Universidad de las Ciencias Informáticas
Facultad1

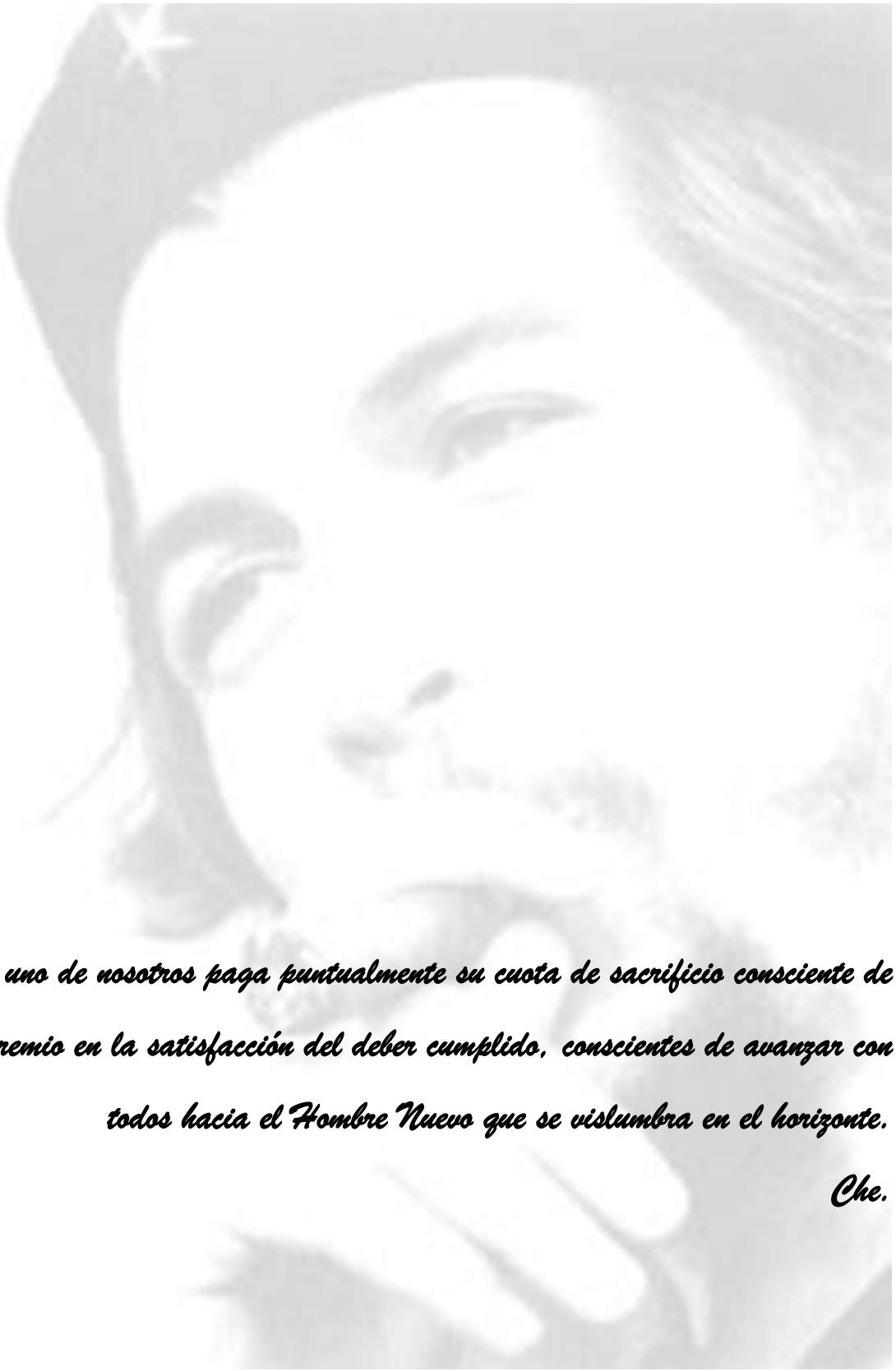
Título: Solución web para la representación de datos y edición de contextos basados en mapas conceptuales

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

Autores:
Yisel Correa Rodríguez
Yoslandy López Cristóbal

Tutor:
Msc. Jorge Landrian García

La Habana, Cuba
Junio 2012



Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.

Che.

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmamos la presente a los ____ días del mes de _____ del año_____.

_____ _____

Yisel Correa Rodríguez. Yoslandy López Cristóbal.
Autor Autor

_____

Msc. Jorge Landrian García.
Tutor

Datos de contacto

Autores:

Yisel Correa Rodríguez

Correo: ycorrear@estudiantes.uci.cu

Universidad de las Ciencias Informáticas. La Habana, Cuba

Yoslandy López Cristóbal

Correo: ycristobal@estudiantes.uci.cu

Universidad de las Ciencias Informáticas. La Habana, Cuba

Tutor:

Jorge Landrian García

Correo: jlandrian@uci.cu

Universidad de las Ciencias Informáticas. La Habana, Cuba

Dedicatoria

Yisel:

A mis padres especialmente a mi papito que aunque la vida no le dio la oportunidad de compartir conmigo este momento estoy segura de que se hubiera sentido muy orgulloso.

Yoslandy:

A mi madre, por estar siempre presente cuando la necesito y a quien le debo todo lo que he logrado en esta vida. A mi abuela, mi padre, mis tíos, en fin a toda mi familia.

Agradecimientos

Yisel:

A todas las personas que me ayudaron durante mis cinco años de estudio en la universidad.

A las amigas que he obtenido en la escuela y de las cuales me llevo muy lindos recuerdos.

A mi compañero de tesis por su arduo trabajo durante todo el año.

A toda mi familia por apoyarme y estar ahí siempre que los necesité, a mis abuelos, tías y tíos y a todas mis primas y primos.

A mi novio por estar conmigo y haberme ayudado en el momento más difícil de mi vida.

A mi mamita y mi hermana que cuidan de mí constantemente y a mi papito que lo hizo durante todo el tiempo de su vida.

Yoslandy:

A mi madre y mi abuela por su amor y apoyo incondicional.

A mi padre, mis tíos y tías que de una u otra forma me han ayudado a convertirme en quien soy.

A mi compañera de tesis por haberme aguantado durante todo el año y por su laborioso y arduo trabajo para que la tesis lograra salir adelante.

A los amigos con los que he compartido estos 5 años de estudios en la universidad.

A nuestro tutor por confiar en nosotros.

A Fidel y a la Revolución por este gran logro social que es la UCI. En fin a todos gracias.

Resumen

El presente trabajo está dirigido al acceso y la navegación por bases de datos a partir de la utilización de los mapas conceptuales como forma de estructurar la información, logrando que la representación de los datos brindada al usuario sea lo más distintiva posible. Se analizaron los conceptos básicos que tienen relación con el diseño teórico y se realizó un estudio del estado del arte de los sistemas existentes para graficar mapas conceptuales lo que permitió definir una interfaz visual para el editor de contextos en la que se pudieran representar los conceptos y relaciones de forma sencilla. A partir del estudio de la solución de referencia "Navegador Conceptual", se logró comprender sus funcionalidades y se describieron las herramientas, tecnologías y los lenguajes a utilizar en la realización de la solución, explicando sus principales características y ventajas. La solución de referencia, "Navegador Conceptual", permite el acceso a un gran volumen de información persistida en bases de datos y agiliza el proceso de obtención de información. Se logró una versión estable de la aplicación web para la representación de datos basada en mapas conceptuales y un editor de contextos gráfico para la gestión de los mismos, la facilidad en cuanto al acceso y manipulación de la información contenida en bases de datos y la integración de los procesos de edición de contextos y navegación en un solo sistema, todo lo cual mejora la usabilidad del "Navegador Conceptual".

Palabras clave:

Contextos, Descripción, Editor de Contextos, Mapas Conceptuales, Navegador Conceptual, Ontología, Sintaxis.

Índice

Introducción 1
Capítulo 1: Fundamentación teórica 6
Introducción 6
1.1 Conceptos asociados al dominio del problema 6
1.2 Solución de referencia 9
1.3 Soluciones existentes para la creación de mapas conceptuales 11
1.4 Lenguajes de implementación 13
1.5 Librerías 17
1.6 Herramientas 18
1.7 Tecnología 19
1.8 Modelado 19
1.9 Metodologías de desarrollo de *software* 20
Conclusiones 25
Capítulo 2: Características del sistema 26
Introducción 26
2.1 Descripción del sistema 26
2.2 Modelo de dominio 28
2.3 Captura de requisitos 30
2.3.1 Técnicas de captura de requisitos 30
2.3.2 Requerimientos funcionales del sistema 31
2.3.3 Requerimientos no funcionales del sistema 31
2.4 Actores del sistema 33
2.5 Historias de usuario 34
2.6 Plan de entregas 35
2.7 Tareas de ingeniería 36
Conclusiones 38
Capítulo 3: Diseño, implementación y validación de la solución 39
Introducción 39

3.1 Tarjetas CRC....39
3.2 Diagrama de clases....39
3.3 Arquitectura....42
3.4 Patrones....44
3.5 Estándares de diseño....48
3.6 Estándares de codificación....50
3.7 Tratamiento de errores....52
3.8 Diagrama de despliegue....52
3.9 Validación....53
3.9.1 Estrategia de prueba....54
3.9.2 Pruebas de unidad....54
3.9.3 Pruebas de aceptación....59
3.10 Resultados de las pruebas unitarias....62
3.11 Beneficios de la aplicación....62
Conclusiones....63
Conclusiones....64
Recomendaciones....65
Bibliografía....66
Anexos....70
Anexo 1. Historias de usuario....70
Anexo 2. Interfaces....77
Anexo 3. Tareas de ingeniería....81
Anexo 4. Tarjetas CRC....91
Anexo 5. Diagramas de clases....94
Anexo 6. Diseño de casos de prueba....96
Anexo 7. Pruebas de caja blanca....96
Anexo 8. Pruebas de aceptación....97

Índice de tablas

Tabla 1. Conceptos del Modelo de Dominio 29
Tabla 2. Requisitos funcionales. 31
Tabla 3. Prioridad de los Requisitos funcionales 33
Tabla 4. Requisitos no funcionales 33
Tabla 5. Actores del sistema 33
Tabla 6. Historia de usuario: Gestionar concepto 35
Tabla 7. Plan de entregas 35
Tabla 8. Tarea de ingeniería: Adicionar concepto 36
Tabla 9. Tarea de ingeniería: Modificar concepto 37
Tabla 10. Tarea de ingeniería: Eliminar concepto 37
Tabla 11. Tarea de ingeniería: Buscar concepto 38
Tabla 12. Tarjeta CRC de la clase ConceptOntology 39
Tabla 13. Caso de prueba: HU04 55
Tabla 14. Descripción de variables 55
Tabla 15. Matriz de datos: SC Adicionar consulta 56
Tabla 16. Matriz de datos: SC Modificar consulta 57
Tabla 17.. Matriz de datos: SC Adicionar columna 57
Tabla 18. Registro de no conformidades 58
Tabla 19. Prueba de aceptación: HU2-P1 Adicionar concepto 60
Tabla 20. Prueba de aceptación: HU2-P2 Modificar concepto 60
Tabla 21. Prueba de aceptación: HU2-P3 Eliminar concepto 61
Tabla 22. Prueba de aceptación: HU2-P4 Buscar concepto 61
Tabla 23. Pruebas a HU por cada iteración 62
Tabla 24. Historia de usuario: Autenticar usuario 70
Tabla 25. Historia de usuario: Gestionar relación 71
Tabla 26. Historia de usuario: Gestionar consulta 72
Tabla 27. Historia de usuario: Gestionar operador 72
Tabla 28. Historia de usuario: Guardar proyecto 73
Tabla 29. Historia de usuario: Cargar proyecto 73

Tabla 30. Historia de usuario: Generar ficheros XML 74
Tabla 31. Historia de usuario: Establecer conexión 75
Tabla 32. Historia de usuario: Realizar consulta 75
Tabla 33. Historia de usuario: Realizar navegación 76
Tabla 34. Historia de usuario: Gestionar investigación 76
Tabla 35. Tarea de ingeniería: Gestionar ficheros XML 81
Tabla 36. Tarea de ingeniería: Crear componente visual para realizar una consulta 82
Tabla 37. Tarea de ingeniería: Guardar historial de consultas realizadas 82
Tabla 38. Tarea de ingeniería: Validar consulta 82
Tabla 39. Tarea de ingeniería: Realizar consulta 83
Tabla 40. Tarea de ingeniería: Realizar navegación 83
Tabla 41. Tarea de ingeniería: Adicionar relación 84
Tabla 42. Tarea de ingeniería: Modificar relación 84
Tabla 43. Tarea de ingeniería: Eliminar relación 85
Tabla 44. Tarea de ingeniería: Adicionar consulta a un concepto 85
Tabla 45. Tarea de ingeniería: Modificar consulta de un concepto 85
Tabla 46. Tarea de ingeniería: Eliminar consulta de un concepto 86
Tabla 47. Tarea de ingeniería: Adicionar columnas a una relación 86
Tabla 48. Tarea de ingeniería: Eliminar columnas de una relación 87
Tabla 49. Tarea de ingeniería: Adicionar operador 87
Tabla 50. Tarea de ingeniería: Modificar operador 88
Tabla 51. Tarea de ingeniería: Eliminar operador 88
Tabla 52. Tarea de ingeniería: Establecer conexión 88
Tabla 53. Tarea de ingeniería: Guardar proyecto 89
Tabla 54. Tarea de ingeniería: Cargar proyecto 89
Tabla 55. Tarea de ingeniería: Adicionar a investigación 90
Tabla 56. Tarea de ingeniería: Eliminar nodo de la investigación 90
Tabla 57. Tarea de ingeniería: Autenticar usuario 91
Tabla 58. Tarjeta CRC de la clase OntologyConcept 91
Tabla 59. Tarjeta CRC de la clase ConceptField 91

