquitecturas de software.

3.1.1.1 Hipótesis nula. La utilización de una wiki semántica con enfoque en administración de conocimiento para la representación, divulgación y recuperación de decisiones de arquitecturas de software no reducirá los tiempos requeridos para la comprensión y toma correcta de decisiones de arquitecturas de software.

3.2 SISTEMA DE VARIABLES

3.2.1 Variables independientes

- Utilización de una wiki semántica (tecnología).
- Modelo de decisiones.

3.2.2 Variables dependientes

- La comprensión de las decisiones.

3.2.3 Variables intervinientes

- Enfoque de administración de conocimiento (base metodológica).
- Enfoque de diseño de arquitectura.

3.3 ANÁLISIS DEL PROBLEMA

Philippe Krutchen expone en uno de sus trabajos *An Ontology of Architectural Design Decisions in Software-Intensive Systems* un modelo de conocimiento para las decisiones de diseño arquitectural, profundizando considerablemente en la decisión como elemento que puede “transformarse” según las circunstancias, es decir, una misma decisión de diseño puede variar según el contexto de la decisión (ontocrítica, pericrítica o diacrítica) y todo lo que lleva a tomar una decisión como esa.

Existen varios trabajos y papers que tratan de aterrizar un modelo de conocimiento para las decisiones de arquitectura de software además de Philippe Krutchen, Hernán Astudillo, Darío Correal y Gilberto Pedraza García, bien trataron este tema

en un artículo llamado "Analysis of Design Meetings of Understanding Software architecture decisions"⁴⁶ al proponer un Diseño de Análisis de Intervención Verbal que busca formalizar las reuniones de diseño de arquitectura para que los resultados se vean en función de las decisiones de arquitectura de software y diversos elementos influyentes. De manera similar Jan Salvador, Anton Jansen, Jos Nijhuis y Jan Bosch en el libro "Rationale Management in software Engineering" tratan en el capítulo 3 "Design Decisions: The Bridge between Rationale and Architecture" la importancia de los argumentos y la administración de los mismos en el proceso de desarrollo de una arquitectura de software así mismo como definen que es necesario que existan las alternativas, las soluciones, las reglas de diseño y restricciones entre otras cosas justo al mismo argumento de la decisión para la construcción de arquitecturas.

Otro trabajo como "Uso de Ontologías para mapear una Arquitectura de Software con su Implementación"⁴⁷ propone métodos matemáticos que permiten evaluar la similitud entre lo diseñado y lo implementado desde un enfoque lingüístico, es de concerns y de estructuras, pero para ello requieren que se hayan creado dos tipos de ontología: Una arquitectónica (la de interés actual) y una de implementación.

Existen motores de wiki y servicios web que utilizan diversos motores de wiki gratuitos en la red, estos siguen la filosofía básica del wiki de difundir contenido a través de páginas que representa algún tema en especial. Un motor de wiki bastante conocido y popular por ser la base de wikipedia es Media Wiki, a este se le han hecho bastantes extensiones siendo una de las más famosas semantic media wiki, una extensión en php que permite implementar semántica a las páginas creadas a través del uso de anotaciones lo cual disminuye las inconsistencias, el contenido repetido y agrega la posibilidad de hacer consultas con carácter semántico en las páginas. Semantic Media Wiki entre sus funciones permite importar ontologías personalizadas que servirán de esqueleto para la creación de las páginas en la wiki, pero solo eso, es decir Semantic Media Wiki no hace mapeo entre las etiquetas que representarán algún elemento de la ontología y el elemento en sí, la ontología importada solo servirá de estructura para el contenido de la wiki.

3.4 TÉCNICAS DE LA RECOLECCIÓN DE LA INFORMACIÓN

Se realizará una entrevista explicativa, en donde se evaluará el nivel de comprensión de los conocimientos de la arquitectura software (eficacia), la

⁴⁶ PEDRAZA-GARCIA, Gilberto, Astudillo, Hernán and Correal, Darío. Analysis of design meetings for understanding software architecture decisions," 2014 XL Latin American Computing Conference (CLEI), Montevideo, 2014, pp. 1-10. doi: 10.1109/CLEI.2014.6965159_..

⁴⁷ VÁSQUEZ H, DIAZ J. Uso de Ontologías para mapear una Arquitectura de Software con su Implementación, p. 101-106.

productividad en la comprensión de la arquitectura y facilidad de uso del prototipo (eficiencia), la cual se aplicará a diferentes grupos de arquitectos de software, quienes desde su experiencia previa con otros enfoques de administración de conocimiento calificarán el prototipo según las categorías de interés expuestas anteriormente. Por lo tanto, es necesario que el prototipo se haya utilizado previamente para poder aplicarla.

3.5 METODOLOGÍA DE DESARROLLO

Para la construcción del prototipo de wiki semántica se seguirá el modelo de desarrollo en espiral definido por Barry Boehm⁴⁸, con ciertas modificaciones en las actividades establecidas en cada uno de los ciclos del modelo original. No se levantará, ni se validará requerimientos con los stakeholders porque ya están definidos en nuestro problema. Las posteriores actividades propuestas en el modelo original se seguirán normalmente, aunque se pueden omitir algunas tareas.

Se optó por seguir este modelo por su enfoque en el control de riesgos a lo largo del ciclo de vida del software al igual que su metodología de desarrollo evolutivo, donde podemos reducir o ampliar el número de ciclos de trabajo según se requiera. Lo anterior unido a una flexibilidad inherente en la determinación de cada una de las fases a desarrollar a lo largo de cada una de las iteraciones, convierte la metodología de desarrollo en espiral en una decisión adecuada.

El desarrollo de la wiki semántica se basará en las tareas y actividades expuestas en la tabla 1.

Tabla 1. Modelo de desarrollo pensado para el desarrollo de la wiki semántica

⁴⁸ BOEHM Barry. A Spiral Model of Software Development and Enhancement. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: csse.usc.edu/TECHRPTS/1988/usccse88-500/usccse88-500.pdf

Actividades	Tareas	Resultados
Diseño de la ontología.	Definición del dominio de la ontología	Modelo conceptual del dominio
	Definición de las clases o entidades.	Documento de diseño de la ontología.
	Definición de los individuos.	
	Definición de las propiedades.	
	Definición de las relaciones.	
Aprendisaje de las herramientas tecnológicas.	Aprendisaje de la herramienta Protégé.	Ninguno.
	Aprendisaje de SPARQL.	
	Aprendisaje de la herramienta Protý.	
Implementación de la ontología en Protégé	Creación del proyecto.	Archivo OWL que representa la ontología (prototipo de la ontología en Protégé).
	Creación de las clases y subclases.	
	Creación y asociación de las propiedades.	
	Creación de los objetos o relaciones.	
	Creación de las instancias o individuos.	
Pruebas a la ontología desarrollada	Creación de consultas SPARQL a la ontología	Permiso para continuar el proceso.
Construir la wiki semántica	Definir la estructura de la wiki semántica.	Documento de diseño de la wiki semántica (plantillas, formularios, categorías).
	Integrar los wireframes con la ontología.	Una wiki semántica funcional (prototipo).
Pruebas a la wiki desarrollada	Diseño del plan de pruebas.	Documento del plan de pruebas.
	Verificación de la navegación entre páginas	Aprobación para continuar.
	Verificación de modificación de contenido textual	
	Verificación de carga apropiada de imágenes	
	Verificación del correcto funcionamiento de los formularios.	
	Verificación de correcta carga de anotaciones	
Diseño del caso de estudio.	Realización del documento del caso de estudio.	Documento del caso de estudio.
Implementación del caso de estudio	Modificar el contenido de la wiki semántica con lo requerido por el caso de estudio en cuestión.	Una wiki semántica adaptada al caso de uso específico.
Pruebas a la wiki semántica adaptada	Verificación de la navegación entre páginas.	Aprobación para continuar.
	Verificación de modificación de contenido textual.	
	Verificación de carga apropiada de imágenes.	
	Verificación de correcta carga de anotaciones.	
Diseño del experimento de validación.	Diseño del experimento.	Guía paso a paso del experimento.
	Diseño de la encuesta.	Encuesta de percepción.
Validación con los arquitectos	Encontrar arquitectos que deseen evaluar.	Hipótesis probada o no, posible replantamiento de la estrategia y modificación de la wiki.
	Explicación del caso de estudio.	
	Aplicar la encuesta.	
	Analizar los resultados.	

Fuente: autores.

necesario conocer ciertas variables que se ven involucradas en el proceso de toma de decisión:

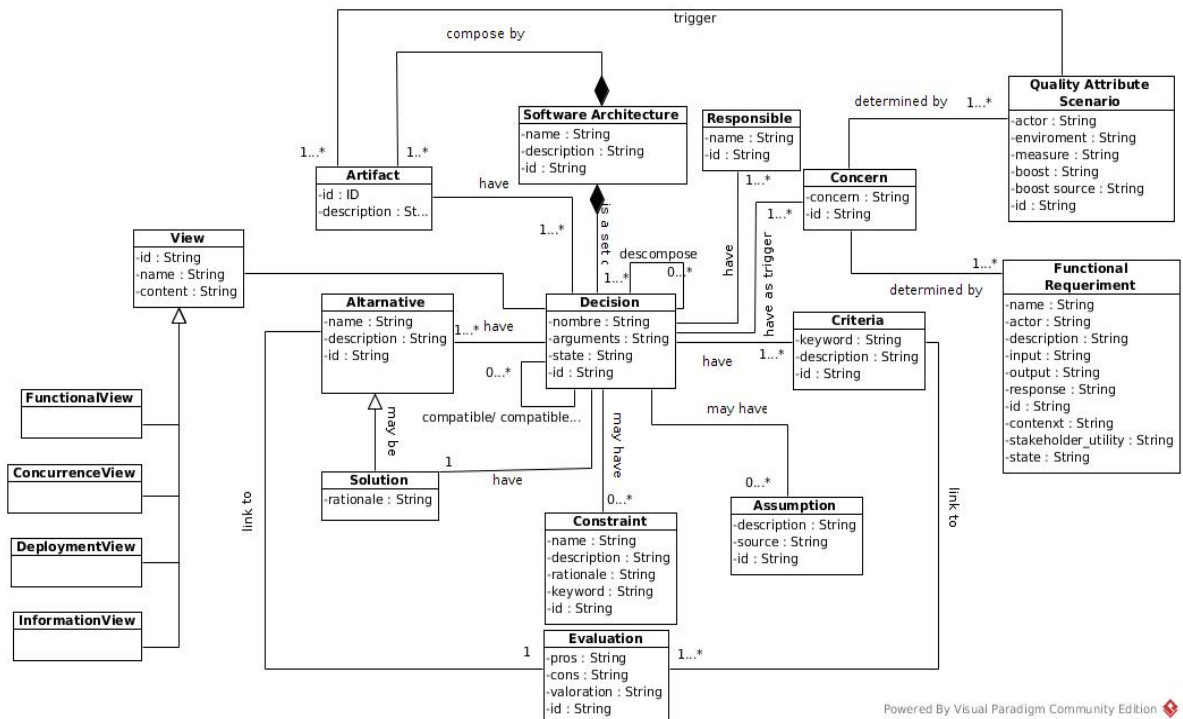
- Atributos de calidad. Estos definen y miden la calidad del software desde diferentes aspectos. Las decisiones se toman en función de satisfacer estos atributos.
- Requerimientos funcionales. Representación formal de una necesidad del cliente. Suelen incluir propiedades como entradas requeridas y salidas necesitadas. Las decisiones dependen existencialmente de ellos.
- Artefactos. productos tangibles que se deciden desarrollar en el ciclo de vida del software. Las decisiones de arquitectura de software se toman para hacer realidad los artefactos, sin mencionar que los artefactos también son de cierta manera soluciones de decisiones.
- Alternativas. Las alternativas representan posibles soluciones para una decisión. Estas se evalúan según ciertos criterios para obtener la solución deseada.
- Solución. Alternativa escogida. La solución debe poseer una justificación que puede o no satisfacer por completo los criterios de evaluación.
- Criterios. Los criterios de evaluación de las decisiones se toman en función de los atributos de calidad y las restricciones que los rigen. Es necesario conocer los criterios ya que estos determinan qué alternativa llegaría a convertirse en solución de una decisión.
- Restricciones. las restricciones son impedimentos que se tienen en cuenta al momento de tomar una decisión. Estos impedimentos suelen ser dados por el cliente o la organización en la que se trabaja. Las restricciones afectan directamente a las alternativas de una decisión, determinan cuáles son consideradas en un proceso de evaluación y cuáles no.
- Evaluación. Es el proceso en que se confronta una alternativa con los criterios que rigen la decisión. En la evaluación se definen qué es lo positivo y negativo de la alternativa con referencia a los criterios de evaluación. Es necesario conocer este proceso ya que aquí se ve en esencia las razones de las decisiones interactuando con los atributos de calidad, restricciones y requerimientos funcionales.
- Las Vistas. Estas son la representación de las decisiones. Los esquemas que representen un aspecto específico de la arquitectura mostrando cómo interactúan

ciertos elementos involucrados. Incluirlas dará cierta facilidad sobre la comprensión de las razones de las decisiones tomadas.