Tabla 60. Tarjeta CRC de la clase OntologyRelation 92
Tabla 61. Tarjeta CRC de la clase RDBConceptualDescription 92
Tabla 62. Tarjeta CRC de la clase DescriptionGestor 92
Tabla 63. Tarjeta CRC de la clase DescriptionConcept 92
Tabla 64. Tarjeta CRC de la clase DescriptionRelation 93
Tabla 65. Tarjeta CRC de la clase ConceptQuerySintaxis 93
Tabla 66. Tarjeta CRC de la clase OperationDefinition 93
Tabla 67. Prueba de aceptación: HU3-P1 Adicionar relación 97
Tabla 68. Prueba de aceptación: HU3-P2 Modificar relación 98
Tabla 69. Prueba de aceptación: HU3-P3 Eliminar relación 98
Tabla 70. Prueba de aceptación: HU3-P4 Buscar relación 99
Tabla 71. Prueba de aceptación: HU4-P1 Adicionar consulta 99
Tabla 72. Prueba de aceptación: HU4-P2 Modificar consulta 100
Tabla 73. Prueba de aceptación: HU4-P3 Eliminar consulta 101
Tabla 74. Prueba de aceptación: HU5-P1 Adicionar operador 101
Tabla 75. Prueba de aceptación: HU5-P2 Modificar operador 102
Tabla 76. Prueba de aceptación: HU5-P3 Eliminar operador 102
Tabla 77. Prueba de aceptación: HU9-P1 Establecer conexión 103

Índice de figuras

Figura 1. Componente visual para la construcción de consultas 9
Figura 2. Columnas a mostrar 10
Figura 3. Investigación 10
Figura 4. Opciones de navegación 11
Figura 5. Fases de la metodología XP 24
Figura 6. Modelo de dominio 30
Figura 7. Diagrama de clases del paquete Ontología 40
Figura 8. Diagrama de clases del paquete Descripción 41
Figura 9. Diagrama de clases del paquete Sintaxis 41
Figura 10. Estilo arquitectónico Cliente-Servidor 42
Figura 11. Diagrama de arquitectura 44
Figura 12. Patrón encapsulamiento 45
Figura 13. Patrón subclases 46
Figura 14. Patrón interfaz 48
Figura 15. Patrón singleton 48
Figura 16. Modelo de despliegue 53
Figura 17. Prueba unitaria al método AddConcept 58
Figura 18. Prueba unitaria al método UpdateConcept 58
Figura 19. Prueba unitaria al método AddFields 59
Figura 20. Resultado de la prueba unitaria 59
Figura 21. Gráfica de prueba a HU por cada iteración 62
Figura 22. Interfaz del Editor 77
Figura 23. Interfaz HU2: Modificar concepto 78
Figura 24. Interfaz HU2: Buscar concepto 78
Figura 25. Interfaz HU3: Modificar relación 79
Figura 26. Interfaz HU9: Establecer conexión 79
Figura 27. Interfaz del Navegador 80
Figura 28. Interfaz HU10: Realizar consulta 80
Figura 29. Interfaz HU11: Realizar navegación 81

Figura 30. Diagrama de clases del paquete Ontología extendido 94
Figura 31. Diagrama de clases del paquete Descripción extendido 95
Figura 32. Diagrama de clases del paquete Sintaxis extendido 96

Introducción

A poco más de una década de su surgimiento la Web se ha convertido en una herramienta de uso cotidiano en la sociedad actual, jugando un papel importante al igual que otros medios como la televisión, la radio, la prensa, la telefonía, etc. La web es actualmente un medio extraordinariamente económico y flexible que permite el desarrollo del comercio, las comunicaciones, los negocios y el acceso a la información y los servicios desde cualquier parte del mundo.

El alto incremento que ha poseído en estos últimos años, en cuanto al número de usuarios que la utilizan y las funcionalidades que brinda, ha traído como consecuencia la necesidad de incorporar nuevas tecnologías que mejoren su potencialidad y usabilidad, las cuales han experimentado una rápida evolución. Desde las primeras: HTML[1] y HTTP[2], hasta nuestros días, el número de estas tecnologías se ha visto incrementado con el surgimiento de otras como son: Java, JavaScript, ASP[3], PHP[4], Flash, XML[5], y SVG[6], las cuales conllevan a una web mejor, más amplia, potente, flexible y fácil de mantener.

Algunas de las tendencias evolutivas más marcadas en estos últimos años referentes a la web son:

- ✓ La generación dinámica de páginas.
- ✓ La adaptación al usuario.
- ✓ La mayor interactividad con el usuario.
- ✓ La concepción de la web como plataforma universal para el despliegue de aplicaciones.

Otra de las tendencias evolutivas asociadas a la web es el acoplamiento con las bases de datos, definidas como "*conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso"* (basesdedatos.org, 2010). Estas juegan un papel muy importante en el ámbito de preservar la información.

Con el avance de las Tecnologías de la Información y las Comunicaciones, diariamente se genera un gran cúmulo de información persistente que es almacenada en las bases de datos y que para acceder a las mismas requieren de personal con amplios conocimientos en SQL[7]. Por esta razón fue necesaria la implementación de soluciones que permitieran la navegación al usuario de una manera rápida y cómoda.

[1] *Hypertext Markup Language*
[2] *Hypertext Transfer Protocol*
[3] *Active Server Pages*
[4] *Hypertext Pre-processor*
[5] *Extensible Markup Language* o Lenguaje Extensible de Marcas
[6] *Scalable Vector Graphics*
[7] *Structured Query Language*

Una forma fácil de visualizar la información la constituyen los mapas conceptuales, los cuales fueron desarrollados para representar gráficamente el conocimiento. Estos constituyen una red de conceptos de un dominio específico que se relacionan entre sí para resumir áreas de estudio y organizar la información.

Según Joseph D. Novak[8] en su artículo *"The Theory Underlying Concept Maps and How to Construct Them"* un mapa conceptual (MC) es *"una herramienta para organizar y representar el conocimiento"* (Novak, 2006).

Actualmente un gran número de aplicaciones hacen uso de esta técnica para el manejo de su información, sobre todo a la hora de interactuar con el usuario, logrando que la comunicación sea más amena e interactiva.

Cuba no está exenta del uso de esta tecnología. En la Universidad de las Ciencias Informáticas existe la herramienta "Navegador Conceptual", que fue creada a partir de la necesidad de acceder y navegar por toda la información persistida en el SAIME (Servicio Administrativo de Identificación, Migración y Extranjería), el cual constituye un conjunto de soluciones de *software* desarrolladas con el objetivo de brindar soporte informático a los procesos de Identificación, Migración y Extranjería de la República Bolivariana de Venezuela.

El "Navegador Conceptual" permite navegar por la información almacenada en una base de datos, a través del uso de los mapas conceptuales como forma de manejar los conceptos y relaciones que en la misma se asocian.

Sin embargo, a pesar de las ventajas que esta herramienta brinda en cuanto al acceso y manipulación de la información contenida en bases de datos, tiene una usabilidad limitada pues es una aplicación *desktop*, la cual requiere de instalación y actualización personalizada y su disponibilidad se limita al ordenador donde está instalada, lo que representa una gran desventaja ante las aplicaciones web, las cuales están disponibles desde cualquier lugar y son multiplataforma, por lo que funcionan independientemente del sistema operativo que se use, siempre están actualizadas y tienen un menor requerimiento de hardware pues sólo se necesita del uso de un navegador. Otra de las desventajas del "Navegador Conceptual" es que no brinda la posibilidad de crear o modificar contextos, entiéndase como contextos los ficheros XML de ontología, sintaxis y descripción, que son utilizados por la aplicación para navegar y obtener los datos

[8] Donald Joseph Novak (n. 1932), profesor emérito en la Universidad de Cornell y científico de investigación en el Institute for Human and Machine Cognition (IHMC). Es conocido por su desarrollo de mapas conceptuales en la década de 1970.

solicitados por el usuario. Como consecuencia, la edición de los contextos es realizada manualmente resultando ser una tarea muy engorrosa.

A partir de esta **situación problemática** se plantea el siguiente **problema científico**: ¿Cómo mejorar la representación de datos para aumentar la usabilidad del "Navegador Conceptual"?

El presente trabajo centra su **objeto de estudio** en la navegación en bases de datos a partir de mapas conceptuales y el **campo de acción** se enfoca en la navegación en bases de datos a partir de mapas conceptuales mediante el uso del "Navegador Conceptual".

Para dar solución al problema anterior se plantea como **objetivo general** desarrollar una solución web para la representación de datos y edición de contextos de forma gráfica que permita mejorar la usabilidad del "Navegador Conceptual", tomando como base los siguientes **objetivos específicos**:

- ✓ Realizar el diseño de la solución web.
- ✓ Desarrollar una interfaz web al componente "ConceptualMap.dll" para mejorar la usabilidad del "Navegador Conceptual".
- ✓ Desarrollar un editor de contextos para la gestión de ficheros XML.
- ✓ Validar la solución web desarrollada.

Para dar cumplimiento a los objetivos trazados se plantean las siguientes **tareas**:

- ✓ Identificación de los problemas de usabilidad existentes en la aplicación *desktop*.
- ✓ Estudio de los sistemas similares.
- ✓ Análisis bibliográfico para la definición de las tecnologías, herramientas y metodología a utilizar.
- ✓ Identificación de los requisitos de la solución web para la navegación conceptual.
- ✓ Identificación de las historias de usuario de la solución web para la navegación conceptual.
- ✓ Descripción de las historias de usuario de la solución web para la navegación conceptual.
- ✓ Realización del diseño de la solución web.
- ✓ Implementación de los componentes gráficos del editor de contextos.
- ✓ Implementación del diseño lógico del editor de contextos.
- ✓ Desarrollo de componentes gráficos para la navegación basada en mapas conceptuales.
- ✓ Realización de las pruebas unitarias del editor de contextos.
- ✓ Realización de las pruebas unitarias de la solución web.

Los **métodos científicos** utilizados en la investigación fueron:

- ✓ **Histórico-Lógico:** En la presente investigación se utiliza este método para realizar un análisis de la solución existente así como del uso de los mapas conceptuales para la representación gráfica de la información.
- ✓ **Analítico-Sintético:** Este método incluye el análisis de la bibliografía disponible para realizar un estudio del estado del problema a resolver y definir los conceptos esenciales. También se especifican las principales características de las herramientas para el desarrollo de la solución y las ventajas del uso de las mismas.
- ✓ **Modelación:** El uso de este método permite la realización de los modelos de la ingeniería de *software* que dan cumplimiento a los requisitos funcionales y no funcionales.
- ✓ **Entrevista:** Se utiliza para recopilar información cualitativa de la investigación, mediante la cual se diagnostica el estado actual del "Navegador Conceptual" y su funcionamiento. Esta información es obtenida a través de personas capacitadas en temas referidos al acceso a la información existente en las bases de datos usando mapas conceptuales.

Con la realización de esta investigación se espera obtener los siguientes **resultados**:

Desde el punto de vista práctico:

- ✓ Versión estable de la aplicación web para la representación de datos basada en mapas conceptuales y un editor de contextos gráfico para la gestión de los mismos.
- ✓ Documentación asociada a la solución desarrollada.

Desde el punto de vista social:

- ✓ Facilidad en cuanto al acceso y manipulación de la información contenida en bases de datos.
- ✓ Integración de los procesos de edición de contextos y navegación en un solo sistema.

Justificación de la investigación:

El presente trabajo centra su investigación en el acceso y la navegación por bases de datos a partir de la utilización de los mapas conceptuales como forma de estructurar la información, permitiendo que la representación de los datos brindada al usuario sea lo más distintiva posible.

La solución de referencia, "Navegador Conceptual", permite el acceso a un gran volumen de información persistida en bases de datos sin la necesidad de requerir de personas con amplios conocimientos de SQL y del negocio.

La solución a desarrollar, además de facilitar la búsqueda de reportes específicos y agilizar el proceso de obtención de información, permitirá el acceso a la misma desde cualquier ordenador sin la necesidad de

haber instalado el programa previamente. También posibilitará la edición de contextos de forma gráfica, agilizando el proceso de creación de los ficheros XML de ontología, sintaxis y descripción. Dicha solución logrará la integración de los procesos de edición de contextos y navegación en un único sistema.

El presente documento consta de tres **capítulos:**

Capítulo 1: Fundamentación teórica.

En este capítulo se sistematiza el basamento teórico de los temas tratados en la investigación. En el mismo se hace un estudio del estado del arte a partir del cual se describen los principales conceptos, tecnologías y herramientas asociados al dominio del problema.

Capítulo 2: Características de la solución.

En este capítulo se ofrece una vista general de la propuesta de la solución. Se exponen los requisitos funcionales y no funcionales que esta requiere y se abordan las principales características de la misma.

Capítulo 3: Diseño, implementación y validación de la solución.

En este capítulo se realiza el diseño de los componentes a implementar y se exponen los artefactos obtenidos como resultado de la implementación. Se describen las pruebas que se le realizaron al sistema.

Capítulo 1: Fundamentación teórica

Introducción

Para realizar una investigación se requiere de una revisión previa del tema en cuestión, un estudio del estado del arte, así como una caracterización de las tecnologías y herramientas que darán respuesta a la solución propuesta.

En el presente capítulo se abordan los conceptos relacionados con la navegación en bases de datos a partir de mapas conceptuales mediante el uso del "Navegador Conceptual". También se realiza un estudio de los sistemas actuales que persiguen propósitos semejantes a la solución que se propone y se expone la selección de las herramientas de desarrollo y modelado utilizadas para desarrollar la aplicación.

1.1 Conceptos asociados al dominio del problema

Mapas conceptuales

Los mapas conceptuales fueron desarrollados por el Profesor Joseph D. Novak como producto de investigaciones realizadas en los años 1960, basándose en la teoría de David Ausubel[9] del aprendizaje significativo. Son instrumentos sencillos y prácticos para la representación del conocimiento que permiten transmitir con claridad mensajes conceptuales complejos y facilitar tanto el aprendizaje como la enseñanza (Dürsteler, 2004).