El conocer aquellas variables que intervienen en el proceso de toma de decisiones permite entender la globalidad del mismo y por tanto las razones que los arquitectos tuvieron al tomar sus decisiones. Las razones son el conocimiento que se desea administrar (el decidir utilizar un determinado servidor de aplicaciones, crear un componente en específico no dicen mucho si solo se conoce eso), en estas se esconden las restricciones de diseño, de organización que se tuvieron en cuenta, las evaluaciones realizadas, los criterios basados en atributos de calidad considerados, las necesidades de los clientes que catalizaron la decisión que se justifica y varios aspectos adicionales que hacen que el que esté intentando comprender las decisiones lo haga realmente.

4.2 DISEÑO DE LA ONTOLOGÍA

Figura 3. Primera parte del diseño de la ontología que representa el modelo de decisiones



Fuente: autores.

La ontología es la base de la wiki semántica que se va a realizar, esta debe representar el modelo de decisiones planteado en la figura 3, más sin embargo no

es un reflejo. Se especificó que la wiki semántica debe realizarse siguiendo ciertos parámetros que Rozanski en su texto *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*⁵⁰ especificó. La wiki semántica no aplicará todos y cada uno de los aspectos propuestos por Rozanski pero si se incluirá las siguientes preocupaciones de seguridad:

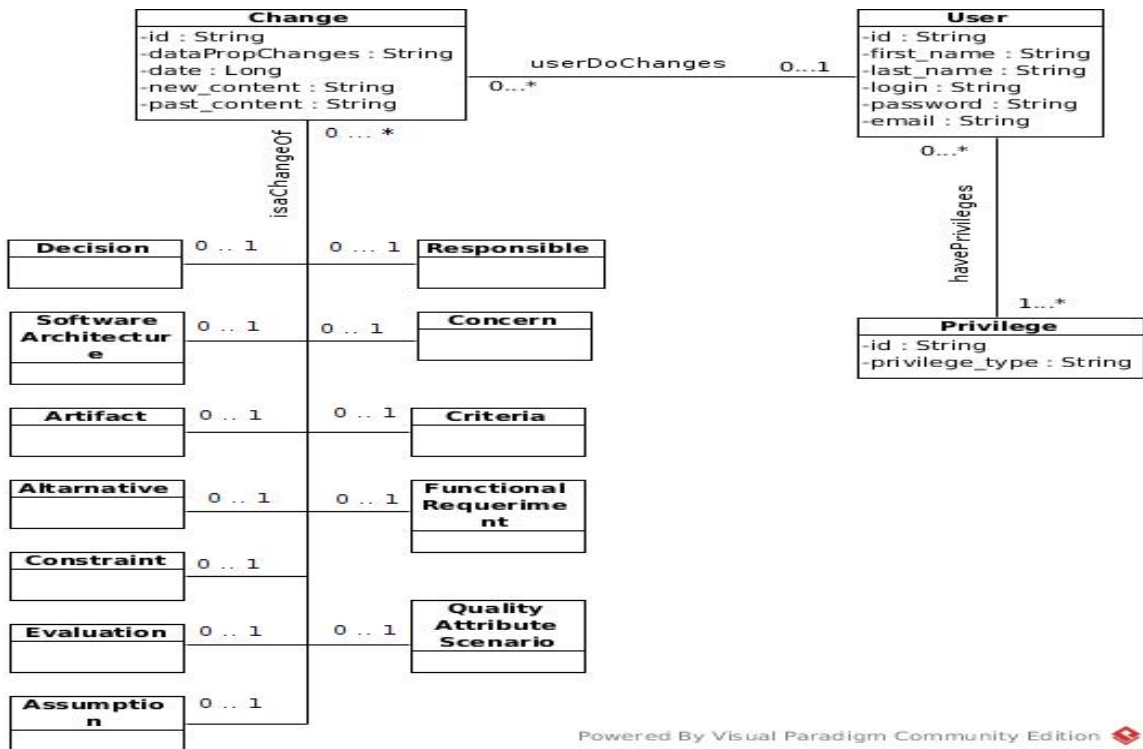
- Los recursos: Son la razón por la cual se tiene seguridad en el sistema, es decir información valiosa que desea ser protegida. En este caso son las decisiones de arquitectura de software.
- Los directores: Personas, roles, equipo tecnológico u otros sistemas informáticos que usarán los recursos. Consideramos solamente tres roles: administrador, editor y lector.
- Las políticas de seguridad: Son reglas, normas o prácticas que guían la toma de decisiones, definen controles y garantías, determinan permisos entre otros y que el sistema debe cumplir. En la wiki semántica solo se incluirá las siguientes:
 - Solo usuarios registrados podrán ver las decisiones de arquitectura de software.
 - Un usuario con rol de editor puede modificar el contenido de una decisión o de elementos relacionados.
 - Un usuario con rol de administrador puede hacer lo que un usuario editor, pero además tiene permitido revertir cambios hechos sobre el contenido cuando sea necesario.
- Confidencialidad: Ocultar los recursos importantes a aquellos que no deben tener acceso a ella. Recurrimos a un proceso de autenticación de usuarios y permisos para cumplir esto.
- Detección y corrección de fallos: Más allá de lo que pueda sugerir el nombre, la detección y corrección de fallos no es un proceso que se le delega al sistema solamente, es también considerado la participación humana y procesos de seguridad empresariales. En la wiki semántica se incluirá un módulo de trazabilidad de cambios realizados que permitirá devolverse a estados anteriores de ser necesario.

⁵⁰ ROZANSKI Nick, WOODS Eoin. *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Segunda Edición, pp 2-13. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: www.cs.du.edu/.../architecture/SW-SystemsArch-WorkingwithStakeholdersUsi.

- **Responsabilidad.** Significa asegurar que cada acción sea seguida y asociada a un director en particular. el módulo de cambios muestra qué usuario con qué rol realizó determinado cambio.

Como la wiki semántica aplicará las preocupaciones de seguridad anteriormente nombradas es requerido que la ontología no solo represente el modelo de decisiones de la figura 4, sino que también se adapte para concretar las preocupaciones de seguridad en especial la responsabilidad, detección y corrección de fallos, los directores y la confidencialidad. La figura 2 muestra la primera parte del diseño de la ontología propuesto, esta parte tiene que ver con el modelo de decisiones presentado en la figura 1 solo que aquí se incluyen ciertos detalles de implementación como el tipo de los datos de propiedad asociados a cada clase y un id obligatorio por individuo del tipo de cada clase. La segunda parte del diseño que se muestra en la figura 4 tiene que ver en cómo la ontología se adapta a las preocupaciones de seguridad.

Figura 4. Segunda parte del diseño de la ontología, se adapta la ontología para aplicar ciertas preocupaciones de seguridad.



Fuente: autores

La inclusión de las clases Change y User nos permite implementar la responsabilidad y la detección y corrección de fallos ya que con esto es posible saber qué director hizo un cambio sobre un individuo de la ontología, cuando lo

hizo y qué hizo además de poder devolverse hasta un estado anterior cuando se requiera. Según las políticas de seguridad consideradas un administrador solo podrá realizar este tipo de acciones.

Al añadir la clase User podemos asegurar de cierta forma la confidencialidad en el sistema al no permitir que cualquiera pueda acceder a ella. Al agregar la clase Privilege se garantiza parte de algo que Rozanski llama Calidad deseada⁵¹.

⁵¹ ROZANSKI Nick, WOODS Eoin. Op. cit. p. 34

5. IMPLEMENTACIÓN DE LA SOLUCIÓN

5.1 DESCRIPCIÓN DE LOS CASOS DE USO

Los casos de uso que se presentan a continuación se basan en consideraciones personales acerca de cómo la información debería ser representada y distribuida en la wiki semántica.

La función de consulta general permitirá al usuario buscar coincidencias textuales utilizando una caja de texto. Esta consulta se realiza sobre todos los datos de propiedad de todos los individuos de la ontología.

Tabla 2. Caso de uso Consultar individuos a través de coincidencias textuales.

Consultar individuos a través de coincidencias textuales.		RF- 0001
1.0.0		Versión
David Pulido, Gabriel Mejía		Autores
Yo como usuario requiero un módulo que permita obtener por medio de un filtro los diversos individuos de la ontología, en donde exista coincidencia textual en los datos de propiedad de uno o más individuos.		Descripción
Usuario con rol de administrador. Usuario con rol de edición. Usuario con rol general.		Actores
Datos cargados en la ontología.		Precondición
Acción	Paso	Secuencia Normal
El usuario ingresa al sistema.	1	
El usuario ingresa al módulo de consulta general	2	
El usuario deberá escribir en el cuadro de texto algo que desee buscar. Una lista de posibles resultados con respecto al texto ingresado.	3	
Acción	Paso	Excepciones
Ninguna	1	
Lista de coincidencias.		Postcondición

Fuente: autores.

Se creará una lista rápida de acciones que facilitará al usuario la navegación entre individuos consultados. Esta lista se actualiza cada vez que se consulta un

individuo en la ontología brindando un acceso directo a él desde cualquier parte de la aplicación.

Tabla 3. Caso de uso Consultar todos los cambios realizados sobre los individuos

Consultar todos los cambios realizados sobre los individuos.		RF- 0002
1.0.0		Versión
David Pulido, Gabriel Mejía		Autores
Yo como usuario requiero una lista dinámica que permita accesos rápidos a los individuos consultados previamente en la aplicación.		Descripción
<ul style="list-style-type: none"> ● Usuario con rol de administrador. ● Usuario con rol de edición. ● Usuario con rol general. 		Actores
<ul style="list-style-type: none"> ● Datos cargados en la ontología. ● Por lo menos un individuo consultado. 		Precondición
Acción	Paso	Secuencia Normal
El usuario ingresa al sistema.	1	
El usuario ingresa al módulo de consulta general	2	
<ul style="list-style-type: none"> ● El usuario deberá escribir en el cuadro de texto algo que desee buscar. ● Una lista de posibles resultados con respecto al texto ingresado. ● Seleccionar un individuo de la lista rápida de acciones. ● Se irá al módulo de consulta del individuo. 	3	
Acción	Paso	Excepciones
<ul style="list-style-type: none"> ● Ninguna 	1	
Lista de todos los cambios realizados en una línea temporal.		Postcondición

Fuente: autores

El módulo de consulta es aquel en donde se puede observar los detalles de un individuo de la ontología. A este módulo es posible acceder desde diferentes partes de la aplicación.

Tabla 4. Caso de uso Consultar individuo

Consultar individuo		RF- 0003
1.0.0		Versión
David Pulido, Gabriel Mejía		Autores
Yo como usuario requiero un módulo que permita visualizar la información de un individuo seleccionado.		Descripción
<ul style="list-style-type: none"> • Usuario con rol de administrador. • Usuario con rol de edición. • Usuario con rol general. 		Actores
<ul style="list-style-type: none"> • Datos cargados en la ontología. 		Precondición
Acción	Paso	Secuencia Normal
El usuario ingresa al sistema.	1	
El usuario ingresa al módulo de consultas	2	
<ul style="list-style-type: none"> • El usuario deberá elegir un individuo. • Se muestra la información de dicho individuo, además se muestra en forma de enlaces directos los individuos relaciones con el consultado. 	3	
Acción	Paso	Excepciones
<ul style="list-style-type: none"> • Ninguna 	1	
Información del individuo encontrado y lista de individuos relacionados a él.		Postcondición

Fuente: autores.

El módulo de consultas no solo debe permitir observar los detalles de cierto individuo, también debe permitir modificar el contenido de uno o más de sus detalles, este es el propósito de la extensión de edición para el módulo de consultas.

Tabla 5. Caso de uso Editar contenido de un individuo

Editar contenido de un individuo.		RF- 0004
1.0.0		Versión
David Pulido, Gabriel Mejía		Autores
Yo como usuario requiero editar el contenido de un individuo consultado para efectuar cambios cuando sean necesarios.		Descripción
<ul style="list-style-type: none"> • Usuario con rol de administrador. • Usuario con rol de edición. 		Actores
<ul style="list-style-type: none"> • Datos cargados en la ontología. • Usuario con rol de administrar y de edición. 		Precondición
Acción	Paso	Secuencia Normal
El usuario ingresa al sistema.	1	
El usuario ingresa al módulo de consultas	2	
<ul style="list-style-type: none"> • El usuario deberá elegir un individuo. • Se muestra la información de dicho individuo, además se muestra en forma de enlaces directos los individuos relaciones con el consultado. • Activar el switch correspondiente del individuo a editar. • Edición completada del individuo. 	3	
Acción	Paso	Excepciones
<ul style="list-style-type: none"> • Ninguna 	1	
Cambios realizados guardados satisfactoriamente.		Postcondición

Fuente: autores

La carga de imágenes es importante en el módulo de consultas para un usuario con permiso de edición o un usuario administrador, siempre y cuando estas imágenes que sean vistas de contexto, de despliegue entre otros diagramas que sirvan para contextualizar o comprender mejor una decisión. Se considerarán tres tipos de carga de imágenes: por disco local, por URL y carga de imagen subida previamente.

Tabla 6. Caso de uso Cargar imágenes de Disco

Cargar imágenes de Disco.		RF- 0005
1.0.0		Versión
David Pulido, Gabriel Mejía		Autores
Yo como usuario requiero cargar imágenes que estén alojadas en disco para enriquecer el atributo de un individuo en particular.		Descripción
<ul style="list-style-type: none"> • Usuario con rol de administrador. • Usuario con rol de edición. 		Actores
<ul style="list-style-type: none"> • Datos cargados en la ontología. • Usuario con rol de administrar y de edición. • Usuario en el módulo de consulta • Se debe tener activado la extensión de edición. 		Precondición
Acción	Paso	Secuencia Normal
El usuario presiona el botón que contiene el icono de imagen.	1	
Seleccionar imagen por disco local.	2	
Brindar un nombre y seleccionar cargar la imagen.	3	
Acción	Paso	Excepciones
<ul style="list-style-type: none"> • Se debe escribir un nombre obligatoriamente. • Solo se admiten archivos con extensión jpg, gif y png. 	1	
Imagen cargada satisfactoriamente del disco.		Postcondición

Fuente: autores

Tabla 7. Caso de uso Cargar imágenes por URL

Cargar imágenes por URL.		RF- 0006
1.0.0		Versión
David Pulido, Gabriel Mejía		Autores
Yo como usuario requiero cargar imágenes desde una dirección URL para enriquecer el atributo de un individuo en particular		Descripción
<ul style="list-style-type: none"> • Usuario con rol de administrador. • Usuario con rol de edición. 		Actores
<ul style="list-style-type: none"> • Datos cargados en la ontología. • Usuario con rol de administrar y de edición. • Usuario en el módulo de consulta • Se debe tener activado la extensión de edición. 		Precondición
Acción	Paso	Secuencia Normal
El usuario presiona el botón que contiene el icono de imagen.	1	
Seleccionar la opción de imagen por URL.	2	
Escribir un nombre y la dirección URL de la imagen.	3	
Acción	Paso	Excepciones
<ul style="list-style-type: none"> • Se debe escribir el nombre de la imagen. • La dirección URL debe ser correcta. 	1	
Imagen cargada satisfactoriamente por URL.		Postcondición

Fuente: autores

Tabla 8. Caso de uso crear cargar imagen existente.

Cargar imagen existente.		RF- 0007
1.0.0		Versión
David Pulido, Gabriel Mejía		Autores
Yo como usuario requiero cargar imágenes que hayan sido subidas anteriormente para enriquecer el atributo de un individuo en particular.		Descripción
<ul style="list-style-type: none"> • Usuario con rol de administrador. • Usuario con rol de edición. 		Actores
<ul style="list-style-type: none"> • Datos cargados en la ontología. • Usuario con rol de administrar y de edición. • Usuario en el módulo de consulta • Se debe tener activado la extensión de edición. • Imágenes cargadas previamente. 		Precondición
Acción	Paso	Secuencia Normal
El usuario presiona el botón que contiene el icono de imagen.	1	
Hacer clic en imagen existente.	2	
Seleccionar una de las imágenes que hayan sido cargadas ya sea de disco local o por URL.	3	
Acción	Paso	Excepciones
<ul style="list-style-type: none"> • Se debe seleccionar una sola imagen. 	1	
Imagen ya existente cargada satisfactoriamente.		Postcondición

Fuente: autores

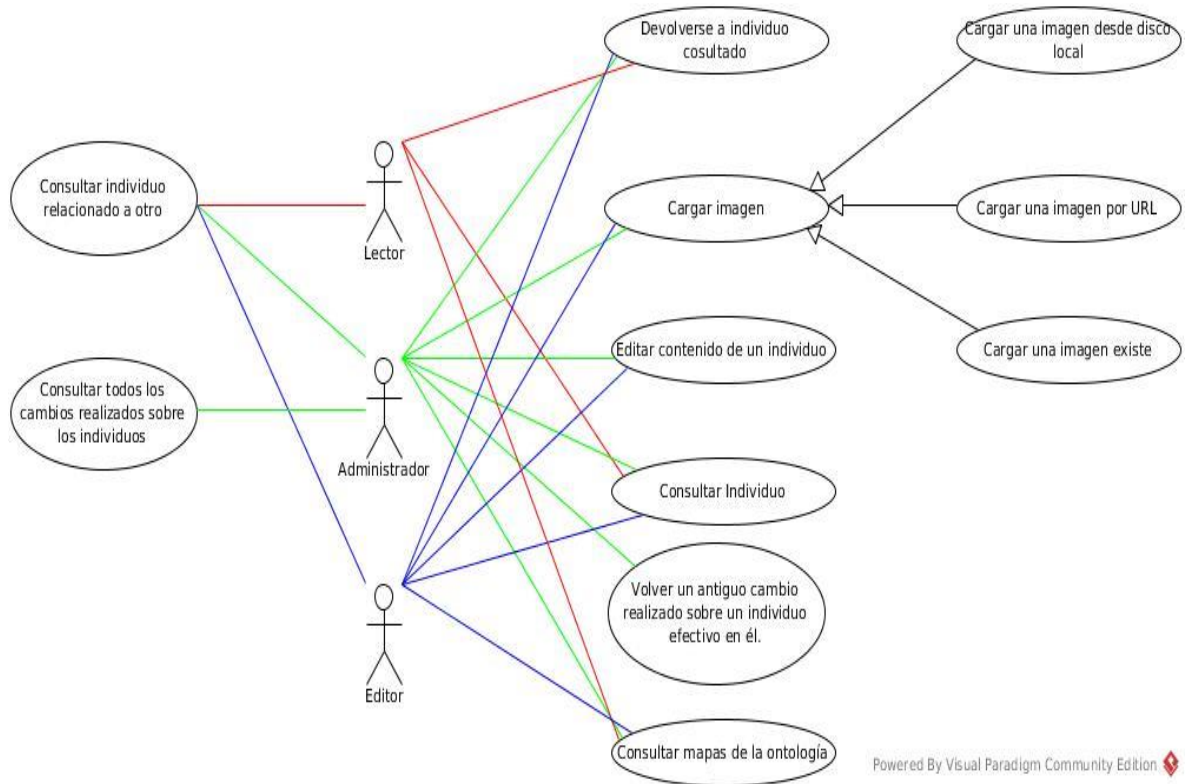
Los individuos de la ontología se pueden relacionar directamente o transitivamente con otros individuos (decisiones relacionándose con alternativas, artefactos relacionándose con alternativas y así). Para evitar la carga de esos individuos relacionados uno a uno en el módulo de consulta se creará un visor o módulo de subconsulta en donde se podrá navegar y observar los detalles de cada individuo relacionado.

Tabla 9. Caso de uso Consultar individuo relacionado a otro

Consultar individuo relacionado a otro		RF- 0008
1.0.0		Versión
David Pulido, Gabriel Mejía		Autores
Yo como usuario requiero consultar la información de los individuos relacionados para evitar perder el contexto de la consulta previa.		Descripción
<ul style="list-style-type: none"> ● Usuario con rol de administrador. ● Usuario con rol de edición. 		Actores
<ul style="list-style-type: none"> ● Datos cargados en la ontología. ● Usuario con rol de administrar y de edición. ● Usuario en el módulo de consulta 		Precondición
Acción	Paso	Secuencia
El usuario presiona el botón de flecha para dirigirse al individuo relacionado sin tener que salir y seleccionar otro.	1	Normal
Acción	Paso	Excepciones
<ul style="list-style-type: none"> ● Ninguna 	1	
Ventana emergente con la información del individuo consultado y una lista de los individuos relacionados a este.		Postcondición

Fuente: autores

Figura 5. Diagrama de casos de uso de la wiki semántica



Fuente: autores

5.2 TECNOLOGÍAS Y PATRONES UTILIZADOS

Para el desarrollo de la ontología se decidió crear un archivo de extensión owl asistido por un aplicativo conocido como Protege, este siendo un entorno de ingeniería de ontologías, brinda facilidades para la creación y modificación de los archivos OWL (Web Ontology Language, lenguaje diseñado para procesar contenido de la información más que solo presentarlo a los humanos) siguiendo el estándar de la W3C.

Posteriormente la manipulación de la ontología a nivel de la wiki semántica se realizó trabajando directamente con RDF (Resources Description Framework), un grupo de especificaciones de la W3C que en su conjunto constituyen un modelo

para el intercambio de datos en la web, y SPARQL para las consultas que se puedan realizar a las ontologías.

Para el desarrollo de la wiki semántica se utilizó java para trabajar el back-end en general, las vistas se trabajaron con angularjs y foundation for apps. En el back-end se recurrió al framework OpenRDF4J para la manipulación de repositorios, consultas y administración general de la ontología. Todo lo anterior se consolidó en un único proyecto web maven desarrollado en NetBeans 8.1

La aplicación wiki semántica se construyó utilizando el patrón de diseño MVC, todas las peticiones (realizadas con Ajax desde la vista) son gestionadas por controladores creados utilizando Spring MVC.

Aunque Spring MVC implementa el patrón de diseño de controlador frontal y es necesario configurar un dispatcher servlet la aplicación no se desarrolla usando dicho patrón. En general se disponen de diferentes controladores, uno por cada clase de la ontología, también se dispone de diferentes clases que implementan métodos que inicializan repositorios, abren conexiones a los repositorios e implementan métodos para consultas y manipulación general de la ontología. Únicamente se maneja un repositorio nativo (en disco), este repositorio es independiente del documento OWL y es sobre este en que los las consultas y cambios se realizan.

¿Por qué java? Es un lenguaje de programación orientado a objetos, multiplataforma y gratuito. Cuenta con un gran número paquetes y librerías que facilitan el desarrollo al concentrarse en lo que se debe hacer en la aplicación y no en otras cosas que ya están resueltas.

¿Por qué Spring? De fácil configuración desde IDEs como Netbeans. Permite la configuración de servicios REST de manera sencilla usando anotaciones, posee un amplio soporte y una documentación extensa.

¿Por qué HTML5? Permite estructurar las páginas webs mejor que su antecesor añadiendo etiquetas que representan una posible sección del documento. La inclusión de nuevas etiquetas de propósito definido que en situaciones especiales son de gran utilidad. Busca mejor distinción entre el estilo (CSS), la funcionalidad (javascript) y la estructura de la página web (HTML5).

¿Por qué CSS3? Para el formato de las páginas web. CSS3 presenta ciertas ventajas con su antecesor al añadir ciertas propiedades para sombras, animaciones y renderizados más complejos entre otras cosas, para evitar recurrirse a ciertos “trucos” para realizarlos.

¿Por qué angular? Angular es framework javascript orientado a plantillas que permite darle estructura a la lógica de presentación, pudiendo implementar una

arquitectura de componentes, una arquitectura de servicios, un patrón de diseño MVC enlazando la vista con un modelo programado usando el framework, además de poder implementar pruebas sobre los módulos y servicios programados.

5.3 IMPLEMENTACIÓN DE LA APLICACIÓN

5.3.1 Implementación de los controladores y el modelo. Se crearon tres tipos de clases:

- Entidades: Artifact, Decisión, Alternative, etc.
- Transacciones: todas aquellas clases que terminan en “Transaction”.
- Controladores: todas las clases que terminan en “Controller”.

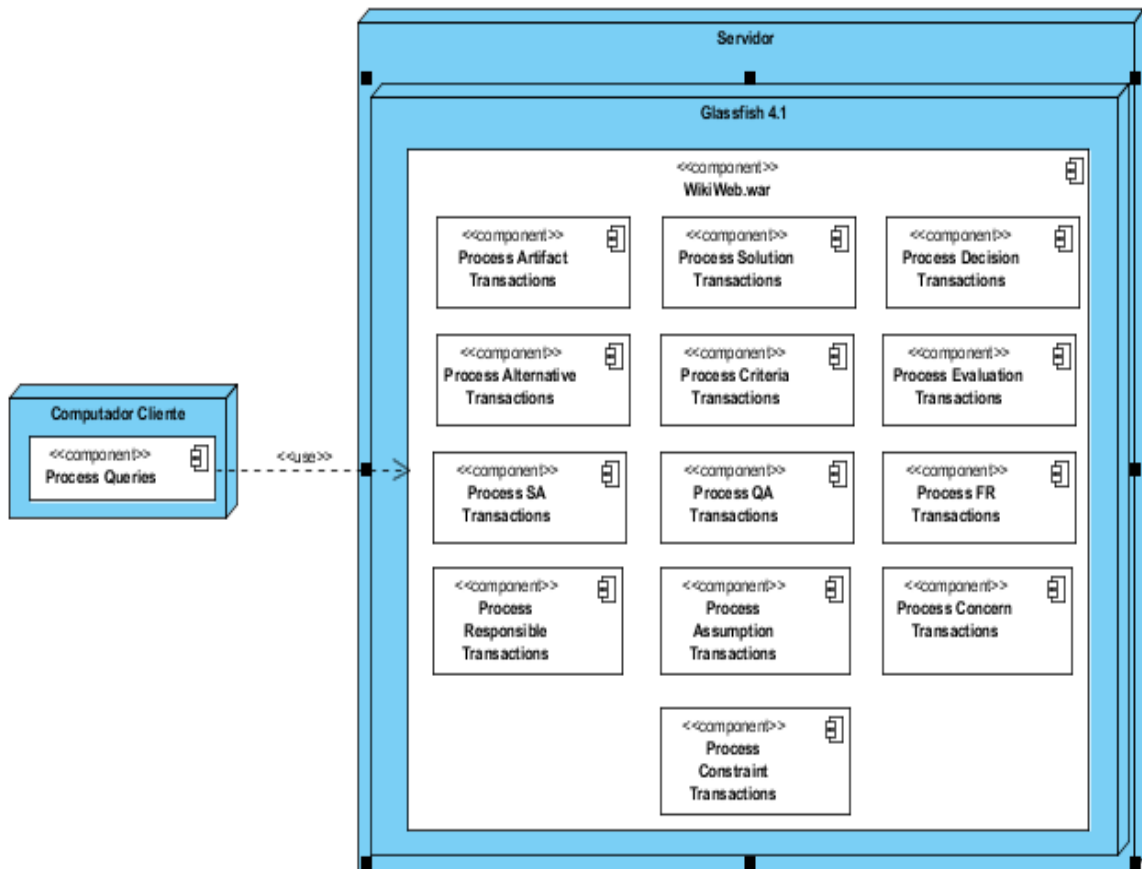
Las entidades serán las clases que moldearán la ontología en el back-end, se decidió hacer de esta manera porque facilitaba el transporte de datos desde el negocio hasta la vista, si se tiene una entidad o lista de entidades JsonFactory podría convertir lo anterior a una cadena de texto en formato json, este formato de texto tiene un amplio soporte nativo en el lenguaje de JavaScript por lo que se podría recorrer los resultados con relativa facilidad.