Su objetivo es representar relaciones entre conceptos en forma de proposiciones. Los conceptos se incluyen en círculos o cajas, mientras que las relaciones entre estos se explican mediante líneas que unen su cajas respectivas. Las líneas, a su vez, tienen palabras asociadas que describen cuál es la naturaleza de la relación que liga los conceptos (Dürsteler, 2004).

Los elementos que conforman los mapas conceptuales son (Ojeda Cabrera, y otros, 2007):

- ✓ **Conceptos:** constituyen los nodos del MC. Pueden ser elementos concretos o abstractos.
- ✓ **Palabras de enlace:** Se escriben en la línea que une a dos nodos, pueden ser unidireccionales, bidireccionales o simplemente asociativas. Son las palabras o frases que sirven para unir los conceptos y expresar el tipo de relación existente entre ellos.
- ✓ **Proposiciones:** Constituyen dos o más conceptos unidos por palabras de enlace para formar una unidad semántica más simple que tiene valor real.

[9] David Paul Ausubel (Nueva York, 1918-2008), psicólogo y pedagogo estadounidense, una de las personalidades más importantes del constructivismo.

Se puede deducir que los mapas conceptuales no son más que gráficos utilizados para representar y organizar el conocimiento de manera que permitan a las personas que los realizan estructurar y comprender mejor la información que ya poseen. Para aquellos que desconozcan de un tema determinado, un MC representa una vía fácil de apropiarse del mismo.

Ontología

El concepto informático de ontología ha venido desarrollándose desde hace aproximadamente tres décadas por una comunidad pequeña de investigadores (Sanz, et al., 2007).

Estas incluyen definiciones de conceptos básicos de un dominio y las relaciones entre ellos, que son útiles para los ordenadores. Codifican el conocimiento de un dominio y también el conocimiento que extiende los dominios. En este sentido, hacen el conocimiento reutilizable (López, 2007).

Debe ser especificada usando un lenguaje formal que pueda ser procesado por ordenadores y no sólo por personas (Sanz, y otros, 2007).

La ontología se ha convertido en una importante herramienta para muchas aplicaciones, lo que ha motivado que en la práctica se hayan producido un gran número de metodologías y herramientas para la creación de sistemas basados en ontologías.

Algunos de los campos en los que se realizan proyectos que aplican las ontologías son (Sanz, et al., 2007):

- ✓ **Comercio electrónico:** Se utilizan en este campo para hacer una descripción tanto del dominio como de los usuarios y sus tendencias para facilitar la agrupación de aquellos con comportamientos similares. Esto permite realizar estudios de mercado de forma más sencilla para que la información que se les muestre a los usuarios sea lo más personalizada posible.
- ✓ **Búsqueda en Internet:** Para guiar la búsqueda en un dominio determinado, incrementando así la precisión y recuperación de los documentos buscados.
- ✓ **Biomedicina:** Este campo constituye uno de los de mayor interés pues presenta una gran cantidad de recursos que tienen que ser procesados e interpretados. Las ontologías ayudan a integrar esta información para hacer que el acceso a la misma se haga de una manera más cómoda.
- ✓ **Bibliotecas digitales:** Las bibliotecas digitales presentan una gran cantidad de información en formato digital la cual puede estar distribuida en diversos sistemas informáticos. Las técnicas basadas en ontologías posibilitan un acceso sencillo a dichos recursos digitales.

En el “Navegador Conceptual” se hace uso del término “Ontología”. Este se refiere al documento que describe los conceptos y las relaciones de un dominio específico y es persistido en un fichero XML, es decir, que no cumple rigurosamente con el término informático sino que toma del mismo solamente algunos aspectos.

Sintaxis

La palabra sintaxis deriva del latín “*syntaxis”*, que a su vez tiene origen en un término griego que significa *“coordinar”*. Se trata de la parte de la gramática que enseña a coordinar y unir las palabras para formar las oraciones y expresar conceptos. En la informática, la sintaxis es el conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje de programación (Real Academia Española, 2010).
En el “Navegador Conceptual” el documento sintaxis es un XML en el que se definen todas las funciones y operadores y la precedencia de los mismos. La sintaxis es utilizada por el *parser* para generar los Tokens correspondientes según la gramática del lenguaje y la sintaxis expresada en el documento (García, 2008).
Es decir, que el objeto sintaxis define las operaciones que pueden ser específicas de la problemática a la cual se le esté dando solución y es independiente de la implementación final que se esté utilizando.

Descripción

Cada concepto de la ontología definido en el “Navegador Conceptual” debe poseer un documento de descripción conceptual de la base de datos en el que por cada concepto se coloca el nombre que es único entre todos los conceptos de la ontología y la consulta SQL a la que corresponde en la base de datos. Si un concepto corresponde a una sola tabla en la base de datos y todos los campos tienen el mismo nombre que la columna correspondiente en la tabla, no es necesario poner una consulta SQL, se puede colocar solamente el nombre de la tabla. Las relaciones entre los conceptos cuando son descritas en el documento de descripción pueden ser de dos tipos; las relaciones simples que son aquellas en las que uno de los dos conceptos posee un identificador del otro, una relación 1-N en la base de datos; y las relaciones con una tabla intermedia, las relaciones M-N. Estas relaciones son RDBConceptRelation y RDBInnerTableConceptRelation, la segunda es una extensión de la primera (García, 2008).

Ficheros XML

XML es un lenguaje usado para estructurar información en un documento o en cualquier fichero que contenga texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos (Villate, 2001).

Comparado con otros sistemas usados para crear documentos, el XML tiene la ventaja de poder ser más exigente en cuanto a la organización del mismo, dando como resultado documentos mejor estructurados que permiten la lectura de datos a través de diferentes aplicaciones.

1.2 Solución de referencia

Navegador conceptual

El "Navegador Conceptual" constituye un mecanismo de acceso a la información que está orientado a reducir la complejidad introducida por la gran cantidad de variantes de acceso a datos existentes, brindando además una organización conceptual de la información que facilita el manejo de la misma tanto por las personas como por los sistemas informáticos.

A partir de la librería ConceptualMap.dll que encapsula toda la lógica necesaria para acceder a la información, el objetivo de la herramienta "Navegador Conceptual" es presentar esta información de forma amena al usuario, explotando todas las potencialidades del lenguaje de acceso creado y permitiendo la navegabilidad a través de las relaciones de los conceptos (García, 2008).

Su utilización permite acceder a toda la información almacenada de una forma lógica, intuitiva y cómoda, posibilitando una visión global y al mismo tiempo detallada de todos los datos recopilados (García, 2008).

Entre las funcionalidades que esta posee se encuentran (García, 2008):

- ✓ Permite al usuario realizar consultas en un lenguaje sencillo mediante un componente visual, el cual se nutre de los elementos existentes en el fichero XML Ontología y de los operadores y funciones del fichero Sintaxis del contexto seleccionado. Esta forma de realizar las consultas permiten a los usuarios poder acceder a la información almacenada en las bases de datos sin necesidad de tener altos conocimientos del negocio.

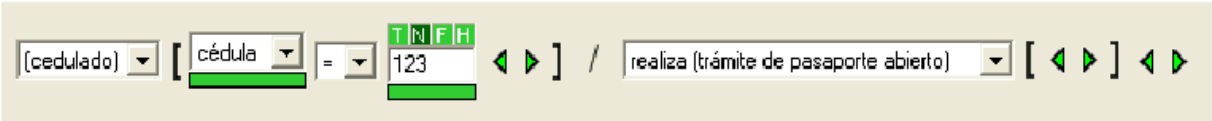


Figura 1. Componente visual para la construcción de consultas

- ✓ La solución brinda la posibilidad de poder definir cuáles campos se desean mostrar como resultado de la consulta creada. Se pueden adicionar elementos complejos al resultado de la consulta, tales como: campos de conceptos, expresiones, funciones, consultas de concepto y valores (literal, numérico, fecha y hexadecimal). Por defecto el sistema sólo muestra los campos pertenecientes al último concepto construido en la expresión.

Figura 2. Columnas a mostrar

- ✓ La herramienta posibilita ir añadiendo elementos a la investigación a medida que se va navegando por las relaciones de los conceptos implicados en el resultado de una consulta generada.

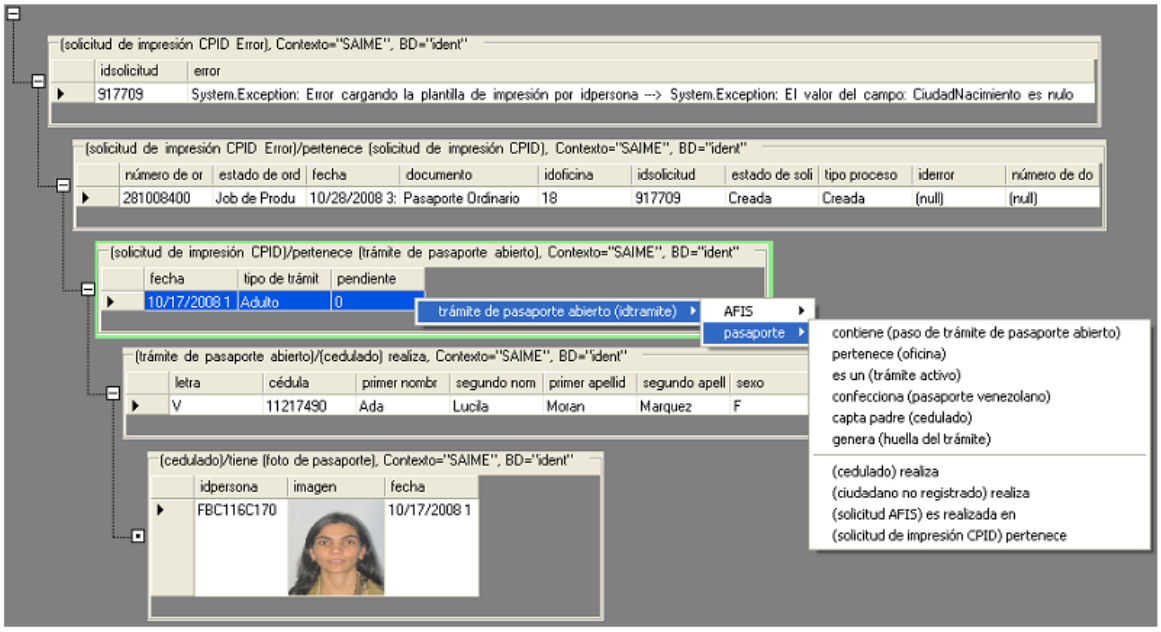


Figura 3. Investigación

- ✓ Se puede indicar con cuáles contextos se desea navegar y a cuál base de datos se desea acceder. Además la herramienta permite definir nuevos contextos, brindado la posibilidad al usuario de

seleccionar los contextos ontología, sintaxis y descripción que serán usados posteriormente para la navegación.

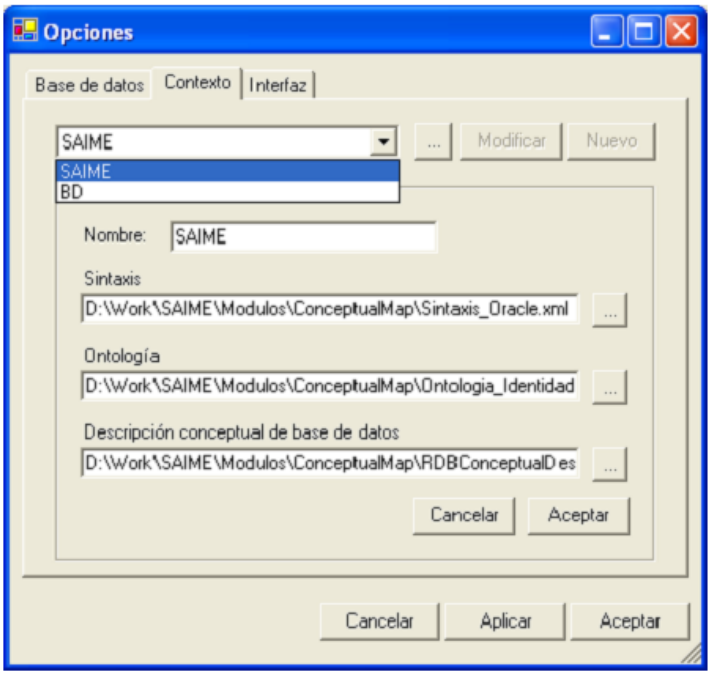


Figura 4. Opciones de navegación

Una de las grandes potencialidades que brinda es el permitir organizar y relacionar conceptualmente la información de un sistema independientemente de donde esté persistida, desacoplando de esta forma los sistemas del mecanismo final de persistencia (García, 2008).

La herramienta “Navegador Conceptual” a pesar de las ventajas que brinda, no permite la elaboración de contextos de forma gráfica. Por tal motivo se hace necesario el estudio de soluciones que posibiliten la representación de la información de forma sencilla para así poder definir una interfaz para el Editor de Contextos.

1.3 Soluciones existentes para la creación de mapas conceptuales

CMapTools

CMapTools es una herramienta desarrollada por el IHMC[10] de la Universidad de *West* Florida, Estados Unidos, con la finalidad de apoyar la construcción de modelos de conocimiento representados en forma de mapas conceptuales (Ávila, 2006).

Entre sus principales ventajas está la posibilidad de crear mapas conceptuales web que incorporen archivos adjuntos o enlaces a otras páginas, por tanto, los esquemas creados con esta herramienta permiten interactuar con su contenido a través de un navegador. En segundo lugar, la posibilidad de compartir los mapas conceptuales y sus recursos a través de Internet. Existen servidores públicos donde se pueden alojar los esquemas y los archivos adjuntos a éstos. Posee un entorno de trabajo sencillo, claro e intuitivo; ventana de estilos que facilita el trabajo; posibilidad de ilustrar los conceptos con símbolos, imágenes, colores, formas, sombras, fuentes y estilos; facilidades para relacionar conceptos de forma sencilla, relaciones que se explican con un texto en los enlaces.

Esta herramienta brinda la posibilidad de exportar los gráficos elaborados en forma de imagen (jpg, gif, png, bmp), página web, texto o formato XML.