Cada entidad posee atributos que corresponden a los datos de propiedad de la clase en la ontología que están representando. Las relaciones, dependiendo de su cardinalidad y sentido, se pueden convertir en atributos de la clase o lista de atributos de la clase de tipo de la clase con la que se relacionan.

Las clases de transacciones implementan métodos que sirven para insertar, eliminar, actualizar y consultar a la ontología. Cada entidad posee su equivalente clase de transacción. Los métodos de consulta devuelven la entidad correspondiente o una lista de estas dependiendo de lo que se desee consultar a la ontología.

Las clases controladoras reciben peticiones desde la vista llamando métodos de las transacciones para efectuar ciertas acciones o devolver resultados. Todos los controladores se desarrollaron utilizando Spring MVC y se concibieron como controladores RestFULL por lo que todo lo que se otorga se considerará como un recurso para la vista desde donde se invocó y no una nueva vista. En este punto todo envío de datos que se haga desde los controladores a la vista se hará en formato json, en cambio los parámetros requeridos por los métodos de los controladores pueden o no ser enviados en formato json. En este orden de ideas y según la figura 5, se dispone de diversos componentes dentro del war, cada uno de esos componentes se traducen en una estructura de clases dependientes del diseño de la ontología.

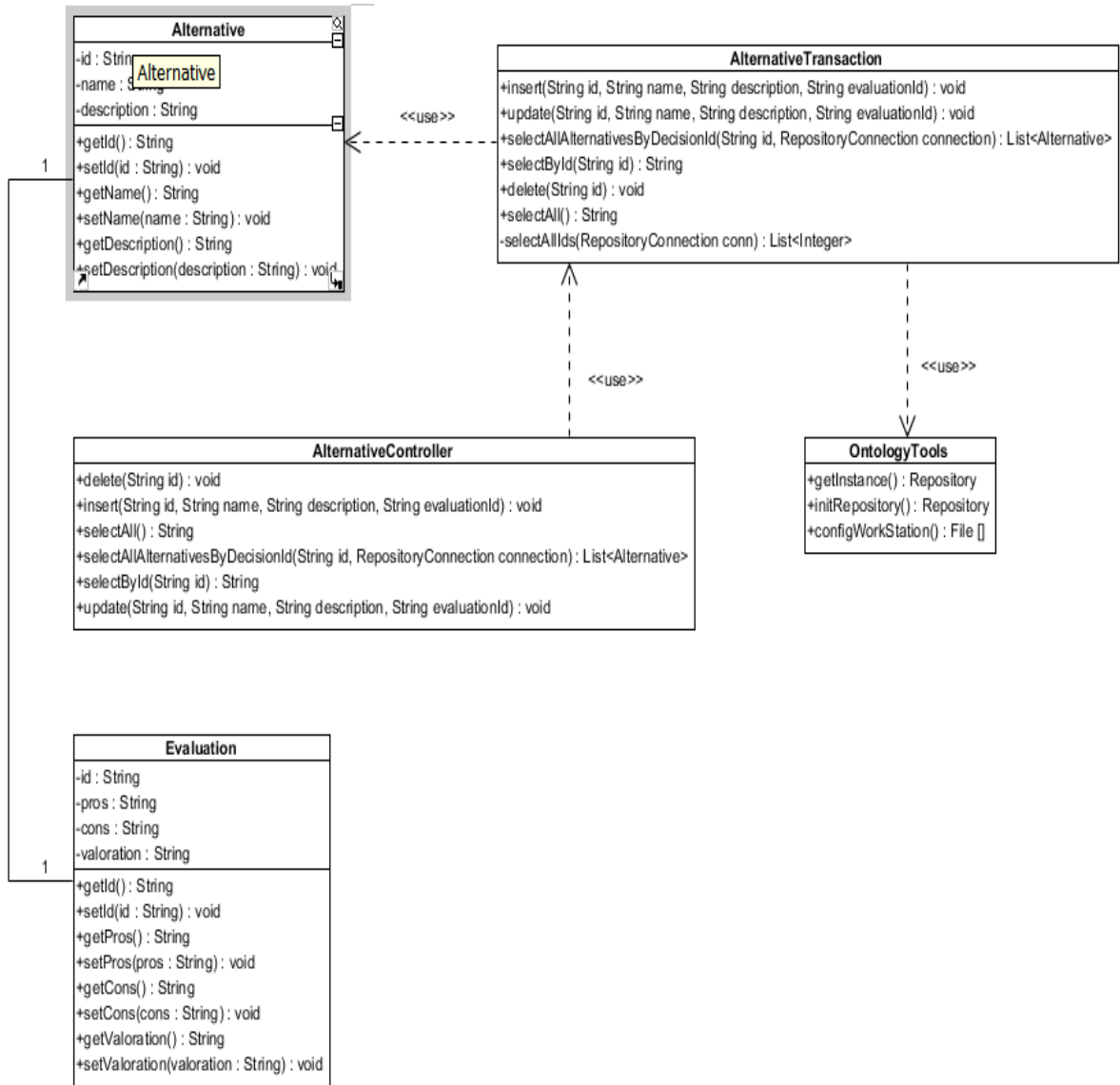
Figura 6. Diagrama de componentes



Fuente: autores

Cada componente sigue una estructura similar a la vista de información mostrada en la figura 7. La estructura cambia dependiendo de la cantidad de los objetos en propiedad en la ontología, por ejemplo, en la figura 6 la clase Alternative y Evaluation se relacionan mediante del objeto de propiedad linked to en la ontología, aquí la Clase Alternative tiene como atributo a la clase Evaluation y viceversa.

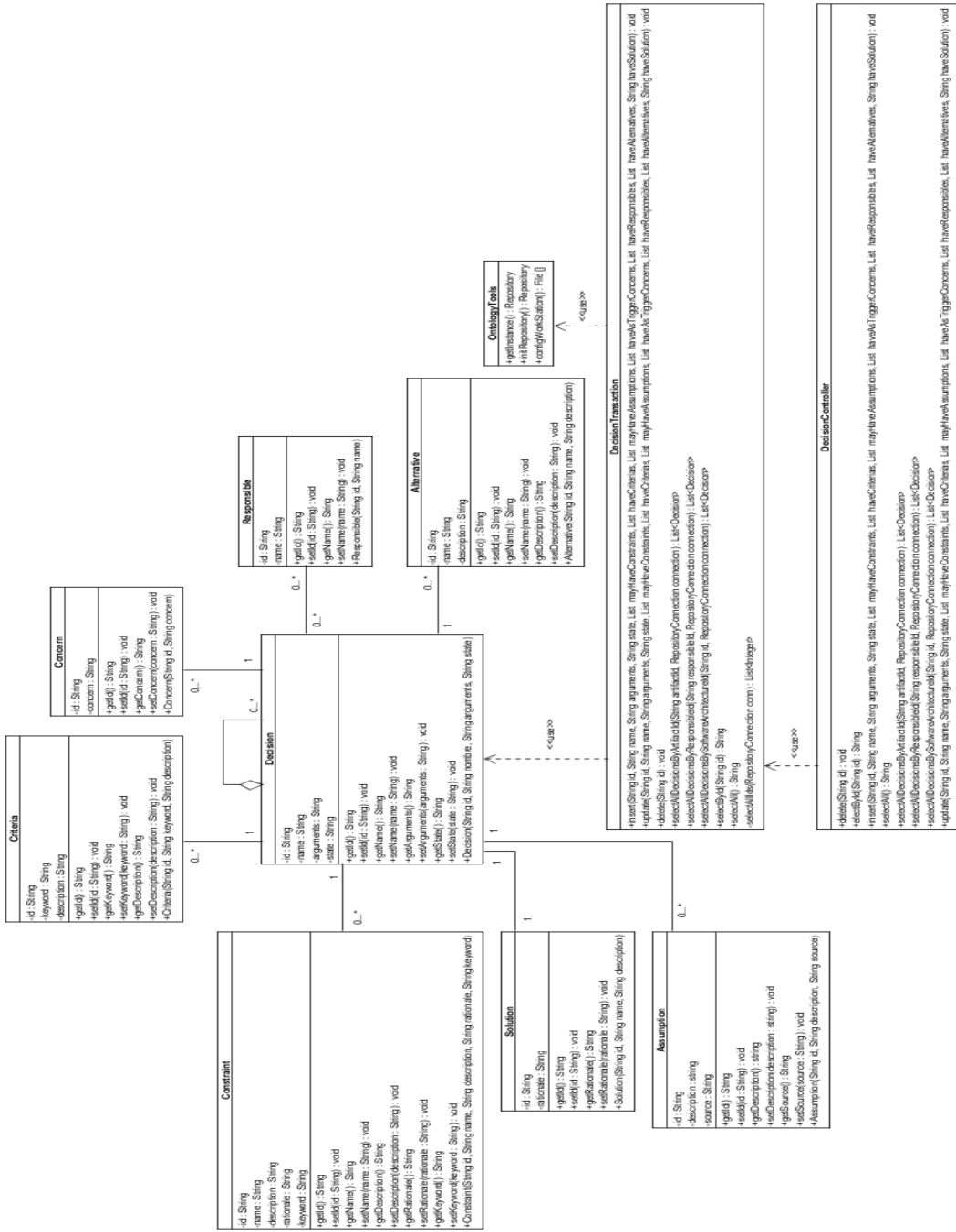
Figura 7. Vista de información del componente Process Alternative Transaction



Fuente: autores

La clase AlternativeTransaction y en general cualquier clase Transaction implementa métodos estáticos para trabajar con una clase en particular de la ontología, en esta oportunidad es la clase Alternative. Los siguientes métodos son comunes en todas las clases Transaction: selectAll y selectById. La figura 8 muestra un ejemplo más complejo de las estructuras de clases, el componente Process Decision Transaction.

Figura 8. Vista de información del componente Process Decisión Transaction

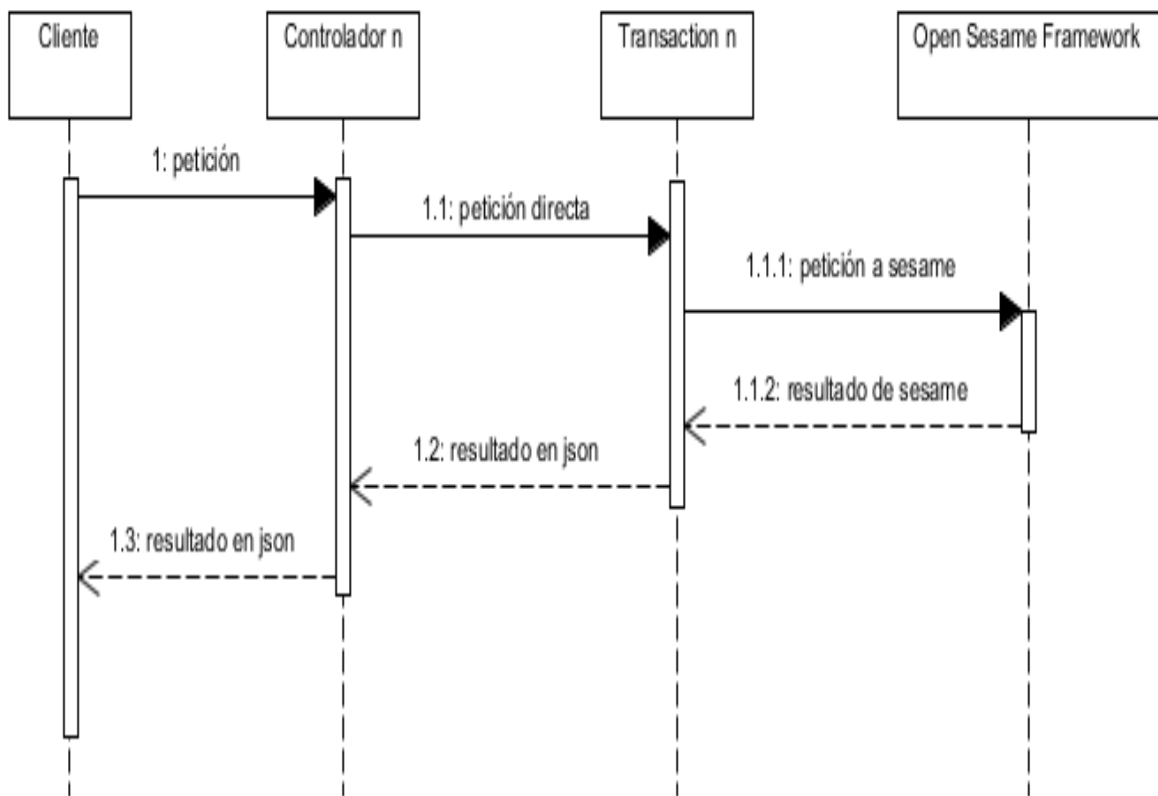


Fuente: autores

5.3.2. Implementación de las vistas. Las vistas se desarrollaron utilizando angular js. Se decidió crear una única página web sobre la cual todo va a mostrarse, eliminarse y borrarse, para esto se hicieron uso de directivas de angular como ngIf, ngShow y ngHide. Para hacer honor a la modularidad y no atiborrar uno de los dos únicos archivos jsp (main.jsp y login.jsp) se recurrió a la creación de directivas angular restringidas a elementos asociadas a módulos que responden a los casos de uso, por ejemplo se crea una directiva que se le inyecta una plantilla html para el módulo de consulta, la directiva se referencia en el jsp u otro html como una etiqueta, luego el compilador de angular hará el trabajo.