DigiDocMap

DigiDocMap es una aplicación para la creación de mapas conceptuales según los requerimientos de la propuesta de Novak. Los usuarios usan el editor entrando la información sobre conceptos y frases de enlace en formularios en una típica interfaz de usuario creada en una página web. El editor genera una página HTML que contiene el mapa en forma gráfica (Rovira, 2005).

DigiDocMap forma parte de un conjunto de herramientas desarrollados por y para el Máster *Online* en Documentación Digital que se ofrecen como herramientas *freeware* de utilidad para documentalistas, bibliotecarios, archiveros, creadores de sitios web y profesionales innovadores de la cultura digital en general. Cuenta con algunas prestaciones adicionales de interactividad, como replegar y desplegar las ramas del árbol de conceptos, ocultar y mostrar zonas del mapa o cambiar los atributos tipográficos, como el tamaño o tipo de letra (Rovira, 2005).

GECOSOFT

La plataforma GECOSOFT es el resultado de la integración de MACOSOFT[11] y SERVIMAP[12]. MACOSOFT permite la inserción y eliminación de los conceptos y relaciones que conforman los mapas

[10] *Institute for Human and Machine Cognition*
[11] *Software* para elaborar Mapas Conceptuales
[12] Servidor de Mapas Conceptuales

conceptuales de forma fácil y cómoda, acompañado de una interfaz muy comunicativa en cuanto a las funcionalidades disponibles. Brinda la posibilidad de poder diferenciar los conceptos a través de una galería de imágenes agrupadas en diferentes categorías. Se pueden incorporar recursos a los conceptos, ya sean documentos, videos, páginas web, entre otros elementos. Los MC elaborados en esta herramienta pueden ser salvados como imágenes o páginas web. SERVIMAP es un sistema que brinda todo un conjunto de funcionalidades para el almacenamiento, clasificación y publicación de los MC y conjuntamente con MACOSOFT, permite el trabajo sobre los MC de forma colaborativa. Con la integración de estas herramientas se logra crear un ambiente distribuido en el que se implementan los mecanismos de elaboración, almacenamiento, organización/clasificación, búsqueda de conocimiento en forma de MC y de recursos, así como el trabajo de forma colaborativa y a distancia. (Simón, et al., 2006)

Aportes de las soluciones estudiadas

Con el estudio de estos sistemas para la creación de mapas conceptuales, se demostró que no existe una herramienta capaz de genera ficheros XML que puedan ser interpretados por el “Navegador Conceptual”, sin embargo se logró definir una interfaz visual para el “Editor de Contextos” de forma que permitiera representar los conceptos y relaciones gráficamente a través de mapas conceptuales para facilitar su comprensión. Los conceptos serán representados en forma de nodos y las relaciones poseerán un texto en los enlaces, el que contendrá la descripción de la misma. El “Editor de Contextos” será capaz de interpretar la información gráfica representada en los mapas conceptuales para generar los archivos XML de ontología, sintaxis y descripción requeridos por el “Navegador Conceptual”.

1.4 Lenguajes de implementación

CSharp

C# es un lenguaje orientado a objetos elegante y con seguridad de tipos que posibilita a los desarrolladores crear una amplia gama de aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Permite crear aplicaciones cliente para Windows tradicionales, servicios web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos y muchas tareas más (Microsoft, 2005).

Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. Todas las variables y métodos, incluido el método *Main* que es el punto de entrada de la aplicación, se encapsulan dentro de definiciones de clase. Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces. Los métodos que reemplazan a los métodos

virtuales en una clase primaria requieren la palabra clave *override* como medio para evitar redefiniciones accidentales. En C#, una estructura es como una clase sencilla; es un tipo asignado en la pila que puede implementar interfaces pero que no admite la herencia (Microsoft, 2005).

El estándar ECMA-334[13] lista las siguientes metas en el diseño para C# (Microsoft, 2006):

- ✓ Lenguaje de programación orientado a objetos simple, moderno y de propósito general.
- ✓ Inclusión de principios de ingeniería de *software* tales como revisión estricta de los tipos de datos, revisión de límites de vectores, detección de intentos de usar variables no inicializadas y recolección de basura automática.
- ✓ Capacidad para desarrollar componentes de *software* que se puedan usar en ambientes distribuidos.
- ✓ Portabilidad del código fuente.
- ✓ Fácil migración del programador al nuevo lenguaje, especialmente para programadores familiarizados con C, C++ y Java.
- ✓ Soporte para internacionalización.
- ✓ Adecuación para escribir aplicaciones de cualquier tamaño: desde las más grandes y sofisticadas como sistemas operativos hasta las más pequeñas funciones.
- ✓ Aplicaciones económicas en cuanto a memoria y procesado.

JavaScript

En los años de 1990, Netscape creó "*Livescript"*. Las primeras versiones de este lenguaje fueron principalmente dedicadas a pequeños grupos de diseñadores web que no necesitaban utilizar un compilador o sin ninguna experiencia en la programación orientada a objetos (Pérez, 2010).

JavaScript es el lenguaje de programación más utilizado en Internet para añadir interactividad a las páginas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario (Pérez, 2010). Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Pérez, 2010).

[13] Norma internacional que especifica la forma y establece la interpretación de los programas escritos en el lenguaje de programación C #.

JavaScript se integra directamente en páginas HTML. A continuación se muestran sus principales características:

- ✓ Es interpretado por el cliente, lo que significa que no es compilado.
- ✓ No es, como Java, un lenguaje de programación orientada a objetos (OOP por sus siglas en inglés). JavaScript no emplea clases ni herencia típicas de la OOP.
- ✓ Su código se integra en las páginas HTML.
- ✓ No es necesario declarar los tipos de variables que van a utilizarse.
- ✓ Las referencias a objetos se comprueban en tiempo de ejecución, por lo tanto no se compila.

La ventaja que presenta JavaScript sobre el HTML es que permite crear páginas más dinámicas, lo que las hace más atractivas para el usuario. Para utilizar y dominar el JavaScript es prerrequisito indispensable saber HTML.

XML

XML es un conjunto de reglas que sirven para definir etiquetas semánticas para organizar un documento. Es una tecnología muy sencilla que tiene a su alrededor otras tecnologías que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores (Alvarez, 2001).

XML brinda la posibilidad al programador de dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos pues algunas tareas tediosas como la validación de los mismos o el recorrido de las estructuras corre a cargo del lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse por ello (Alvarez, 2001).

CSS

Las hojas de estilos aparecieron poco después que el lenguaje de etiquetas SGML[14], alrededor del año 1970. Desde la creación de SGML se observó la necesidad de definir un mecanismo que permitiera aplicar de forma consistente diferentes estilos a los documentos electrónicos. En 1995, la W3C decidió apostar por el desarrollo y estandarización de CSS[15] y lo añadió a su grupo de trabajo de HTML (Pérez, 2010).

[14] *Standard Generalized Markup Language* o Estándar de Lenguaje de Marcado Generalizado. Consiste en un sistema para la organización y etiquetado de documentos.
[15] Hojas de Estilo en Cascada

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML[16]. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas (Pérez, 2010). Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas pues obliga a crear documentos HTML/XHTML bien definidos y con significado completo, también llamados "*documentos semánticos*". Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y posibilita visualizar el mismo documento en infinidad de dispositivos diferentes (Pérez, 2010).

HTML

HTML es la "*lengua franca"* para publicar hipertexto en la web. Se trata de un formato no propietario basado en SGML y puede ser creado y procesado por una amplia gama de herramientas, desde simples editores de texto sin formato a sofisticadas herramientas de creación (W3C, 2012).

Tiene sus limitaciones, así que a menudo se utilizan otras herramientas como las hojas de estilo, que le dan mayor libertad al diseñador. En concreto, el HTML le da las indicaciones mencionadas al navegador para que presente el documento en la pantalla de la manera adecuada (maestrosdelweb, 2010).

El HTML se hizo popular por su sencillez, es fácil de aprender y eso lo hace accesible a mayor número de personas. Estos documentos web deben estar escritos con el mismo lenguaje para que diferentes ordenadores puedan leerlos, si alguien utiliza un sistema diferente no podrá compartir su información con los que usan el HTML ni podrá visualizar los de los demás, por eso la tendencia es crear un estándar que evoluciona poco a poco y que es compatible con la versión anterior (maestrosdelweb, 2010).

El físico nuclear Tim Berners Lee[17] definió la primera versión de HTML en el año 1989. Desarrolló su idea mientras trabajaba en el CERN (Centro Europeo para la Investigación Nuclear). Pretendía facilitar el acceso a todos los datos del Centro independientemente del ordenador en el que se encontrara esa información, tanto si estaba en el propio centro o en los ordenadores de las diferentes instituciones que colaboraban con el CERN. El producto de su idea sería una plataforma de tipo hipertexto y un protocolo de comunicaciones que se denominó HTTP (maestrosdelweb, 2010).

SVG

[16] *eXtensible HyperText Markup Language*

[17] Sir Timothy "Tim" John Berners-Lee, considerado el padre de la web. Él y su grupo crearon lo que por sus siglas en inglés se denomina Lenguaje HTML, el protocolo HTTP y el sistema de localización de objetos en la web URL (*Uniform Resource Locator*).

SVG[18] es un lenguaje para describir gráficos bidimensionales en XML. SVG permite tres tipos de objetos gráficos: formas gráficas vectoriales (por ejemplo, rutas que consisten en curvas y líneas rectas), imágenes y texto. Los objetos gráficos, incluido texto, se pueden agrupar, se les puede asignar un estilo, transformar y componer en objetos previamente representados. El conjunto de funciones SVG incluye transformaciones anidadas, trazados de recorte, máscaras alfa y objetos de plantilla (Microsoft, 2012).

Los dibujos SVG pueden ser interactivos y dinámicos. Por ejemplo, las animaciones se pueden definir y activar mediante el uso de scripts. Las aplicaciones de SVG sofisticadas son posibles mediante el uso de scripts que obtienen acceso a DOM y proporcionan acceso total a todos los elementos, atributos y propiedades. Un gran conjunto de controladores de eventos como "*onmouseover*" y "*onclick*" se puede asignar a cualquier objeto gráfico SVG (Microsoft, 2012).

Debido a la compatibilidad de SVG y el aprovechamiento de otros estándares web, las funciones como el *scripting* se pueden realizar en elementos SVG y XHTML de forma simultánea dentro de la misma página web (Microsoft, 2012).

1.5 Librerías

jQuery

jQuery es una librería de JavaScript rápida y concisa que simplifica el recorrido de documentos HTML, el manejo de eventos, animación, y las interacciones Ajax[19] para el desarrollo web rápido. jQuery está diseñado para cambiar la forma en que se escribe JavaScript (jQuery Project, 2010). Al igual que otras bibliotecas, ofrece una serie de funcionalidades que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. El uso de jQuery brinda algunas ventajas como las mencionadas a continuación (jQuery Project, 2010):

- ✓ Simplificar la manera de interactuar con los documentos HTML.
- ✓ Manipular el árbol DOM.
- ✓ Manejar eventos.
- ✓ Desarrollar animaciones.
- ✓ Agregar interacción con la técnica Ajax a páginas web.

jQuery UI

[18] *Scalable Vector Graphics*
[19] *Asynchronous Java Script And XML*

jQuery UI es una librería de componentes para el *framework* jQuery que provee un conjunto de complementos *(plugins)*, efectos visuales avanzados y de alto nivel y componentes reutilizables *(widgets)* (jQuery UI Team, 2012). Añade comportamiento a los nodos obtenidos o creados a partir de la invocación de métodos como el *$.draggable()* y permite personalizar las interfaces mediante la selección de temas existentes o la creación de los mismos según el gusto del usuario. El selector y personalizador de temas se encuentra publicado en una página en Internet.

Esta aporta elementos como (jQuery UI Team, 2012):

- ✓ Componentes de interacción: *drag & drop, sorting, resizing*, etc.
- ✓ Efectos: *show, hide, toggle, color animation, class manipulation*, etc.
- ✓ *Widgets: accordion, date picker, dialog, slider* y *tabs*.

1.6 Herramientas

Microsoft Visual Studio 2010

Microsoft Visual Studio es un potente Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) que asegura código de calidad durante todo el ciclo de vida de la aplicación, desde el diseño hasta la implementación (Microsoft, 2012).

Microsoft Visual Studio 2010 incluye mejoras visuales pues se ha rediseñado con el fin de mejorar la legibilidad. También se han quitado las líneas y los degradados innecesarios para conseguir una mayor claridad. Contiene herramientas para ayudar a explorar el código con rapidez como el Editor de código, la Búsqueda mientras se escribe y la Jerarquía de llamadas (Microsoft, 2012).

Visual Studio 2010 incorpora nuevas características mejoradas que hacen que todo el proceso de desarrollo sea más sencillo, desde el diseño a la implementación. Permite personalizar el área de trabajo mediante la compatibilidad con varios monitores y crea aplicaciones enriquecidas para *SharePoint* y la Web (Microsoft, 2011).

Altova XMLSpy

XMLSpy es un avanzado editor de XML para modelar, editar, transformar y depurar tecnologías relacionadas con XML. El editor de XML proporciona la potencia necesaria para crear las aplicaciones web y XML más avanzadas. Al mismo tiempo es lo suficientemente flexible como para permitir trabajar con cualquier tecnología XML de manera que se adapte a la complejidad del documento (Altova, 2012).

XMLSpy abstrae de la dificultad de trabajar con tecnologías XML por medio de una interfaz de usuario intuitiva y una gran variedad de vistas de edición de XML (Altova, 2012).

El editor XML incluye, entre otras, las siguientes características y funcionalidades (Altova, 2012):

- ✓ Editor XML inteligente.
- ✓ Validación de XML con gestión avanzada de errores.
- ✓ Editor visual de esquemas XML.
- ✓ Integración con bases de datos.
- ✓ Soporta *Sharepoint Server*.
- ✓ Generación de código Java / C# / C++ a partir de esquemas XML.
- ✓ Editor JSON[20] y conversión entre JSON y XML.
- ✓ Soporta ficheros grandes.
- ✓ Versiones de 32 y 64 bits.