Se crearon módulos y controladores para implementar el patrón MVC en el front-end, los módulos responden a los casos de uso en la mayoría de ocasiones, en otras se traducen en métodos en el alcance del controlador o del módulo principal de la vista. La figura 9 muestra el diagrama de secuencia de la aplicación.

Figura 9. Diagrama de secuencias general de la aplicación.

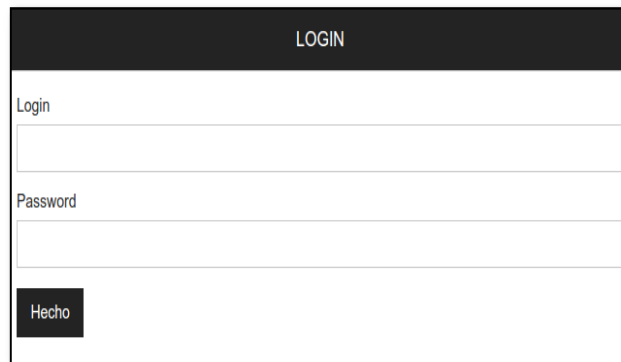


Fuente: autores

5.4 DISEÑO DE INTERFACES

La pantalla de login es lo primero que un usuario verá cuando se ingrese la aplicación

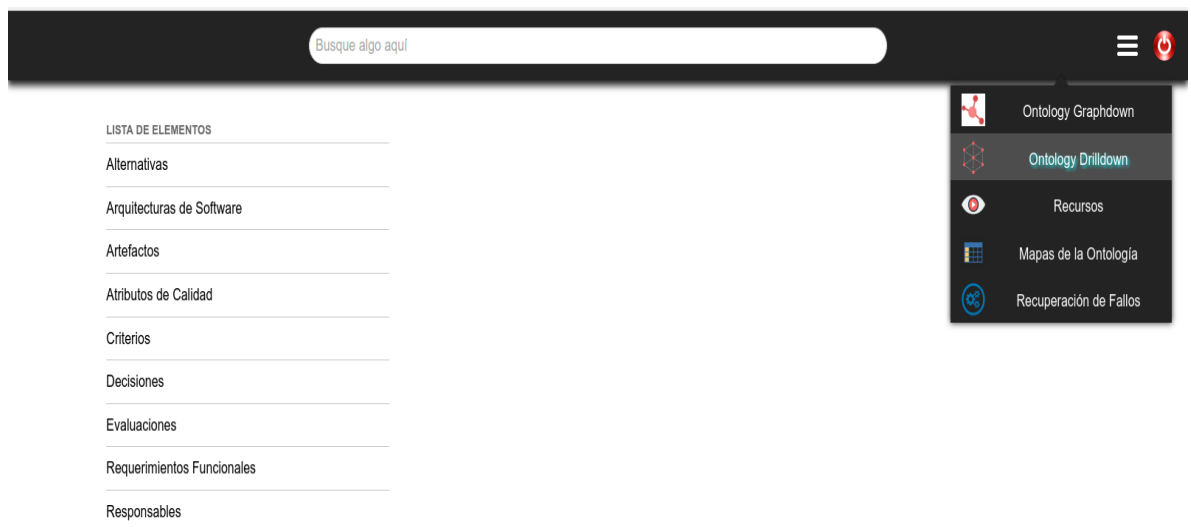
Figura 10. Pantalla de login



Fuente: Aplicación

Una vez autenticado el usuario se mostrará la página principal de la aplicación como se muestra en la figura 11. Aquí el usuario podrá consultar el drilldown de la ontología, ver los mapas de la ontología entre otras cosas.

Figura 11. Pantalla principal de la aplicación



Fuente: Aplicación

Las consultas generales muestran resultados como los mostrados en la figura 12.

Figura 12. Resultado de una consulta general.



Fuente: Aplicación

El módulo de consulta puede ser accedido de diferentes partes de la aplicación, cuando se accede de algún lado un resultado similar al de la figura 13 se verá.

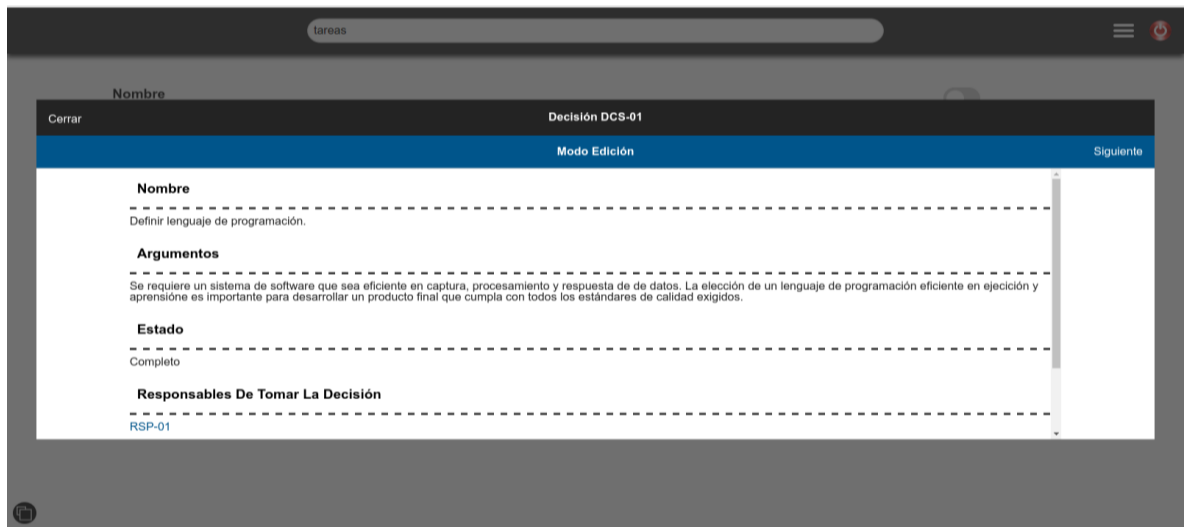
Figura 13. Módulo de consulta.



Fuente: Aplicación

Del módulo de consultas se desprende un módulo de subconsultas que se obtiene al consultar un elemento relacionado, la figura 14 muestra cómo se vería.

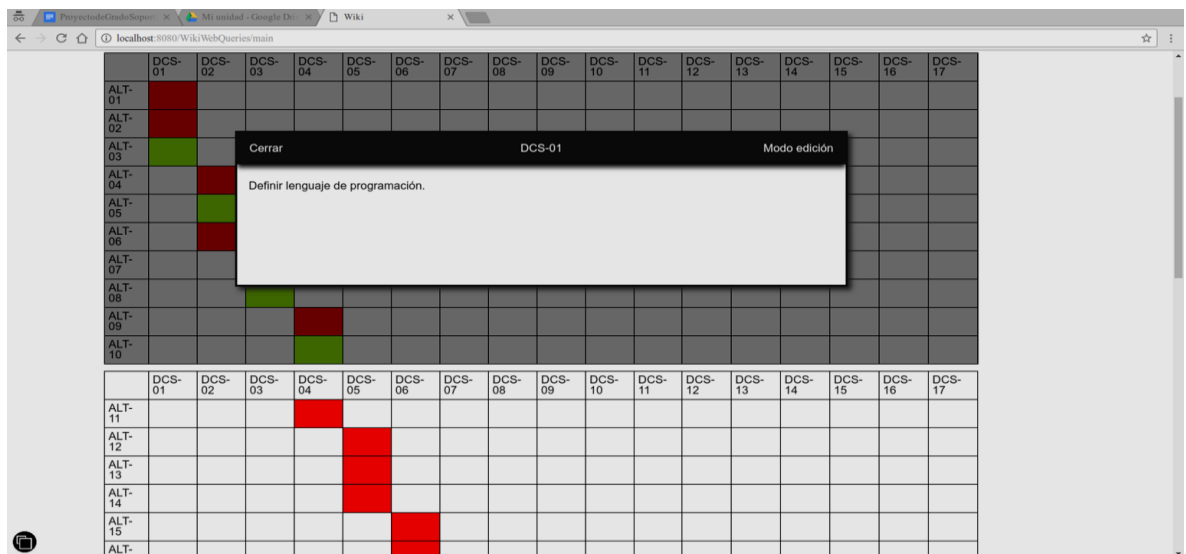
Figura 14. Módulo de subconsultas



Fuente: Aplicación

Es posible ver mapas de la ontología, son representaciones tabulares de la relación existente entre dos clases, la figura 15 muestra un posible resultado.

Figura 15. Mapas de la ontología.



Fuente: Aplicación

5.5 PRUEBAS

En general se hicieron pruebas unitarias y pruebas de integración en toda la aplicación. Para las pruebas de integración se planearon y ejecutaron los siguientes casos de prueba:

Tabla 10. Caso de prueba CP-01.

Identificador	CP-01	
Descripción	Permite verificar el correcto funcionamiento del caso de uso RF-0001 - Consultar individuos a través de coincidencias textuales.	
Precondiciones	<ul style="list-style-type: none">● Copiar la carpeta native_store en [domain]/outreach/, donde domain es la ruta de la carpeta del domino utilizado en glassfish.	
Secuencia		
Pasos	Resultado esperado del sistema	
<ul style="list-style-type: none">● Ingresar al sistema usando las credenciales gmejia y gmejia (login, password).	<ul style="list-style-type: none">● Ingreso exitoso.	
<ul style="list-style-type: none">● En el campo de texto que se encuentra en la parte superior del aplicativo escribir java.	<ul style="list-style-type: none">● En la parte inferior del aplicativo se deberá mostrar el individuo ALT-02 de la siguiente manera: "Alternativa ALT-02".	

Fuente: Autores.

Tabla 11. Caso de prueba CP-02.

Identificador	CP-02	
Descripción	Permite verificar el correcto funcionamiento del caso de uso RF-0002 - Consultar todos los cambios realizados sobre los individuos.	
Precondiciones	<ul style="list-style-type: none"> ● Copiar la carpeta native_store en [domain]/outreach/, donde domain es la ruta de la carpeta del domino utilizado en glassfish. 	
Secuencia		
Pasos	Resultado esperado del sistema	
<ul style="list-style-type: none"> ● Ingresar al sistema con las credenciales gmejia, gmejia (login, password). 	<ul style="list-style-type: none"> ● Ingreso exitoso. 	
<ul style="list-style-type: none"> ● En el menú del aplicativo hacer clic en Recuperación de Fallos. 	<ul style="list-style-type: none"> ● Se debe mostrar una línea temporal con los siguientes cambios CHNG-01, CHNG-02. ● Una lista con las clases definidas en la ontología: alternativa, artefactos, supuestos, criterios, preocupaciones, restricciones, decisiones, evaluaciones, escenarios de calidad, escenarios de operación, responsable, arquitectura de software. 	

<ul style="list-style-type: none"> ● Seleccionar la clase decisión. 	<ul style="list-style-type: none"> ● La línea temporal se debe modificar con los cambios realizados sobre los individuos de la clase decisión, en este caso CHNG-02. ● Debe aparecer otra lista debajo de la lista de en donde se muestre los datos de propiedad asociados a la clase decisión: nombre, argumentos y estado.
<ul style="list-style-type: none"> ● Seleccionar el elemento nombre de la segunda lista. 	<ul style="list-style-type: none"> ● La línea temporal debe quedar intacta.
<ul style="list-style-type: none"> ● Seleccionar el elemento argumentos de la segunda lista. 	<ul style="list-style-type: none"> ● La línea temporal debe quedar vacía.
<ul style="list-style-type: none"> ● Hacer clic en el elemento nombre, luego hacer clic el cambio CHNG-02. 	<ul style="list-style-type: none"> ● Se debe mostrar una ventana emergente o modal con el cambio realizado en su momento y el estado actual del atributo nombre de la decisión DCS-09.

Fuente: Autores.

Tabla 12. Caso de prueba CP-03.