1.7 Tecnología

Ajax

AJAX es una técnica de desarrollo web para crear aplicaciones rápidas y dinámicas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible actualizar partes de una página web sin la necesidad de cargar la página entera (w3schools.com, 2012).

Este lenguaje combina varias tecnologías (w3schools.com, 2012):

- ✓ Presentación basada en estándares usando XHTML y CSS.
- ✓ Exhibición e interacción dinámicas usando el DOM.
- ✓ Intercambio y manipulación de datos usando XML y XSLT[21].
- ✓ Recuperación de datos asincrónica usando XMLHttpRequest[22].
- ✓ Uso del lenguaje JavaScript.

1.8 Modelado

UML

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema. Proporciona una forma estándar de escribir los planos de un sistema, tales como procesos del negocio, funciones del sistema, clases escritas en un lenguaje de programación específico, esquemas de

[20] *JavaScript Object Notation*
[21] *Extensible Stylesheet Language Transformations*
[22] *Extensible Markup Language / Hypertext Transfer Protocol*

bases de datos y componentes *software* reutilizables. Es la sucesión de una serie de métodos de análisis y diseño orientados a objetos. Está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante UML es posible establecer la serie de requisitos y estructuras necesarias para plasmar un sistema de *software* previo al proceso intensivo de escribir código (Booch, et al., 2005).
Los objetivos de UML se sintetizan en las siguientes funciones (Booch, et al., 2005):

- ✓ **Visualizar:** Expresar de forma gráfica un sistema de forma que otro lo pueda entender.
- ✓ **Especificar:** Especificar cuáles son las características de un sistema antes de su construcción.
- ✓ **Construir:** A partir de los modelos especificados se pueden construir los sistemas diseñados.
- ✓ **Documentar:** Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Altova UModel 2009

Altova UModel es una herramienta de modelado basada en UML. Es compatible con Eclipse y Visual Studio. Permite generar código Java o C# a partir de sus planos, o hacer ingeniería inversa de programas existentes a diagramas UML para abarcar rápidamente su arquitectura de *software*. UModel soporta la especificación de intercambio XMI[23] 2.1 para abrir y editar modelos creados en herramientas Tiene una rica interfaz visual y usabilidad superior que ayudan a disminuir la curva de aprendizaje de UML, permitiendo a los desarrolladores manejar rápidamente UML, potenciando su productividad y maximizando resultados (Altova, 2012).

1.9 Metodologías de desarrollo de *software*

Una metodología de desarrollo de *software* es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo *software*. Puede seguir uno o varios ciclos de vida que indican qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales (Fernández Medina, 2007).

MSF for CMMI

[23] *XML Metadata Interchange*

MSF for CMMI [24] es una metodología ágil de desarrollo de *software* diseñada para ayudar a las organizaciones a alcanzar con mayor rapidez el nivel de madurez 3 de CMMI. Se centra en la gestión de los procesos.

Define cinco fases durante el ciclo de vida del proyecto que encapsula flujo de actividades y actividades. Entre los diferentes roles definidos por esta metodología se encuentran el líder, jefe de desarrollo, jefe de producto, el arquitecto de *software*, desarrollador, analista, diseñadores, probador, integrador, documentador-capacitador, administrador de la calidad, entre otros. Los miembros del equipo de trabajo pueden desempeñar roles distintos durante el ciclo de vida del proyecto y son responsables de cumplir con las actividades y de generar la documentación (Microsoft, 2005).

SXP

SXP consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, que es uno de los requisitos para llegar el éxito del proyecto.

Scrum XP surgió en Cuba, en la Universidad de las Ciencias informáticas (UCI) y consiste en la unión de XP y Scrum para el logro de un buen desarrollo de *software*. XP fue la metodología candidata para guiar el proceso ingenieril, puesto que le precedía su alto grado de aceptación por la comunidad internacional de desarrollo ágil, además que facilitaba una documentación más discreta y mayor dinamismo para el desarrollo. SCRUM es entonces la metodología ideal para toda la gestión de proyectos, sirviendo de soporte para acelerar el dinamismo que se identificó en XP (Peñalver, et al., 2010).

XP (eXtreme Programming)

La programación extrema o XP se basa en una serie de reglas y principios que se han ido gestando a lo largo de toda la historia de la ingeniería del *software*. Se puede englobar dentro de las metodologías ligeras, que son aquellas en la que se da prioridad a las tareas que dan resultados directos y que reducen la burocracia.

Algunos de los roles que esta metodología involucra son (Echeverry Tobón, et al., 2007):

- ✓ **Programador:** Es el responsable de las decisiones técnicas y de construir el sistema sin distinción entre analistas, diseñadores o codificadores. En XP, los programadores diseñan, programan y realizan las pruebas.

[24] *Microsoft Solutions Framework for Capability Maturity Model Integration*

- ✓ **Cliente:** Es parte del equipo, determina qué construir y cuándo, escribe las historias de usuario y establece las pruebas funcionales para validar su implementación.
- ✓ **Encargado de pruebas:** Ayuda al cliente con las pruebas funcionales y se asegura de que las pruebas funcionales se superan.

Los artefactos que genera esta metodología son:

- ✓ **Historias de Usuario:** Son utilizados como herramienta para dar a conocer los requerimientos del sistema al equipo de desarrollo y emplean terminología del cliente sin lenguaje técnico. Se realiza una por cada característica principal del sistema. Las historias de usuario juegan un papel similar a los casos de usos en otras metodologías.
- ✓ **Tareas de ingeniería:** Las tareas de ingeniería son un conjunto de acciones a desarrollar para resolver las historias de usuario que organizan el proceso de implementación. Además posibilitan que sea conocido el grado de complejidad de cada historia de usuario teniendo en cuenta la cantidad de tareas asociadas a ella.
- ✓ **Pruebas de aceptación:** Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación.
- ✓ **Pruebas unitarias:** El código no se considerará completo si este no consta de su unidad de *test* correspondiente.
- ✓ **Plan de entregas:** El plan de entregas define el marco temporal de realización del sistema y se debe trazar en función de dos parámetros: tiempo de desarrollo ideal y grado de importancia para el cliente.

La principal suposición que se realiza en XP es la posibilidad de disminuir la curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a la aplicación disciplinada de las siguientes prácticas (Fernández, 2002).

- ✓ **Planificación:** El cliente establece la prioridad de cada historia de usuario de acuerdo con el valor que aporta para el negocio. Los programadores estiman el esfuerzo asociado a cada historia de usuario. Se ordenan las historias de usuario según prioridad y esfuerzo, y se define el contenido de la entrega.

- ✓ **Entregas pequeñas:** Producir rápidamente versiones del sistema que sean operativas aunque no cuenten con toda la funcionalidad pretendida para el sistema pero sí que constituyan un resultado de valor para el negocio.
- ✓ **Metáfora:** El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema.
- ✓ **Diseño simple:** Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente.
- ✓ **Tarjetas CRC:** Cada tarjeta representa una clase con su nombre, responsabilidades y las clases que le sirven de soporte. Su objetivo es facilitar la comunicación y documentar los resultados.
- ✓ **Pruebas:** La producción de código está dirigida por las pruebas unitarias. Son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse.
- ✓ **Refactorización:** Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios.
- ✓ **Programación en parejas:** Toda la producción de código debe realizarse con trabajo en parejas de programadores. Las principales ventajas de introducir este estilo de programación son que muchos errores son detectados conforme son introducidos en el código, los diseños son mejores y los problemas de programación se resuelven más rápido pues posibilita la transferencia de conocimientos de programación entre los miembros del equipo.
- ✓ **Propiedad colectiva del código:** Cualquier programador puede cambiar cualquier parte del código en cualquier momento.
- ✓ **Integración continua:** Cada pieza de código es integrada en el sistema una vez que esté lista. Todas las pruebas son ejecutadas y tienen que ser aprobadas para que el nuevo código sea incorporado definitivamente.
- ✓ **Cuarenta horas por semana:** Se debe trabajar un máximo de cuarenta horas por semana pues el trabajo extra desmotiva al equipo.

- ✓ **Cliente *in-situ*:** El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Gran parte del éxito del proyecto XP se debe a que es el cliente quien conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada.
- ✓ **Estándares de programación:** XP enfatiza la comunicación de los programadores a través del código, por lo cual es indispensable que se sigan ciertos estándares de programación.

El ciclo de vida de XP incluye cuatro fases que son mostradas a continuación: (Fernández, 2002)

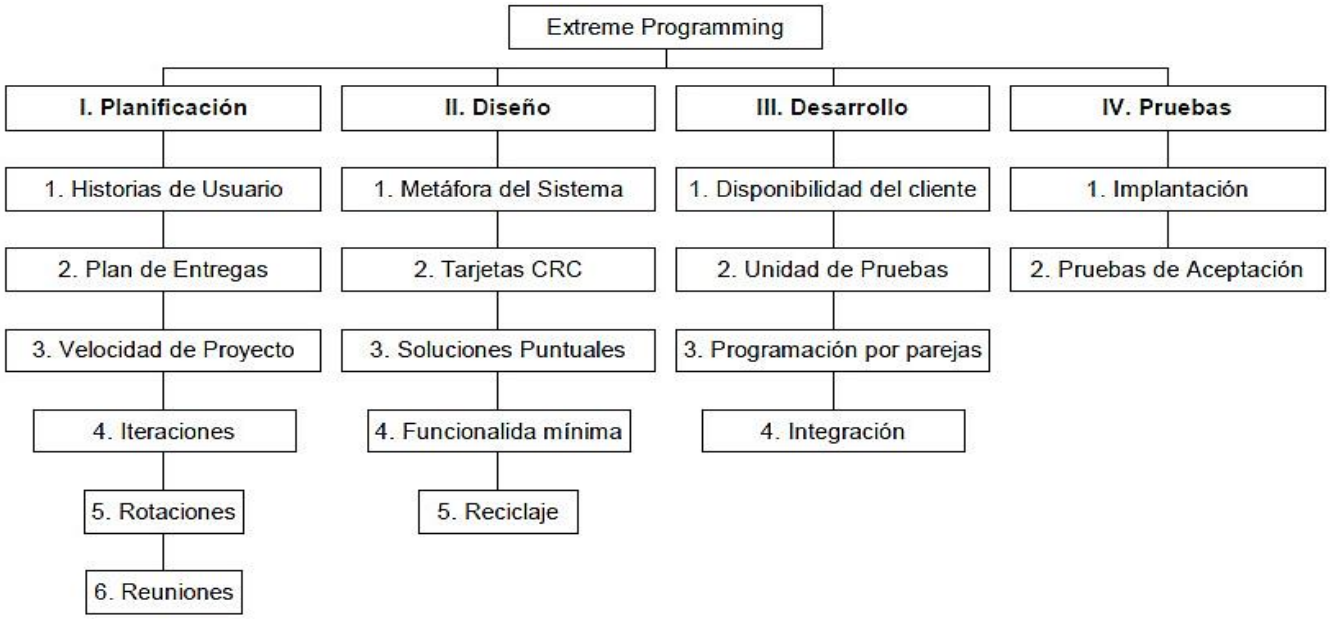


Figura 5. Fases de la metodología XP

Justificación de la metodología utilizada

Luego del análisis de varias metodologías de desarrollo de *software* se decide utilizar XP por las siguientes razones:

- ✓ Constituye una de las metodologías ágiles más exitosas en la actualidad.
- ✓ Encaja perfectamente con el tipo de proyecto, las condiciones y con la idea de desarrollo que se tiene de la solución.
- ✓ Es utilizada en proyectos de corto plazo y con pequeño equipo de desarrollo.

- ✓ Se adapta perfectamente a los proyectos cuyos requerimientos cambian a menudo, es decir a medida que el proyecto avanza pueden surgir nuevas expectativas o ideas que pueden ser incorporadas fácilmente permitiéndole mayor adaptabilidad al producto.
- ✓ Posee pocos roles.
- ✓ Plantea que todos los programadores pueden realizar cambios en cualquier parte del código en cualquier momento.
- ✓ Enfatiza el uso de líneas directivas para la codificación que están bien establecidas, permitiendo una comunicación entre los programadores a través del código.
- ✓ Consiste en una programación rápida o extrema.
- ✓ Tiene como particularidad que el usuario final forma parte del equipo, constituyendo uno de los requisitos fundamentales para llegar al éxito del proyecto.
- ✓ Se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del *software*.
- ✓ Promueve el trabajo en equipo.
- ✓ Se preocupa por el aprendizaje de los desarrolladores, proporcionando un buen clima de trabajo.

Conclusiones

A partir del estudio de los sistemas existentes para graficar mapas conceptuales se logró definir una interfaz visual para el editor de contextos en la que se pudieran representar los conceptos y relaciones de forma sencilla. Este estudio fue necesario debido a que después del análisis de la solución de referencia "Navegador Conceptual" se comprobó que la misma no permite la edición de contextos de forma gráfica.

Se decide que la metodología que guiará el proceso de desarrollo de la solución será XP. Para el modelado de clases se hará uso de la herramienta Altova UModel 2009. El entorno de desarrollo que se utilizará será Visual Studio 2010. Se definió la utilización de los lenguajes C#, CSS, HTML, XML, SVG y JavaScript haciendo uso de la librería jQuery. También se usará la tecnología Ajax para la interacción con servicios web.

Capítulo 2: Características del sistema

Introducción

La realización de un sistema informático conlleva el desarrollo de artefactos que, luego de ser integrados, posibilitan responder a las necesidades del cliente. Estos se van creando por etapas, que van desde la declaración del problema y los requerimientos que necesita el sistema, hasta las pruebas y la liberación del producto.

Las metodologías de desarrollo proveen de una guía que ayuda al grupo de desarrollo a organizarse en tiempo, actividades y artefactos a desarrollar. Dentro de las metodologías de desarrollo, las de enfoque ágil posibilitan que el cliente se sienta como parte del equipo de desarrollo y pueda ir revisando todos los artefactos que se generen; además reducen drásticamente los tiempos de desarrollo pero mantienen una alta calidad.