Identificador	CP-03
Descripción	Permite verificar el correcto funcionamiento del caso de uso RF-0003 - Consultar individuo.
Precondiciones	<ul style="list-style-type: none"> ● Copiar la carpeta native_store en [domain]/outreach/, donde domain es la ruta de la carpeta del domino utilizado en glassfish.

Secuencia	
Pasos	Resultado esperado del sistema
<ul style="list-style-type: none"> ● Ingresar al sistema por medio de las credenciales dpulido, dpulido (login, password). 	<ul style="list-style-type: none"> ● Ingreso exitoso.
<ul style="list-style-type: none"> ● En el menú del sistema hacer clic en ontology drilldown, hacer clic en alternativas, luego en ALT-01. 	<ul style="list-style-type: none"> ● Se debe mostrar toda la información asociada al individuo ALT-01 junto con la evaluación asociada EVT-01.
<ul style="list-style-type: none"> ● En el menú del sistema hacer clic en ontology graphdown. 	<ul style="list-style-type: none"> ● Una lista izquierda con las clases de la ontología: alternativas, criterios, evaluaciones, preocupaciones, decisiones, escenarios de operación, escenarios de calidad, supuestos, artefactos, responsables, vistas y restricciones. ● Un botón en la parte inferior que dice graficar.
<ul style="list-style-type: none"> ● Hacer clic en alternativas y luego en el botón graficar. 	<ul style="list-style-type: none"> ● En la parte derecha un conjunto de nodos nombrados desde ALT-01 hasta ALT-38.
<ul style="list-style-type: none"> ● Hacer clic en el nodo ALT-01. 	<ul style="list-style-type: none"> ● Debe aparecer un modal o ventana emergente con la información del individuo ALT-01 junto con las opciones: modo edición y cerrar.
<ul style="list-style-type: none"> ● Hacer clic en modo edición. 	<ul style="list-style-type: none"> ● Se debe mostrar toda la información asociada al individuo ALT-01 junto con la evaluación asociada EVT-01 en la pantalla principal de la aplicación.

<ul style="list-style-type: none"> ● Deslogearse y volverse a logear con las credenciales gmejia, gmejia (login, password). 	<ul style="list-style-type: none"> ● Deslogeo y logeo exitoso.
<ul style="list-style-type: none"> ● En el menú del sistema hacer clic en recuperación de fallos hacer clic en el cambio CHNTG-01, luego hacer clic en modo edición. 	<ul style="list-style-type: none"> ● Se debe mostrar toda la información asociada al individuo ALT-01 junto con la evaluación asociada EVT-01 en la pantalla principal de la aplicación.

Fuente: Autores.

Tabla 13. Caso de prueba CP-04.

Identificador	CP-04	
Descripción	Permite verificar el correcto funcionamiento del caso de uso RF-0004 - Editar contenido de un individuo.	
Precondiciones	<ul style="list-style-type: none"> ● Copiar la carpeta native_store en [domain]/outreach/, donde domain es la ruta de la carpeta del domino utilizado en glassfish. 	
Secuencia		
Pasos	Resultado esperado del sistema	
<ul style="list-style-type: none"> ● Ingresar al sistema con las credenciales gmejia, gmejia (login, password). 	<ul style="list-style-type: none"> ● Ingreso exitoso. 	
<ul style="list-style-type: none"> ● Consultar individuo ALT-01 en cualquier parte del aplicativo (ver el caso de prueba CP-03). 	<ul style="list-style-type: none"> ● Se debe mostrar la información del individuo junto con una lista de los individuos relacionados a este. Los datos de propiedad (no todos) deben tener un elemento switch que esta desactivado por defecto 	

<ul style="list-style-type: none"> ● Activar el switch en el dato de propiedad descripción. 	<ul style="list-style-type: none"> ● El contenido del dato de propiedad debe pasar al modo edición (un área de texto con diversas funciones de edición) conservando su contenido “El jefe de la misión envía un comando de cancelación desde el Centro de Comando y Control”.
<ul style="list-style-type: none"> ● Cambie la descripción de la alternativa por: “El jefe de la misión envía un comando de cancelación desde el Centro de Comando ubicado en la atalaya este”. 	<ul style="list-style-type: none"> ● Debe aparecer temporalmente la pantalla en gris impidiendo realizar acciones sobre la página mientras se guarda los cambios. El apartado de la descripción debe pasar a su estado original con el nuevo texto escrito.

Fuente: Autores.

Tabla 14. Caso de prueba CP-05.

Identificador	CP-05	
Descripción	Permite verificar el correcto funcionamiento del caso de uso RF-0005 - Cargar imágenes desde disco.	
Precondiciones	<ul style="list-style-type: none"> ● Copiar la carpeta native_store en [domain]/outreach/, donde domain es la ruta de la carpeta del domino utilizado en glassfish. 	
Secuencia		
Pasos	Resultado esperado del sistema	

<ul style="list-style-type: none"> ● Ingresar al sistema con las credenciales gmeja, gmeja (login, password). 	<ul style="list-style-type: none"> ● Ingreso exitoso.
<ul style="list-style-type: none"> ● Consultar el individuo ALT-01 en cualquier parte del aplicativo (ver el caso de prueba CP-03). 	<ul style="list-style-type: none"> ● Se debe mostrar la información del individuo junto con una lista de los individuos relacionados a este. Los datos de propiedad (no todos) deben tener un elemento switch que esta desactivado por defecto
<ul style="list-style-type: none"> ● Activar el switch asociado al dato de propiedad descripción. 	<ul style="list-style-type: none"> ● El contenido del dato de propiedad debe pasar al modo edición (un área de texto con diversas funciones de edición) conservando su contenido “El jefe de la misión envía un comando de cancelación desde el Centro de Comando y Control”.
<ul style="list-style-type: none"> ● Seleccionar el ícono que se asemeja a una imagen. 	<ul style="list-style-type: none"> ● Se debe mostrar una lista con las opciones: cargar archivo de disco local, cargar de url y cargar archivo existente.
<ul style="list-style-type: none"> ● Seleccionar la opción Cargar archivo de disco local. 	<ul style="list-style-type: none"> ● Debe aparecer un modal con un campo para el nombre de la imagen y un botón para seleccionar un archivo.
<ul style="list-style-type: none"> ● Hacer clic donde dice hecho. 	<ul style="list-style-type: none"> ● El campo del nombre deberá aparecer en rojo.

<ul style="list-style-type: none"> ● Ingresar el nombre imagen1 y luego hacer clic en hecho. 	<ul style="list-style-type: none"> ● Se deberá mostrar en rojo el botón de seleccionar un archivo.
<ul style="list-style-type: none"> ● Seleccionar un archivo con formato diferente a jpg, jpeg, gif y png. Hacer clic en hecho. 	<ul style="list-style-type: none"> ● Deberá aparecer una advertencia que diga que solo debe subir un archivo de formato jpg, jpeg, png o gif.
<ul style="list-style-type: none"> ● Seleccionar un archivo de imagen válido y luego hacer clic en hecho. 	<ul style="list-style-type: none"> ● La imagen se debe cargar en el área de texto. En la carpeta outreach en el dominio del servidor glassfish debe aparecer una carpeta images con la imagen cargada cuyo nombre de archivo es imagen1 para la imagen y su extensión es el de la imagen cargada.

Fuente: Autores.

Tabla 15. Caso de prueba CP-06.

Identificador	CP-06	
Descripción	Permite verificar el correcto funcionamiento del caso de uso RF-0006 - Cargar imágenes por URL.	
Precondiciones	<ul style="list-style-type: none"> ● Copiar la carpeta native_store en [domain]/outreach/, donde domain es la ruta de la carpeta del domino utilizado en glassfish. 	
Secuencia		
Pasos	Resultado esperado del sistema	

<ul style="list-style-type: none"> ● Ingresar al sistema con las credenciales gmejia, gmejia (login, password). 	<ul style="list-style-type: none"> ● Ingreso exitoso.
<ul style="list-style-type: none"> ● Consultar el individuo ALT-01 en cualquier parte del aplicativo (ver el caso de prueba CP-03). 	<ul style="list-style-type: none"> ● Se debe mostrar la información del individuo junto con una lista de los individuos relacionados a este. Los datos de propiedad (no todos) deben tener un elemento switch que esta desactivado por defecto
<ul style="list-style-type: none"> ● Activar el switch asociado al dato de propiedad descripción. 	<ul style="list-style-type: none"> ● El contenido del dato de propiedad debe pasar al modo edición (un área de texto con diversas funciones de edición) conservando su contenido “El jefe de la misión envía un comando de cancelación desde el Centro de Comando y Control”.
<ul style="list-style-type: none"> ● Seleccionar el ícono que se asemeja a una imagen. 	<ul style="list-style-type: none"> ● Se debe mostrar una lista con las opciones: cargar archivo de disco local, cargar de url y cargar archivo existente.
<ul style="list-style-type: none"> ● Seleccionar la opción cargar de url. 	<ul style="list-style-type: none"> ● Debe aparecer dos campos de texto uno para el nombre y otro para la url.

<ul style="list-style-type: none"> ● Hacer clic donde dice hecho. 	<ul style="list-style-type: none"> ● El campo del nombre deberá aparecer en rojo.
<ul style="list-style-type: none"> ● Ingresar el nombre imagen2 y luego hacer clic en hecho. 	<ul style="list-style-type: none"> ● Se deberá mostrar una alerta diciendo que el enlace suministrado está roto.
<ul style="list-style-type: none"> ● Ingresar una dirección de url errónea de una imagen, luego hacer clic en hecho. 	<ul style="list-style-type: none"> ● Se deberá mostrar una alerta diciendo que el enlace suministrado está roto.
<ul style="list-style-type: none"> ● Ingresar una dirección de url correcta de una imagen, luego hacer clic en hecho. 	<ul style="list-style-type: none"> ● La imagen se debe cargar en el área de texto. En la carpeta outreach en el dominio del servidor glassfish debe aparecer una carpeta images con la imagen cargada cuyo nombre de archivo es imagen1 para la imagen y su extensión es el de la imagen cargada.
<ul style="list-style-type: none"> ● Seleccionar un archivo de imagen válido y luego hacer clic en hecho. 	<p>La imagen se debe cargar en el área de texto. En la carpeta outreach en el dominio del servidor glassfish debe aparecer una carpeta registers con un archivo data.json con un contenido similar a: <code>[[{"type":"url","content": "<span img=\"imagen2\" h=\"285px\" w=\"200px\" url=\"[dirección_url_suministrada]\", \"name\": \"imagen2\", \"references\": [\"/Alter native/\"]"}]]</code></p>

Fuente: Autores.

Tabla 16. Caso de prueba CP-07.

Identificador	CP-07	
Descripción	Permite verificar el correcto funcionamiento del caso de uso RF-0007 - Cargar imágenes por URL.	
Precondiciones	<ul style="list-style-type: none"> ● Copiar la carpeta native_store en [domain]/outreach/, donde domain es la ruta de la carpeta del domino utilizado en glassfish. ● Copiar la carpeta images en [domain]/outreach/, donde domain es la ruta de la carpeta del domino utilizado en glassfish. ● Copiar la carpeta registers en en [domain]/outreach/, donde domain es la ruta de la carpeta del domino utilizado en glassfish. 	
Secuencia		
● Ingresar al sistema con las credenciales gmejia, gmejia (login, password).	● Ingreso exitoso.	
● Consultar el individuo ALT-01 en cualquier parte del aplicativo (ver el caso de prueba CP-03).	● Se debe mostrar la información del individuo junto con una lista de los individuos relacionados a este. Los datos de propiedad (no todos) deben tener un elemento switch que esta desactivado por defecto	
● Activar el switch asociado al dato de propiedad descripción.	● El contenido del dato de propiedad debe pasar al modo edición (un área de texto con diversas funciones de edición) conservando su contenido "El jefe de la misión envía un comando de cancelación desde el Centro de Comando y Control".	

<ul style="list-style-type: none"> ● Seleccionar el ícono que se asemeja a una imagen. 	<ul style="list-style-type: none"> ● Se debe mostrar una lista con las opciones: cargar archivo de disco local, cargar de url y cargar archivo existente.
<ul style="list-style-type: none"> ● Seleccionar la opción cargar archivo existente. 	<ul style="list-style-type: none"> ● Debe aparecer un modal con dos botones que permitan intercambiar entre imágenes locales y por url: imágenes locales e imágenes por url.
<ul style="list-style-type: none"> ● Hacer clic donde dice hecho. 	<ul style="list-style-type: none"> ● Debe aparecer una alerta diciendo que no ha seleccionado imagen.
<ul style="list-style-type: none"> ● Seleccionar imágenes locales. 	<ul style="list-style-type: none"> ● Se debe cargar una imagen cuyo nombre es imagen1 y dimensión es 1920x1080.
<ul style="list-style-type: none"> ● Seleccionar la imagen1, luego hacer clic en hecho. 	<ul style="list-style-type: none"> ● La imagen1 se debe cargar en el área de texto.
<ul style="list-style-type: none"> ● Seleccionar imágenes por url. 	<ul style="list-style-type: none"> ● Se debe cargar una única imagen con nombre imagen2 y dimensión 200x285.
<ul style="list-style-type: none"> ● Seleccionar la imagen2 y luego hacer clic en hecho. 	<ul style="list-style-type: none"> ● La imagen se debe cargar en el área de texto.