En el presente capítulo se definen los requisitos funcionales y no funcionales que debe poseer el sistema y las características de la solución propuesta, analizando sus elementos principales. Para ello se hará uso de la metodología ágil XP, de la que se incluyen algunos de sus artefactos como son: las Historias de Usuario arquitectónicamente significativas, el Plan de Entregas y las Tareas de Ingeniería. Se muestra además el Modelo de Dominio, que ayuda a comprender los conceptos asociados a la solución.

2.1 Descripción del sistema

La solución web para la representación de datos y edición de contextos tiene como objetivo mejorar la usabilidad de la herramienta “Navegador Conceptual”. Con esta solución el usuario autenticado podrá acceder a las opciones que brinda la solución como son la de poder realizar la edición de contextos de forma gráfica y navegar a través de la información contenida en las base de datos.

Para acceder al sistema el usuario tiene que estar previamente registrado. Al iniciar sesión el sistema mostrará la página para la navegación, que contará con una barra de menú donde el usuario podrá definir los términos de la navegación. Para cada usuario registrado en el sistema debe existir un directorio que contenga todas las configuraciones realizadas por el mismo así como los proyectos que haya creado.

El sistema debe ser capaz de brindar la posibilidad de seleccionar las bases de datos de las cuales quiera obtener información y los contextos, es decir, que el usuario debe poder definir a qué base de datos se desea conectar y con qué ficheros sintaxis, ontología y descripción desea navegar. La solución brindará la

posibilidad de incorporar nuevos contextos para realizar la navegación, así como definir nuevas bases de datos a las cuales se puedan acceder.

Una vez seleccionados los términos para la navegación, el usuario podrá realizar las consultas de forma gráfica. Las consultas se realizarán en un componente visual, que de forma sencilla y cómoda permitirá al usuario representar la consulta a ejecutar. Este componente visual mostrará la información a partir del fichero ontología y el documento sintaxis del contexto seleccionado y validará la consulta representada antes de realizar la traducción y ejecución en el gestor de base de datos seleccionado. Por defecto se mostrarán los campos pertenecientes al último concepto de la consulta creada por el usuario, sin embargo, se debe brindar la posibilidad de seleccionar las columnas o las expresiones complejas que se deseen mostrar.

El sistema contará además con un historial que contendrá todas las consultas realizadas hasta ese momento, brindando la posibilidad de mostrar el resultado de dicha consulta. El historial será configurable y se podrán eliminar o modificar las consultas seleccionadas.

Un usuario puede realizar varias consultas y cada resultado puede ser añadido a la investigación. En caso de adicionar un resultado a la investigación, el sistema mostrará una pestaña que contiene toda la información de la investigación realizada hasta el momento. La información mostrada en la investigación podrá ser eliminada. Se brindará la opción de realizar una consulta a partir de un resultado mostrado. Se podrá crear o abrir una investigación realizada con el objetivo de poder analizar la misma sin necesidad de estar conectado a la base de datos.

Por otro lado, la página de la navegación contará con un vínculo de acceso a la página del “Editor de Contextos”.

El “Editor de Contextos” tendrá un menú de opciones facilitando al usuario el trabajo con la solución.

La solución brindará las opciones de poder crear, eliminar o modificar conceptos o relaciones al fichero ontología de forma gráfica. En caso de que el usuario seleccione la opción de crear concepto, el sistema debe permitir adicionar, eliminar o modificar campos al concepto. Por cada concepto y relación de la ontología se deberá realizar una consulta SQL que se almacenará en el archivo descripción. Además el usuario debe poder gestionar los operadores según el gestor de base de datos que desee utilizar.

Una vez creado el proyecto, el usuario podrá generar los ficheros XML ontología, sintaxis y descripción, siendo el sistema capaz de validar estos ficheros a partir de los XSD[25] que contienen la estructura correcta para cada uno de ellos.

Los usuarios con sesiones iniciadas en el "Editor de Contexto" podrán acceder a los proyectos creados, brindando la posibilidad de cargarlos, modificarlos y guardarlos. Cada usuario del sistema solo podrá acceder a los ficheros creados por él.

2.2 Modelo de dominio

Luego de haber realizado el análisis correspondiente, se arribó a la conclusión de que la solución web para la representación de datos y edición de contextos de forma gráfica no necesita de un modelado completo del negocio, pues en esta no se identificaron procesos de negocios que estuvieran bien definidos, sólo elementos conceptuales, por lo que se propone la realización de un modelo de dominio que permita obtener una vista general de las entidades o conceptos que intervienen en el negocio debido a que este modelo "*explica los conceptos significativos en un dominio del problema*" (Larman, 2004).

Con la realización del modelo de dominio se pretende ayudar a comprender los conceptos con que interaccionan los usuarios y con los que deberá trabajar la solución. Para su elaboración fue necesario el estudio de las funcionalidades del sistema "Navegador Conceptual", realizándose las siguientes actividades:

- ✓ Identificar clases conceptuales y las relaciones que se establecen entre ellas.
- ✓ Realizar el diagrama de clases.

A continuación se describen todos los conceptos y sus significados, los cuales están presentes en el modelo de domino:

Concepto	Significado
Navegación.	Conjuntos de pasos necesarios para poder acceder a una base de datos determinada y mostrar el resultado de las consultas realizadas.
Ficheros XML.	Archivos necesarios para la navegación que contiene toda la información referente al contexto

[25] *XML Schema Definition*

	(base de datos o negocio).
Ontología.	Fichero XML que contiene las definiciones de conceptos de un dominio y las relaciones entre ellos.
Sintaxis.	Fichero XML en el que se definen todas las funciones y operadores específicos de la problemática a la cual se le esté dando solución y es independiente de la implementación final que se esté utilizando.
Descripción.	Fichero XML que tiene como objetivo representar o detallar el aspecto de cada concepto y relación de la ontología mediante la consulta SQL a la que corresponde en la base de datos.
Fichero de configuración.	Archivo que contiene toda la información gráfica de la ontología.
Proyecto.	Conjunto de archivos pertenecientes a un mismo contexto.
Conceptos.	Conjunto de objetos dentro del contexto.
Relaciones.	Asociación existente entre dos conceptos.
Consultas.	Métodos para acceder a la información persistida en las bases de datos.
Operadores.	Conjunto de operaciones que han de realizarse.

Tabla 1. Conceptos del Modelo de Dominio

Teniendo en cuenta la identificación de los conceptos anteriormente explicados se pudo estructurar el modelo de dominio de la siguiente forma:

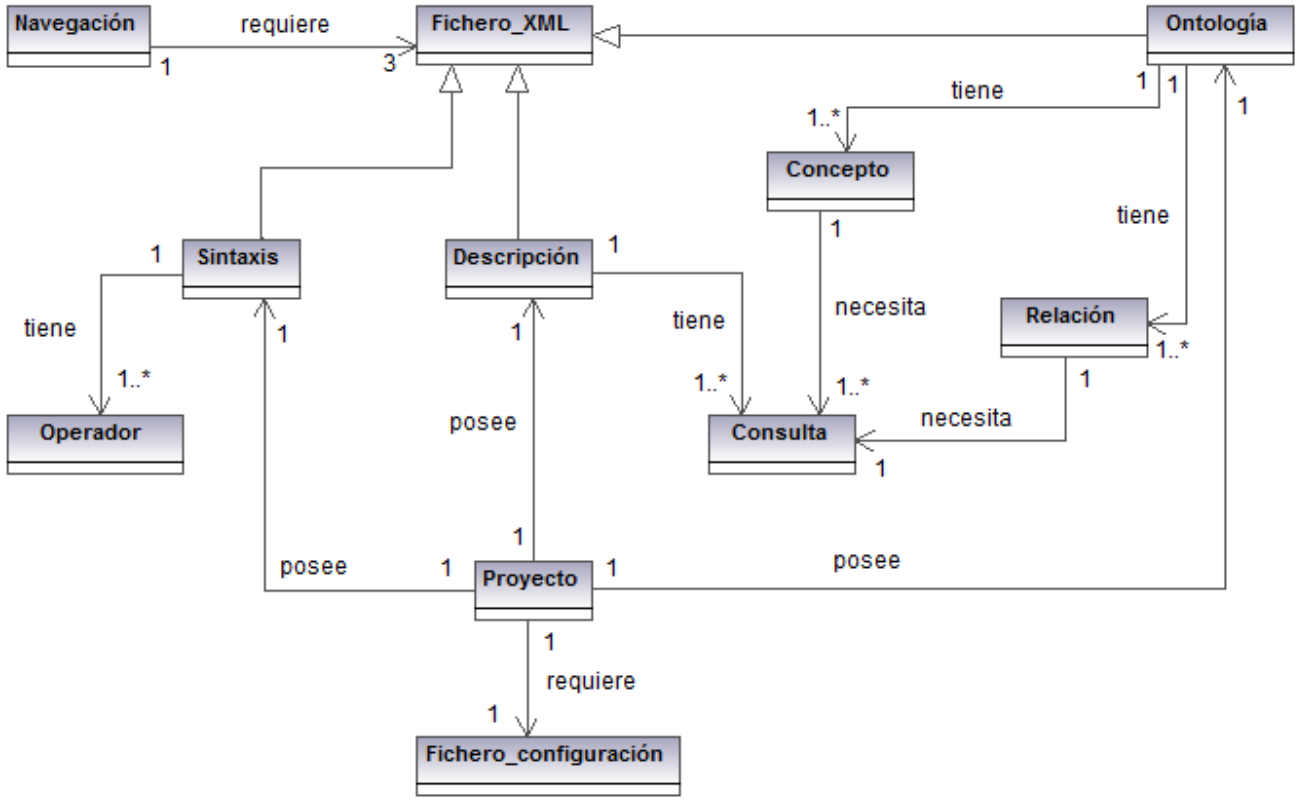


Figura 6. Modelo de dominio

2.3 Captura de requisitos

La captura de requisitos tiene como objetivos:

- ✓ **Enumerar los requisitos candidatos:** Se crea un listado de las funcionalidades que el usuario concibe para el sistema y se indica si se deben incorporar o no al mismo.
- ✓ **Comprender el contexto del sistema:** Se realiza el modelado de dominio donde se describen los principales conceptos.
- ✓ **Capturar requisitos funcionales:** Identificar las funcionalidades que debe cumplir la aplicación.
- ✓ **Capturar requisitos no funcionales:** Identificar las propiedades que debe tener la aplicación

2.3.1 Técnicas de captura de requisitos

Las técnicas de captura de requisitos que se utilizaron en la investigación fueron:

- ✓ **Entrevistas:** se realizaron conversatorios abiertos y cerrados con las personas familiarizadas con la solución a realizar.
- ✓ **Escenarios:** esta técnica constituye un valioso medio para proporcionar contexto a las exigencias del consumidor. En la investigación se usa para propiciar un marco de trabajo con el usuario al permitirle realizar preguntas como: "y si" y "cómo se hace esto".

2.3.2 Requerimientos funcionales del sistema

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir sin alterar la funcionalidad del producto (IEEE, 1990).

Un requisito funcional es una característica requerida del sistema que expresa una capacidad de acción del mismo, una funcionalidad; generalmente expresada en una declaración en forma verbal.

Requisito funcional	Nombre
RF1.	Autenticar usuario
RF2.	Gestionar concepto
RF3.	Gestionar relación
RF4.	Gestionar consulta
RF5.	Gestionar operador
RF6.	Guardar proyecto
RF7.	Cargar proyecto
RF8.	Generar ficheros XML
RF9.	Establecer conexión
RF10.	Realizar consulta
RF11.	Realizar navegación
RF12.	Gestionar investigación

Tabla 2. Requisitos funcionales.

2.3.3 Requerimientos no funcionales del sistema

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Representan las características que hacen al producto atractivo, usable, rápido o confiable (IEEE, 1990).

Un requisito no funcional es una característica requerida del sistema, del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto del desarrollo, que señala una restricción del mismo. Estos constituyen todas las exigencias de cualidades que se imponen al proyecto, ya sean exigencias de usar un cierto lenguaje de programación o una plataforma tecnológica específica.

Requisitos de *software*:

RnF1. Las estaciones de trabajo deben contar con Internet Explorer 9.0 o superior, Mozilla Firefox 1.5 o superior o Chrome (cualquier versión) como navegadores web.

Restricciones en el diseño y la implementación:

RnF2.Metodología Ágil XP, puesto que responde a los cambios rápidamente.

RnF3. Microsoft Visual Studio 2010 como entorno de desarrollo.

Usabilidad:

RnF4. La navegación podrá ser realizada por cualquier usuario que tenga conocimientos básicos del negocio.

RnF5. La edición de contextos podrá ser realizada solamente por personas con amplios conocimientos de SQL y del negocio.

RnF6. La solución debe brindar una interfaz gráfica que posibilite un fácil entendimiento de sus funcionalidades.

RnF7. La solución debe contar con una interfaz gráfica atractiva a la vista del usuario.

Seguridad:

- ✓ Confiabilidad:

  RnF8. Únicamente el administrador tiene el control total del sistema.

  RnF9. Los usuarios deberán estar autenticados antes de realizar cualquier operación.

  RnF10. Los usuarios deben tener roles asignados, asegurando que sólo tengan acceso a las operaciones de acuerdo a los permisos que estos poseen.

- ✓ Integridad:

  RnF11. Los cambios en el sistema sólo pueden ser realizados por las personas autorizadas y de la forma autorizada.

- ✓ Disponibilidad:

  RnF12. El sistema podrá ser usado en cualquier momento por todos los usuarios autorizados.

  RnF13. Las fallas del *software* se dividirán en dos categorías:

Simples: la solución y actualización se realizará en el momento.

Complejas: la solución y actualización se realizará en un tiempo que se definirá posterior a una evaluación detallada.

Luego de haber realizado la captura de requisitos, se identificaron doce requisitos funcionales y trece requisitos no funcionales. En la Tabla 3 se muestra la clasificación que se define para cada requisito funcional y en la Tabla 4 se pueden observar las diferentes clasificaciones de los requisitos no funcionales.