Fuente: Autores.

Tabla 17. Caso de prueba CP-08

Identificador	CP-08	
Descripción	Permite verificar el correcto funcionamiento del caso de uso RF-0008 - Consultar individuo relacionado a otro.	
Precondiciones	<ul style="list-style-type: none"> ● Copiar la carpeta native_store en [domain]/outreach/, donde domain es la ruta de la carpeta del domino utilizado en glassfish. 	
Secuencia		
Pasos	Resultado esperado del sistema	
<ul style="list-style-type: none"> ● Ingresar al sistema con las credenciales gmejia, gmejia (login, password). 	<ul style="list-style-type: none"> ● Ingreso exitoso. 	
<ul style="list-style-type: none"> ● Consultar el individuo ALT-01 en cualquier parte del aplicativo (ver el caso de prueba CP-03). 	<ul style="list-style-type: none"> ● Se debe mostrar la información del individuo junto con una lista de los individuos relacionados a este. Los datos de propiedad (no todos) deben tener un elemento switch que esta desactivado por defecto 	
<ul style="list-style-type: none"> ● Hacer clic en EVT-01. 	<ul style="list-style-type: none"> ● Debe aparecer un modal con la información del individuo seleccionado. Como alternativa relacionada debe aparecer ALT-01. 	

Fuente: Autores.

6. VALIDACIÓN

6.1 OBJETIVOS, HIPÓTESIS Y VARIABLES

Se necesita validar la aplicación desde tres factores muy importantes: La eficacia, la eficiencia y el grado de satisfacción del usuario en cuanto a la comprensión de las decisiones de arquitecturas de software y uso de la aplicación.

No todas las empresas colombianas o extranjeras a nivel general disponen de arquitectos de software que se encarguen de la primera de etapa de desarrollo de un sistema informático, por experiencia de algunos en el medio el proceso de diseño es usualmente desarrollado por los mismos desarrolladores o equipo de trabajo, en especial si en la empresa en que trabajan aplican metodologías ágiles de desarrollo como SCRUM, donde se centran en el desarrollo de tareas relativamente pequeñas en un tiempo no mayor de un mes por ciclo de SCRUM (sprint), donde las tareas de administración o de control se hacen a lo largo del ciclo a través de reuniones cortas (daily). El anterior enfoque no permite realizar un proceso formal de desarrollo de arquitectura de software estimando todas, las posibles alternativas de construcción, o la mayoría cuan menos, que permita desarrollar una solución cercana a la óptima desde el principio y no recurriendo a la solución de posibles futuras incidencias que se fueran a presentar tras no hacer un análisis lo suficiente profundo de la situación, necesidades y en especial las alternativas de solución a la necesidades (a partir de aquí se mencionarán a cualquiera que tenga cierto papel en la creación de algún diseño de software o arquitectura como diseñador).

Según lo anterior mencionado nos es necesario evaluar la aplicación que en el presente proyecto es expuesto desde el punto de vista de la eficiencia, la eficacia y el grado de satisfacción de uso de los operarios en contraste con el uso tradicional de documentación física (en papel) que suele hacerse en las empresas. Para esto se planeó realizar dos diseños experimentales factoriales 2x2, dos factores de validación por dos grupos sobre los cuales se realizará el experimento donde: El primer experimento, trabaja con dos grupos que utilizarán la aplicación:

El segundo, trabajará con dos grupos diferentes al primero que no utilizarán la aplicación.

Tabla 18. Diseño factorial 2x2 para uso de la aplicación.

Diseñadores Expertos	Diseñadores Novatos	
Grupo 2	Grupo 1	Eficiencia
Grupo 2	Grupo 1	Eficacia

Fuente: autores.

Tabla 19. Diseño factorial 2x2 son uso de la aplicación.

Diseñadores Expertos	Diseñadores Novatos	
Grupo 4	Grupo 3	Eficiencia
Grupo 4	Grupo 3	Eficacia

Fuente: autores.

Cada grupo que conformará cada experimento se clasificará como “Participantes Novatos” o “Participantes Expertos”. Se considerarán los siguientes filtros de clasificación:

- Nivel de capacitación de los diseñadores en temas de diseño y arquitectura.
- Nivel de experiencia (tiempo) que tenga el diseñador diseñando software.
- Número de proyectos de software extensos en el que haya participado activamente y que hayan requerido el diseño riguroso de una arquitectura de software.
- Si el diseñador es o fue alguna vez líder de proyecto o desarrollo.
- Si alguna vez el diseñador ha liderado o participado en proyectos de reestructuración de antiguas arquitecturas.

A cada grupo en función de unos escenarios definidos se les pedirá que comprendan la misma arquitectura de software pero que dependiendo del experimento se les entregará en la aplicación o en un formato impreso. Por ejemplo, según las tablas 1 y 2 los grupos 1 y 2 recibirán la documentación de la arquitectura de software en la aplicación, por ende, estos se desenvolverán con

esta; En cambio los grupos 3 y 4, se les entregará la misma documentación pero en formato impreso.

Cada grupo al final tendrá la labor de realizar algún cambio sobre la arquitectura, este cambio será lo suficientemente pequeño como para que se pueda evaluar como correcto o incorrecto.

En cuanto al factor de satisfacción se recurrirá a una encuesta descriptiva con preguntas de alternativa que servirán para evaluar el nivel de satisfacción del diseñador de la aplicación tras usarla.

En este contexto definimos las siguientes hipótesis en base a las necesidades del proyecto que en el documento se especifica y a los factores de validación:

- H1': Los grupos que estén utilizando la aplicación presentarán mejores tiempos en los desafíos propuestos en comparación con el grupo equivalente que no esté usando la aplicación.
- H0': Los grupos que estén utilizando la aplicación no presentarán mejores tiempos en los desafíos propuestos en comparación con el grupo equivalente que no esté usando la aplicación.
- H1'': Los grupos que estén utilizando la aplicación presentarán un mayor o igual número de resultados correctos en los desafíos propuestos en comparación con el grupo equivalente que no esté usando la aplicación.
- H0'': Los grupos que estén utilizando la aplicación presentarán menor número de resultados correctos en los desafíos propuestos en comparación con el grupo equivalente que no esté usando la aplicación.
- H1''': Los resultados de la encuesta de satisfacción de que se realizará a los diseñadores que utilizaron la aplicación serán según la escala de medición utilizada, positivos.
- H0''': Los resultados de la encuesta de satisfacción de que se realizará a los diseñadores que utilizaron la aplicación serán según la escala de medición utilizada, negativos o neutrales

6.2 DISEÑO EXPERIMENTAL

Para probar las hipótesis planteadas desarrollará un proyecto piloto que incluye ingenieros de sistemas y arquitectos de software de diversas empresas. Se considera lo siguiente:

- Participantes. La prueba se desarrollará a diferentes ingenieros de sistemas arquitectos de software, en general profesionales en sistemas con cierta experiencia en el diseño de software.

- Objeto de estudio. Se propone la construcción de un sistema de software para un Sistema de Monitoreo y Control remoto de vuelo de vehículo espacial no tripulado (e.g., un cohete). La figura 1 muestra el diagrama de contexto de la arquitectura, los principales subsistemas son:

- o Vehículo espacial maneja los instrumentos y computadores a bordo de los vehículos de misión. Entre los instrumentos están los sensores que recolectan información del ambiente como velocidad del viento, temperatura, localización, altura y cantidad de combustible.

- o Plataforma en Tierra es una estación intermedia entre Vehículo espacial y Centro Comando Control Misión (C3) y tiene como responsabilidades monitorear, compactar, procesar y emitir información (en forma de tramas) desde el Vehículo espacial al C3. Recibe la información por medio de antenas receptoras de información telemétrica desde Vehículo espacial.

- o Centro Comando Control Misión (C3) recibe información de la plataforma en tierra y la despliega a operarios y al jefe de misión. Además, tiene como responsabilidad difundir datos específicos del desarrollo de la misión y transmitir en vídeo el desarrollo de la misma a público general.

A continuación, se expresan algunas de las preocupaciones en torno de la definición de la arquitectura y diseño.

- o Se debe ofrecer información a los usuarios responsables en tiempo real para evaluar el avance de la misión y tomar decisiones acerca de su continuidad o no.

- o La integridad de la información recolectada durante la misión es fundamental para la planeación de futuras misiones.

- o Es fundamental el procesamiento de todas las tramas con datos provenientes del vehículo espacial.

- Bloqueos. Para evitar confusiones y ambigüedades por parte de los

participantes del experimento se les realizará una capacitación completa del objetivo y proceso a desarrollar durante el experimento. Para aquellos participantes que vayan a utilizar la aplicación se les capacitará en su uso. Para asegurar el mayor grado de objetividad en los resultados después de clasificar a los participantes se asignará de manera aleatoria a grupos de experimentos definidos.

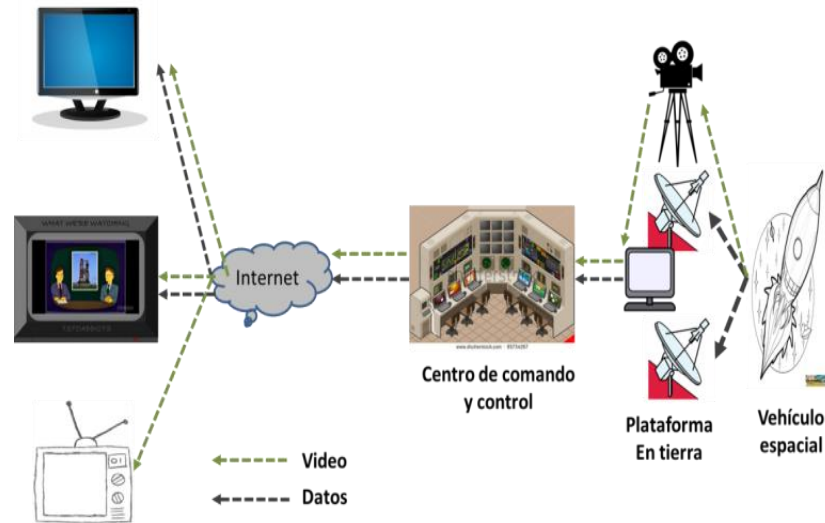
- Instrumentación. El experimento será realizado en tres fases.

- o Capacitación. Consiste en la presentación de los conceptos y procesos que se llevarán a cabo durante el experimento a cada uno de los grupos, además de capacitar en el uso de la aplicación a aquellos que, por el sorteo, les haya tocado participar en ese experimento.

Se definieron los siguientes instrumentos.

- Cuestionarios para la clasificación de los participantes.
- Sorteo para asignación de participantes
- Capacitación del problema y el proceso a los participantes.
- Capacitación de la aplicación a aquellos participantes que les haya tocado participar en el experimento con uso de la aplicación.
- Experimento. A los participantes se les otorgó una arquitectura de software (en la aplicación o impreso dependiendo del tipo de experimento) y se les otorgó una tarea o desafío que deberán completar. Las siguientes actividades e instrumentos fueron propuestas:
 - o Entrega de la documentación de la arquitectura, o la aplicación con la documentación dependiendo del participante.
 - o Entrega de la descripción del problema a manera de cuestionario de preguntas de alternativa.
 - Post Experimento. Se recolectará el tiempo requerido desde que se entrega el cuestionario hasta que el participante lo devuelve, también se registró que participante lo entregó y si este participó en el experimento que incluye a la aplicación se le facilitará una encuesta de satisfacción para valorar otros datos importantes. Se consideran los siguientes instrumentos y actividades:
 - o Entrega y diligencia del cuestionario de satisfacción.

Figura 16. Diagrama de contexto del objeto de estudio



Fuente: SAD – Sistema de Control y Comando de Misiones Aeroespaciales ⁵²

6.3 EJECUCIÓN

6.3.1 Preparación del experimento. Se reunirá a los diseñadores que participarán en la encuesta para otorgarles el cuestionario de clasificación para que sea respondido. A medida que se vaya entregando se irá evaluando saber a qué grupo pertenece el diseñador en cuestión. Una vez clasificados a todos se proseguirá a realizar un sorteo para saber a qué experimento pertenecerá cada diseñador. Todo lo anterior se estima que se realizará en una hora.

Lo siguiente requerido fue la capacitación general esta se desarrollará en un tiempo cercano a la hora y media. En esta capacitación general se describe:

- Los objetivos del experimento.
- Los objetivos de la clasificación previa
- El proceso general del experimento.
- Las tareas que posteriormente tendrán que realizar.

Luego de haber culminado la capacitación general se recurrió a cada persona capacitada individualmente y se le preguntó acerca lo que se requiere hacer para este experimento, si alguno no responde correctamente indica que no logró entender a plenitud lo requerido para el experimento, es entonces que se le hará la respectiva aclaración.

⁵²SAD – Sistema de Control y Comando de Misiones Aeroespaciales [en línea]
Disponible en: <http://archidecisions.referata.com/>

Una vez culminada la capacitación se prosigue a realizar la capacitación de la aplicación a aquellos diseñadores que hayan quedado asignados, por el sorteo, al experimento que requiere el uso de la aplicación. Esta capacitación durará cerca de dos horas. Para corroborar que se haya entendido se solicitará la solución de tareas muy puntuales acerca de lo expuesto.