Prioridad	Cantidad
Alta	3
Media	8
Baja	1
Total	**12**

Tabla 3. Prioridad de los Requisitos funcionales

Clasificación	Cantidad
Software	1
Usabilidad	4
Seguridad	6
Restricciones de diseño e implementación	2
Total	**13**

Tabla 4. Requisitos no funcionales

2.4 Actores del sistema

Actor	Descripción
Usuario	Usuarios que acceden al sistema y tienen la posibilidad de realizar la navegación y la edición de contextos.
Administrador	Persona encargada de administrar los datos de los usuarios que se registran en la aplicación y de las acciones a realizar por dicho sistema.

Tabla 5. Actores del sistema

2.5 Historias de usuario

La identificación de las Historias de Usuario es la técnica utilizada para especificar los requisitos del *software*. Tienen el mismo propósito que los casos de uso, pero no son lo mismo, las escriben los propios clientes tal y como ven ellos las necesidades del sistema, por tanto son descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica.

El tratamiento de las Historias de Usuario es muy dinámico y flexible. Cada una es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Los desarrolladores deben hacer una estimación de cuánto tiempo, idealmente, requerirá la implementación de cada historia de usuario. Las condiciones ideales son aquellas en las que se codifica la historia de usuario sin otras distracciones y sabiendo exactamente qué es lo que hay que implementar. Como resultado se debe obtener un periodo ideal de una, dos o tres semanas. Más de tres semanas implica que se deben dividir la historia de usuario en partes. Menos de una semana implica que la historia de usuario es demasiado sencilla por lo que se deben unir dos o más de ellas. (Echeverry Tobón, et al., 2007)

A continuación se muestra la "Historia de Usuario Gestionar concepto" como ejemplo de la identificación de las historias de usuario. El resto de las historias de usuario se pueden encontrar en el Anexo 1.

Historia de Usuario	
Código: HU02	**Nombre Historia de Usuario:** Gestionar concepto.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: RF2.	
Prioridad: Media.	**Iteración Asignada:** 2
Riesgo en Desarrollo: Medio.	**Puntos Estimados:** 3 semanas.
Asignado a: Yisel Correa Rodríguez.	**Puntos Reales:** 3 semanas.
Descripción: El sistema debe brindar la posibilidad de adicionar un concepto (nombre, descripción, campos, índices y etiquetas) al seleccionar la opción "Adicionar concepto". El mismo debe ser creado dentro del área de trabajo especificada. Al dar doble clic sobre el concepto, el sistema debe mostrar	

una ventana con su información (nombre, descripción, campos, índices, etiquetas) y dar la posibilidad de modificar esa información. Al seleccionar la opción “Eliminar concepto” y dar clic sobre un concepto el sistema debe eliminar el mismo. También debe dar la posibilidad de buscar el concepto y localizarlo dentro del área de trabajo.
Observaciones: Si el concepto tiene relaciones asociadas, estas tienen que ser eliminadas también.

Tabla 6. Historia de usuario: Gestionar concepto

Las interfaces definidas para dar cumplimiento a las Historias de Usuario se encuentran en el Anexo 2.

2.6 Plan de entregas

El plan de entregas define el marco temporal de realización del sistema y se debe trazar en función de dos parámetros: tiempo de desarrollo ideal y grado de importancia para el cliente. El plan de entregas se usará para crear los planes de iteración para cada iteración.

No.	Historia de usuario	Prioridad	Riesgo	Iteración
1	Autenticar usuario	Baja	Medio	5
2	Gestionar concepto	Media	Medio	2
3	Gestionar relación	Media	Medio	2
4	Gestionar consulta	Media	Medio	3
5	Gestionar operador	Media	Medio	3
6	Guardar proyecto	Media	Medio	4
7	Cargar proyecto	Media	Medio	4
8	Generar ficheros XML	Alta	Alto	1
9	Establecer conexión	Media	Medio	4
10	Realizar consulta	Alta	Alto	1
11	Realizar navegación	Alta	Alto	1
12	Gestionar investigación	Media	Medio	4

Tabla 7. Plan de entregas

Iteración 1: Se codificarán las historias de usuario de mayor prioridad y riesgo como son las de Generar ficheros XML, Realizar consulta y Realizar navegación.

Iteraciones 2, 3 y 4: Se codificarán las historias de usuario con prioridad media y riesgo de desarrollo medio. Estas son Gestionar concepto, Gestionar relación, Gestionar consulta, Gestionar operador, Establecer conexión, Guardar proyecto, Cargar proyecto y Gestionar investigación.
Iteración 5: Se codificará la historia de usuario con prioridad baja Autenticar usuario.

2.7 Tareas de ingeniería

Al comenzar cada iteración las historias de usuario son descompuestas en tareas y asignadas a los desarrolladores, los cuales estiman el tiempo necesario para su realización. La estimación de estos tiempos varía entre uno a tres días ideales, es decir, sin distracciones. Estas estimaciones son más exactas que las realizadas en el Plan de entregas.
Las tareas de ingeniería son un conjunto de acciones a desarrollar para resolver las historias de usuario que organizan el proceso de implementación. Además posibilitan que sea conocido el grado de complejidad de cada historia de usuario teniendo en cuenta la cantidad de tareas asociadas a ella.
A continuación se muestran las tareas para la "Historia de Usuario Gestionar concepto" correspondiente a la segunda iteración. El resto de la Tareas de ingeniería se pueden encontrar en el Anexo 3.

Iteración 2:

Tarea de ingeniería	
Número Tarea: 7	**Número de Historia de Usuario:** 2
Nombre Tarea: Adicionar concepto.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Febrero.	**Fecha Fin:** Febrero.
Programador Responsable: Yisel Correa Rodríguez.	
Descripción: El sistema debe dar la posibilidad de adicionar un concepto (nombre, descripción, campos, índices, etiquetas) a la ontología.	

Tabla 8. Tarea de ingeniería: Adicionar concepto

Tarea de ingeniería

Número Tarea: 8	**Número de Historia de Usuario:** 2
Nombre Tarea: Modificar concepto.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Febrero.	**Fecha Fin:** Febrero.
Programador Responsable: Yisel Correa Rodríguez.	
Descripción: El sistema debe permitir modificar los campos (nombre, descripción, campos, índices, etiquetas) de un concepto seleccionado.	

Tabla 9. Tarea de ingeniería: Modificar concepto

	Tarea de ingeniería
Número Tarea: 9	**Número de Historia de Usuario:** 2
Nombre Tarea: Eliminar concepto.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Febrero.	**Fecha Fin:** Febrero.
Programador Responsable: Yisel Correa Rodríguez.	
Descripción: El sistema debe brindar la posibilidad de eliminar un concepto seleccionado de la ontología.	

Tabla 10. Tarea de ingeniería: Eliminar concepto

	Tarea de ingeniería
Número Tarea: 10	**Número de Historia de Usuario:** 2
Nombre Tarea: Buscar concepto.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1

Fecha Inicio: Febrero.	**Fecha Fin:** Febrero.
Programador Responsable: Yisel Correa Rodríguez.	
Descripción: El sistema debe permitir localizar un concepto determinado dentro del área de trabajo.	

Tabla 11. Tarea de ingeniería: Buscar concepto

Conclusiones

En este capítulo se realizó la descripción del sistema y la planificación del mismo a través de la realización de las Historias de Usuario, el Plan de Entregas y las Tareas de Ingeniería, todos estos conceptos fundamentales manejados por esta etapa dentro de la metodología utilizada. Se definieron doce requerimientos funcionales y trece no funcionales. Se identificaron las personas que intervendrán en la aplicación y se realizó una estimación del tiempo que se empleará para implementar las Historias de Usuarios, dicha implementación se llevará a cabo en cinco iteraciones.

Capítulo 3: Diseño, implementación y validación de la solución

Introducción

En el presente capítulo se modelan algunos artefactos generados por la metodología ágil XP como el Diagrama de despliegue. Se describen los patrones de diseño así como la arquitectura a tener en cuenta en el sistema. Se especifican además los estándares de diseño y de codificación para la creación de la solución.

3.1 Tarjetas CRC

Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Estas posibilitan que el equipo completo contribuya en la tarea del diseño.

A continuación se presenta la tarjeta CRC para la clase *ConceptOntology,* las demás se encuentran en el Anexo 4.

Tarjeta CRC	
Clase: ConceptOntology	
Responsabilidades	**Colaboraciones**
Almacenar toda la información referente a la ontología. Controlar la gestión de los conceptos y las relaciones. Brindar la información necesaria a los elementos representados en el editor de contextos.	OntologyConcept OntologyRelation Description

Tabla 12. Tarjeta CRC de la clase ConceptOntology

3.2 Diagrama de clases

Un diagrama de clases tiene como propósito representar los objetos fundamentales del sistema, es decir los que percibe el usuario y con los que espera tratar para completar su tarea. Sirve para visualizar las relaciones entre las clases que involucran el sistema.

A continuación se presenta el diagrama de clases de la solución propuesta y una breve explicación de las mismas. El diagrama solamente muestra los nombres de las clases, para ver el diagrama completo ver el Anexo 5.

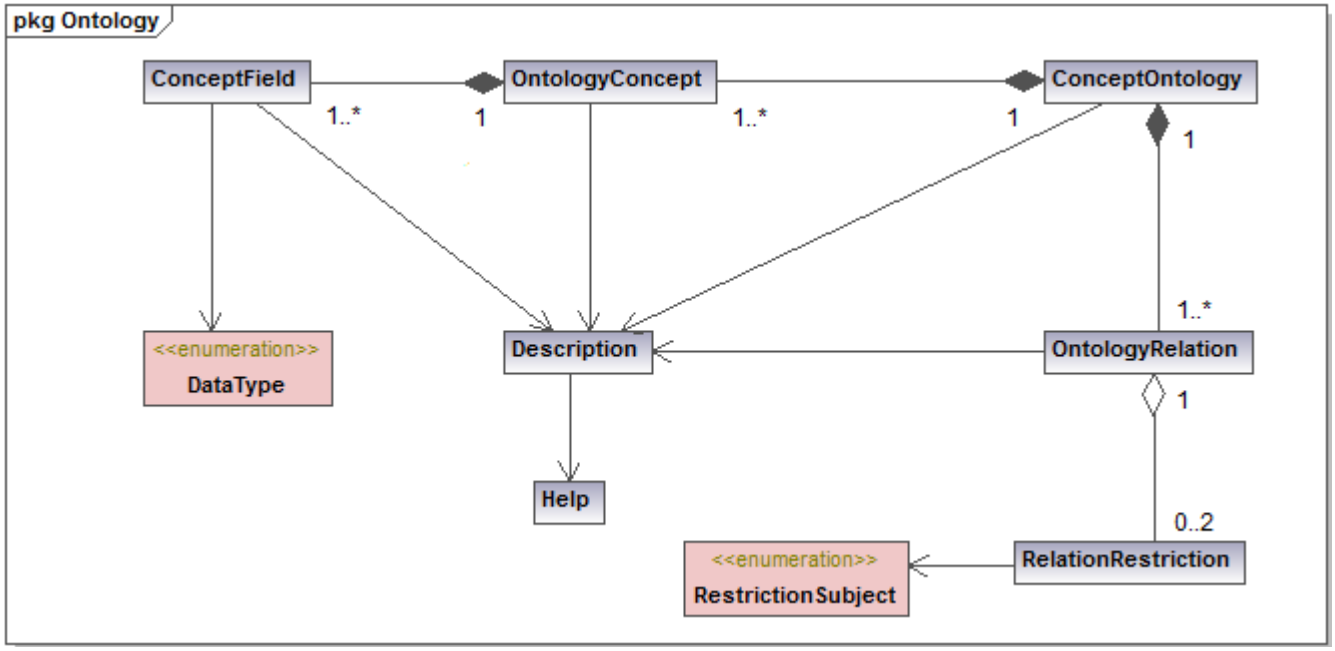


Figura 7. Diagrama de clases del paquete Ontología

En el paquete de la Ontología se encuentran las siguientes clases:

- ✓ **ConceptOntology:** Es la clase que se encarga de manipular los conceptos y relaciones del negocio.
- ✓ **OntologyConcept:** Contiene la información relacionada a los conceptos.
- ✓ **ConceptField:** Es la clase que contiene la información de los campos asociados al atributo *_fieldsField* del concepto.
- ✓ **OntologyRelation:** Contiene toda la información de la relación.
- ✓ **RelationRestriction:** Es la clase que contiene la información asociada a las restricciones incluidas en una relación.
- ✓ **Description:** Contiene la información de ayuda que es almacenada en la clase *Help*.
- ✓ **DataType:** Clase de tipo enumerativo que contiene los tipo de datos asociados al atributo *_fieldTypeField.* Estos pueden ser *string, number, data, bytearray,* xml*, image..*

- ✓ **RestrictionSubject:** Enumerativo que contiene los tipos de sujeto pertenecientes a una restricción de la relación. Pueden ser *OwnerConcept* o *RelatedConcept.*

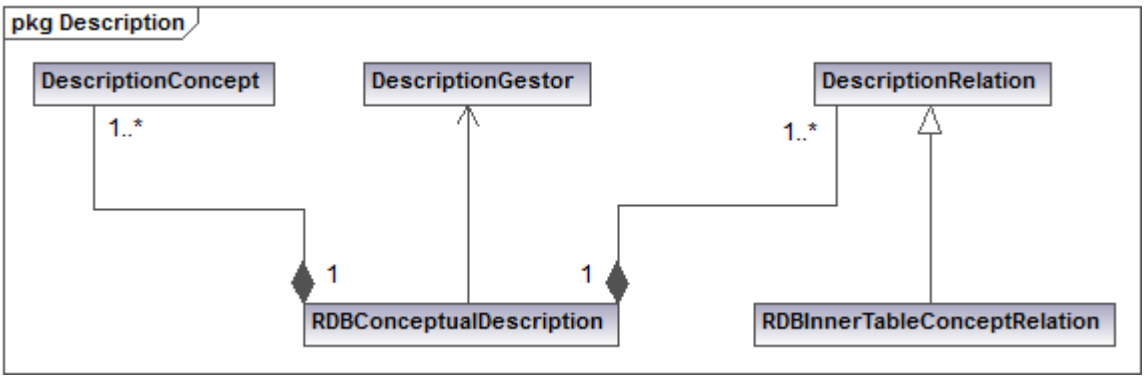


Figura 8. Diagrama de clases del paquete Descripción

En el paquete de la Descripción se encuentran las siguientes clases:

- ✓ **RDBConceptualDescription:** Es la clase que se encarga de adicionar elementos de la descripción a los conceptos y las relaciones del negocio.
- ✓ **DescriptionConcept:** Contiene los elementos de la descripción para un concepto.
- ✓ **DescriptionRelation:** Contiene los elementos de la descripción para una relación simple.
- ✓ **RDBInnerTableConeptRelation:** Contiene elementos de la descripción para una relación N-M; es una extensión de la clase *DescriptionRelation*.
- ✓ **DescriptionGestor:** Contiene la información asociada al campo *_gestorField* de la clase *RDBConceptualDescription*.