Una vez considerado que la capacitación se haya completado satisfactoriamente se proseguirá a la etapa de ejecución.

6.3.2 Ejecución del experimento. Se ejecutará lo expuesto en el diseño experimental para ello se citará a los individuos de prueba en la Universidad Piloto de Colombia. Se dividirán en los grupos de aquellos que utilizarán la aplicación y aquellos que no la usarán en dos salas de sistemas previamente apartadas. Aquí comienza el juego.

7. CONCLUSIONES Y TRABAJO FUTURO

Trabajos relativamente recientes de administración de decisiones que involucran ontologías se centran en ciertos aspectos de las mismas, Krutchen por ejemplo se centra en los tipos de decisiones y lo que mínimo requerido para tomar una decisión de dicho tipo o Tyree aunque no modela una ontología expone la información mínima que debe tener para que esta esté bien documentada y se logre comprender correctamente. Gran parte de esos trabajos se centran específicamente en algún elemento en particular de la arquitectura y a partir de eso se desarrolla brindando soluciones muy específicas. La ontología que se propuso aquí equilibra lo general y lo específico para que junto con la wiki se brinde una visión global acerca de las decisiones tomadas y el proceso seguido con el suficiente grado de profundidad como para brindar detalles del mismo.

La aplicación de la wiki se desarrolló como una herramienta para la representación y distribución de decisiones de diseño como conocimiento de información, clave para facilitar la comprensión en el momento justo, buscando evitar procesos largos en aras de consultar lo que se necesita cuando se necesita.

Los filtros que se exponen es una forma de cómo se logró dicho cometido. Esta wiki es la herramienta para la administración de conocimiento que debe ser probada para verificar si realmente el modelo de decisiones, la ontología desarrollada y la misma wiki semántica cumple las expectativas expuestas en las hipótesis del presente proyecto de grado, por eso se considera como trabajo futuro la ejecución de la validación propuesta en la sección 6, para luego analizar los resultados y obtener conclusiones más específicas sobre el desempeño de los individuos usando nuestra alternativa frente a aquellos que no la utilizan.

BIBLIOGRAFÍA

BOSCH Jan, JANSEN Anton. Software Architecture as a Set of Architectural Design Decisions. p.2. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: www.ics.uci.edu/~andre/ics223w2006/jansenbosch.pdf

BARNARD. Sociología de las organizaciones. 1938, [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: datateca.unad.edu.co/contenidos/.../libro_libre_Sociologia_de_las_organizaciones.pdf

BARRAZA, Fernando. Decisiones de arquitectura de software. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://prezi.com/shgc48bzwo0u/decisiones-en-el-diseno-arquitectonico/>

GONZÁLEZ, Oscar et al. MONO+KM: Knowledge Management in Collaborative Project Development. Ingeniería y Universidad: Engineering for Development, [S.l.], v. 20, n. 2, p. 267-302, jun. 2016. ISSN 2011-2769. Available at: <http://revistas.javeriana.edu.co/index.php/iyu/article/view/14862>. Date accessed: 15 Dec. 2016. doi:<http://dx.doi.org/10.11144/Javeriana.iyu20-2.mkkm>.

BENAROCH, M. citado por: ZAPATA Carlos, GIRALDO Gloria y URREGO Germán, "Specifying Local Ontologies in Support of Semantic Interoperability of Distributed Inter-organizational Applications," en Proc. of the 5th Intl. Workshop on Next Generation Inf. Techn. and Systems, Caesarea, Israel, 2002, pp. 90-106. En: Rev. ing. univ. Medellín. No.16 (Jun, 2010). ISSN 1692-3324.

BENCH-CAPON, Citado por: ZAPATA Carlos, GIRALDO Gloria, URREGO Germán. "The Role of Ontologies in the Verification and Validation of Knowledge Based Systems: Las ontologías en la ingeniería de *software*: un acercamiento de dos grandes áreas del conocimiento. Usa: 9th International Workshop on Database and Expert Systems Applications (DEXA), 1998. P. 20

BOEHM Barry. A Spiral Model of Software Development and Enhancement. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: csse.usc.edu/TECHRPTS/1988/usccse88-500/usccse88-500.pdf

DALKIR, J. Knowledge Management in Theory and Practice, CA: Elsevier Inc, p. 10. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: [25.https://dianabarbosa.files.wordpress.com/.../knowledge-management-kimiz-dalkir.pdf](https://dianabarbosa.files.wordpress.com/.../knowledge-management-kimiz-dalkir.pdf)

DIALNET. ¿Qué es wiki semántica? ?. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://dialnet.unirioja.es/descarga/articulo/2924515.pdf>

FOWLER Martin. ¿Who Needs an Architect?, pp 2-4. [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: martinfowler.com/ieeeSoftware/whoNeedsArchitect.p

GIRALDO, Isabel. ¿Qué son requerimientos funcionales. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://prezi.com/t7fmr4mkgwym/requerimientos-funcionales>

GUARINO N. Formal Ontology in Information Systems. 4p. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: uosis.mif.vu.lt/.../Guarino98-Formal%20Ontology%20and%20Inf..

GUIOTECA. ¿Qué son los stakeholders?. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://www.guioteca.com/rse/que-son-los-stakeholders/>

IBM citado por Kruchten Philippe. IBM.Rational Unified Process: Best Practices for Software Development Teams: An Ontology of Architectural Design Decisions in software-Intensive Systems. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

KRUCTEN, . Philippe . An Ontology of Architectural Design Decisions in software-Intensive System. 2004, Octubre, [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: <https://philippe.kruchten.com/architecture/>

_____. IEEE ORG.1471-2000 . IEEE Recommended Practice for Architectural Description for Software-Intensive Systems: An Ontology of Architectural Design Decisions in software-Intensive Systems. p.1. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: <http://standards.ieee.org/findstds/standard/1471-2000.html>

MONOGRAFÍAS.COM. Administración del conocimiento. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: www.monografias.com › Administración y Finanzas › Recursos Humanos

MURRAY. J. Knowledge Management in Modern Organizations. San Diego: State University, USA 2005, p. 22

PEDRAZA-GARCIA, Gilberto, Astudillo, Hernán and Correal, Darío. Analysis of design meetings for understanding software architecture decisions," 2014 XL Latin American Computing Conference (CLEI), Montevideo, 2014, pp. 1-10. doi: 10.1109/CLEI.2014.6965159

PEDRAZA-GARCIA, Gilberto, Astudillo, Hernán and Correal, Darío. Modeling Software Architecture Process with a Decision-Making Approach 2014 33rd International Conference of the Chilean Computer Science Society (SCCC), Talca, 2014, pp. 1-6. doi: 10.1109/SCCC.2014.27

PEDRAZA-GARCIA, Gilberto, Astudillo, Hernán and Correal, Darío. An Approach for Software Knowledge Sharing based on Architectural Decisions, 2016 Latin American Computing Conference (CLEI), Valparaiso, 2016, pp. 1-10.

PÉREZ, Isabel. ¿Qué es Wiki?. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: www.isabelperez.com/taller1/wiki.htm

PROEXPORT. Soluciones corporativas. 2009. [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: [www.talent.upc.edu/ media/Soluciones Corporatives_esp.pdf](http://www.talent.upc.edu/media/Soluciones_Corporatives_esp.pdf)

REUTELSHOFER Jochen; BAUMEISTER Joachim; PUPPE Frank y . Knowwe A Semantic Wiki for Knowledge Semántica Engineering. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: <https://pdfs.semanticscholar.org/.../c09d67fbfd152629aed183bbe>

R. Johan, Capital intelectual: el valor intangible de la empresa, CA: México: Paidós, p 9-18.

RONEN, P. Edna., The Complete Guide to Knowledge Management A Strategic Plan to Leverage Your Company's Intellectual Capital, USA: John Wiley & Sons, pp. 10-17.

ROZANSKI Nick, WOODS Eoin. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. Segunda Edición, pp 2-13. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: www.cs.du.edu/.../architecture/SW-SystemsArch-Workingwith Stake holders Usi.

SG BUZZ. Arquitectura del software. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://sg.com.mx/revista/27/arquitectura-software>

SIGNIFICADOS.COM. Qué significa ontología. [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: <https://www.significados.com/ontologia/>

TYREE, Jeff y AKERMAN Art. Architecture Decisions: Demystifying Architecture. 20 p. [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: https://www.researchgate.net/.../24535_2502_An_Ontology_of_Architectural_Design_

VÁSQUEZ H, DIAZ J. Uso de Ontologías para mapear una Arquitectura de Software con su Implementación, p. 101-106.

VLIET Hans y BOER, Remco. Experiences with Semantic Wikis for Architectural Knowledge Management. [En línea], [consultado el 8 de octubre de 2015]. Disponible en: ieeexplore.ieee.org/abstract/document/5959696/?section=abstract-

WILEY, John y SONS, The Complete Guide to Knowledge Management A Strategic Plan to Leverage Your Company's Intellectual Capital, p. 10-17. [En línea], [consultado el 2 de noviembre de 2016]. Disponible en: apollon1.alba.edu.gr/OKLC2002/Proceedings/pdf.../ID138.pdf

W3C.ES. ¿Qué es WEB Semántica? [En línea], [consultado el 23 de noviembre de 2016]. Disponible en: [www.w3c.es > Documentos y Guías > Guías Breves](http://www.w3c.es/Documentos_y_Guías/Guías_Breves)

_____. Guía Breve de Web Semántica [En línea], [consultado el 8 de octubre de 2015]. Disponible en: [http://www.w3c.es/Divulgacion/GuiasBreves /Web Semantica.](http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica)

WONGTHONGTHAM Pornpit, KHADEER Farookh, CHANG Elizabeth, DILLON Tharam. Multi-site Software Engineering Ontology Instantiations Management Using Reputation Based Decision Making. 2 p.

ZAPATA Carlos, GIRALDO Gloria, URREGO Germán. Las ontologías en la ingeniería de *software*: un acercamiento de dos grandes áreas del conocimiento. En: Rev. ing. univ. Medellín. No.16 (Jun, 2010). ISSN 1692-3324.

ANEXOS

(PLAN ADMINISTRATIVO DE LA INVESTIGACIÓN)

Anexo A. Plan administrativo del proyecto

Los precios mostrados en la tabla dos se basan en las actividades necesarias para el desarrollo de un prototipo de wiki semántica general, adaptable a cualquier propósito de administración del conocimiento de arquitectura de software, es decir, sin la aplicación y la validación del caso de un caso de estudio específico.

Tabla 1. Cotización por actividad de los involucrados.

Precio acumulado	Precio	Actividad
500000	700000	Diseño de la ontología
1350000	850000	Implementación de la ontología
4750000	400000	Pruebas a la ontología en Protégé
2650000	900000	Construcción de la wiki semántica
3250000	600000	Pruebas a la wiki desarrollada
\$3250000	Costo Total	

Fuente: autores

RECURSOS TECNOLÓGICOS

Los precios mostrados en la tabla dos representan los costos necesarios para suplir necesidades tecnológicas, herramientas necesarias para el desarrollo de la wiki semántica.

Tabla 2. Cotización por recursos tecnológicos.

Valor acumulado	Valor	Herramientas a usar
192000	192000	Computadores X 2
282000	90000	Internet
\$282000	Costo total	

Fuente: autores

ANÁLISIS COSTO-BENEFICIO

La mayor parte del costo invertido en la creación de un software se ve en el mantenimiento del mismo. Mantener un software conlleva un trabajo arduo de comprensión, en especial si las personas que lo necesitan no estuvieron involucradas en el proceso inicial de diseño y desarrollo. En estas circunstancias es claro que el tiempo invertido necesario para empezar un trabajo de mantenimiento puede ser bastante amplio, aún más si se trata de proyectos de gran envergadura. Por eso el uso de la wiki semántica representaría una inversión a largo plazo que reduciría el tiempo necesario para comenzar trabajos de mantenimiento, posiblemente en un factor de 2 a 3 (reduce el tiempo a la mitad hasta un tercio del mismo). Lo anterior representaría una reducción proporcional de los costos necesarios posteriores a la salida del software.

Anexo B. Cronograma de trabajo

Cronograma de trabajo por actividades

Tiempo a invertir (horas)	Actividad	Semana
12	Diseño de la ontología	01.12.14 - 07.12.14
8	Aprendizaje de las herramientas tecnológicas	
12	Aprendizaje de las herramientas tecnológicas	08.12.14 - 14.12.14
8	Implementación de la ontología en Protégé	
8	Pruebas a la ontología desarrollada	15.12.14 - 21.12.14
12	Construir la wiki semántica	
20	Construir la wiki semántica	22.12.14 - 28.12.14
8	Construir la wiki semántica	29.12.14 - 04.01.15
8	Pruebas a la wiki desarrollada	
4	Diseño del caso de estudio	
6	Diseño del caso de estudio	05.01.15 - 11.01.15
14	Implementación del caso de estudio	
8	Implementación del caso de estudio	12.01.15 - 18.01.15
8	Pruebas a la wiki semántica adaptada	

Fuente: autores

El cronograma de trabajo mostrado en la tabla cuatro se basa en las actividades identificadas y expuestas en la tabla 1, sobre el cual se estima lo siguiente:

- Un total de 136 horas para trabajar.
- Un promedio de 9 horas laborales por semana distribuidas en 4 horas diarias.