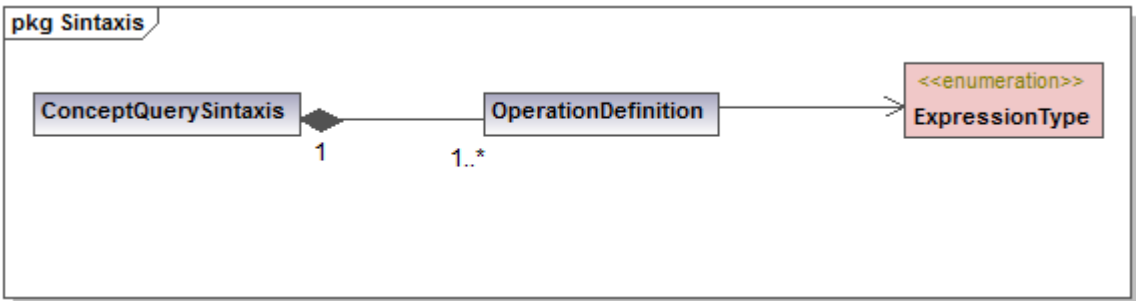


Figura 9. Diagrama de clases del paquete Sintaxis

En el paquete de la Sintaxis se encuentran las siguientes clases:

- ✓ **ConceptQuerySintaxis:** Es la clase que contiene la información de la sintaxis.

- ✓ **OperationDefinition:** Contiene la información del campo *_operationDefinitionField* de la clase *ConceptQuerySintaxis*.
- ✓ **ExpressionType:** Enumerativo que define el tipo de expresión asociada a los parámetros y el valor de retorno de un operador o de una función. Estos valores pueden ser *string, number, data*, *bytearray,* xml*, image, boolean,* conceptquery*, table, null*.

3.3 Arquitectura

La Arquitectura de *Software* es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (IEEE 1471-2000).

Estilo arquitectónico

Se puede definir un estilo arquitectónico como un conjunto de reglas de diseño que identifican las clases de componentes y conectores que se pueden utilizar para componer un sistema o subsistema, junto con las restricciones locales o globales de la forma en que la composición se lleva a cabo (Díaz Pastrana, et al., 2008).

Dentro de los diferentes grupos de estilos existentes se encuentran los estilos jerárquicos. Para la estructuración de la arquitectura del sistema se adoptó el estilo Cliente-Servidor que se muestra en la Figura 10. La arquitectura cliente/servidor consiste en varios clientes distribuidos en diferentes nodos, conectados en red a uno o varios nodos servidores, donde el servidor puede atender a varios clientes a la vez. En los nodos clientes se encuentra la presentación de usuario y en los nodos servidores la lógica del negocio. La arquitectura cliente/servidor es una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realiza se efectúa con la mayor eficiencia posible y posibilita simplificar las actualizaciones y mantenimiento del sistema.

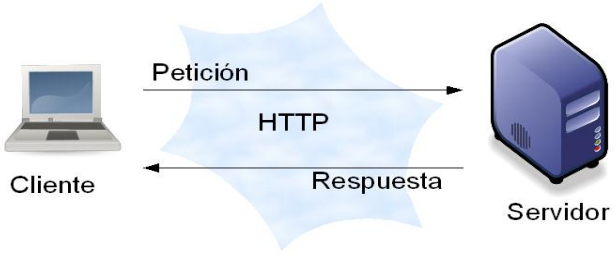


Figura 10. Estilo arquitectónico Cliente-Servidor

Arquitectura n-capas

Los sistemas o arquitecturas en capas se definen como una organización jerárquica en la que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas. Sus principios fundamentales son (Pelaez , 2009):

- ✓ Es un estilo para definir el despliegue de las capas en una instalación.
- ✓ La arquitectura de N-capas está caracterizada por la descomposición funcional de la aplicación, los componentes de servicio y su instalación distribuida, mejorando la escalabilidad, disponibilidad, administración, y utilización de recursos.
- ✓ Cada capa es completamente independiente de las otras capas, excepto aquella que está inmediatamente debajo de ella. La capa n sólo necesita saber cómo manejar una solicitud de la capa n+1, cómo hacer la solicitud a la capa n-1 (si existe) y cómo manejar el resultado de la petición.

A partir de las características de la arquitectura n-capas se definieron las siguientes capas para la solución.

- ✓ **Presentación:** Está compuesta por todas las interfaces de usuarios y los componentes necesarios para su correcto funcionamiento, dígase JavaScript, CSS, y servicios consumidos por Ajax. Además, en la misma se hace uso de las facilidades que brinda la librería jQuery para la construcción de interfaces amigables a la vista del usuario. Esta capa tiene interacción directa con la capa de negocio, permite la visibilidad de los datos de entrada y muestra los datos de salida correspondientes al proceso de navegación.
- ✓ **Negocio:** Es donde se realizan todas las funcionalidades del sistema, se reciben las peticiones del usuario y se envían las respuestas luego de ejecutada la petición. Esta capa se comunica con la capa de presentación para recibir las solicitudes y mostrar los resultados, además tiene comunicación directa con la capa de acceso a datos. Dentro de los principales conceptos que se manejan en esta capa se encuentra la librería ConceptualMap.dll, encargada de la manipulación del negocio en lo concerniente a la navegación conceptual. Esta capa está conformada también por los paquetes de clases de Ontología, Sintaxis y Descripción, que son las responsables del negocio del editor de contextos.

- ✓ **Acceso a datos:** Constituye la capa que sirve como puente entre la capa del negocio y la capa de base de datos. Esta capa pretende encapsular las especificidades del proveedor de datos, dígase Oracle, SqlServer, OleDB, o archivos XML. Tiene como objetivo mantener la comunicación con los proveedores y en caso de existir cambios en alguno de estos no se afectarían el resto de las capas. Dentro de la misma se encuentran ubicadas las clases encargadas de establecer la conexión con el proveedor de datos.
- ✓ **Base de datos:** Brinda la información necesaria a la capa de acceso a datos, permitiendo dar respuestas a las peticiones de los usuarios.

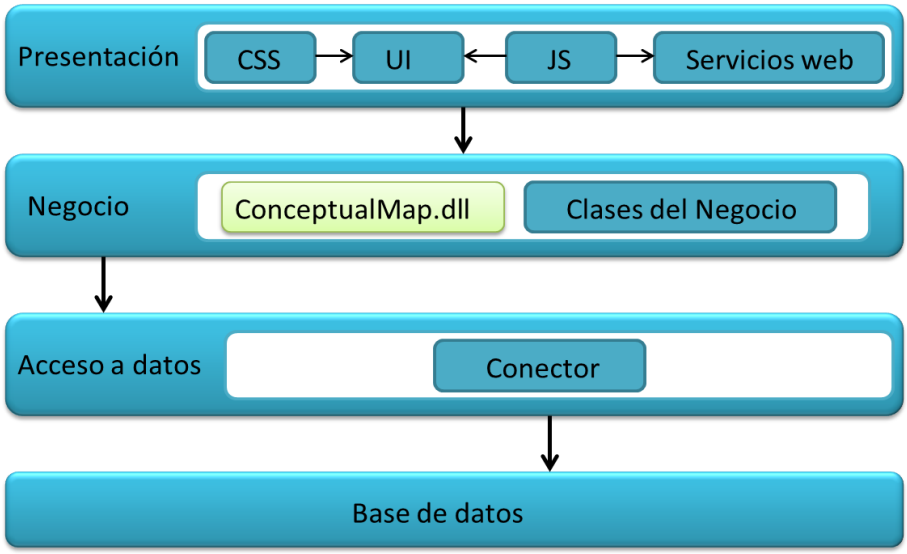


Figura 11. Diagrama de arquitectura

3.4 Patrones

Un patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas (Larman, 2004).

Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del *software*. Todo lo contrario, intentan codificar el conocimiento, las expresiones y los principios ya existentes.

Patrones de diseño

Los patrones de diseño "s*on soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y se ha demostrado que funcionan"* (García, 2005).

Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifican y describen formas de solucionar problemas que ocurren de forma frecuente en el desarrollo.
Dentro de los patrones que se utilizan en el diseño están dos grupos fundamentales: los patrones conocidos como GRASP[26] y los patrones GOF[27], teniendo cada una de estas agrupaciones varios patrones que se aplican en la solución de un problema. Los patrones de encapsulamiento y subclase no pertenecen a ninguno de los grupos mencionados anteriormente.
A continuación se presentan los patrones de diseño utilizados en la solución:

- ✓ **Encapsulamiento:** propone esconder algunos componentes permitiendo sólo accesos estilizados a objetos. Se hace uso de este patrón en todas las clases, logrando que estas sólo posean como elementos públicos aquellos que son exclusivamente necesarios.

```
public UsersUser[] User
{
    get
    {
        return this.userField;
    }
    set
    {
        this.userField = value;
    }
}
```

Figura 12. Patrón encapsulamiento

- ✓ **Subclase:** propone heredar miembros por defecto de una superclase[28], seleccionando la implementación correcta a través de resoluciones sobre qué implementación debe ser ejecutada. Se puede encontrar este patrón en las clases de conexión como se muestra a continuación.

[26] *General Responsibility Assignment Software Patterns* o Patrones Generales de *Software* para Asignar Responsabilidades
[27] *Gang of Four*
[28] Término utilizado en la programación orientada a objetos para nombrar a las clases padres, es decir, aquellas clases que tendrán atributos y/o métodos que pueden ser heredados por las clases hijas o subclases.

```
public class OleDbConnector: Connector
{
    private readonly string _location;
    private OleDbConnection _oleDbConnection;
    private OleDbConnectionStringBuilder _oleDbConnectionStringBuilder;

    private OleDbConnector(string user, string password, string location):base(user, password)
    {
        _location = location;
    }
```

Figura 13. Patrón subclases

Patrones GRASP

GRASP, nombre elegido para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el *software* orientado a objetos (Larman, 2004). Para la realización de la solución se utilizaron los patrones: Experto, Creador, Alta Cohesión, Bajo Acoplamiento y Controlador, todos incluidos dentro del grupo de patrones GRASP. A continuación se dará una breve descripción de estos patrones y ejemplos donde su uso fue necesario.

- ✓ **Bajo acoplamiento:** El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases. Acoplamiento alto significa que una clase recurre a muchas otras clases. El bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios y que aumentan la oportunidad de una mayor productividad (Larman, 2004). Este patrón se utilizó con el fin de tener las clases con la menor dependencia posible una de la otra para que en caso de producirse una modificación en alguna tuviera una mínima repercusión en el resto, potenciando la reutilización y disminuyendo la dependencia.
- ✓ **Alta Cohesión:** La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Este patrón mejora la calidad y facilidad del diseño, genera un bajo acoplamiento y promueve la reutilización (Larman, 2004). En la solución es necesario controlar la complejidad de cada clase utilizada para mantener un buen comportamiento de las mismas, por esa razón, las clases identificadas con una gran cantidad de

funcionalidades se dividieron en otras clases de manera que se repartiera equitativamente el peso de la complejidad, manteniendo además la coherencia entre las mismas.

- ✓ **Experto:** Permite asignar una responsabilidad al experto en información, es decir, la clase que cuenta con la información necesaria para cumplir la responsabilidad. El uso de este patrón se evidencia en las clases que se utilizan para realizar las conexiones a las bases de datos.
- ✓ **Creador:** Tiene como propósito fundamental encontrar un creador que se conecte con el objeto producido en cualquier evento. Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos.
- ✓ **Controlador:** La mayor parte de los sistemas reciben eventos de entrada externa. En estos casos hay que elegir controladores que manejen esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas en la elección de los controladores de eventos. Su utilización propicia que las operaciones del sistema se manejen en la capa de dominio de los objetos, y no en la de presentación (Larman, 2004). Se utilizó para garantizar la comunicación entre los eventos externos del sistema y los componentes de la capa de negocio.

Patrones GoF

Los patrones de diseño GoF deben su nombre a sus creadores: Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides. Estos tuvieron un gran éxito en el mundo de la informática a partir de la publicación del libro "*Design Patterns*" en el cual se recogen veintitrés patrones de diseño comunes.

Estos veintitrés patrones se dividen en tres grupos de acuerdo a la naturaleza de cada uno. (Gamma, et al., 2005)

- ✓ **Creación:** Esta categoría agrupa a los patrones que proporcionan guías de cómo construir objetos, cuando su creación implique la toma de una decisión. Esta decisión puede ser básicamente elegir qué subclase dentro de una jerarquía de herencia instanciar, y qué clase tiene la responsabilidad de su creación.
- ✓ **Estructurales:** Los patrones de esta categoría describen mecanismos genéricos para organizar diferentes clases de objetos entre sí.
- ✓ **Comportamiento:** Caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos.

A continuación se dará una breve descripción de estos patrones y ejemplos donde su uso fue necesario.

- ✓ **Interfaz:** Permite definir un comportamiento similar a un grupo de clases que implementan un modelo *interface* previamente definido. Todas las clases o instancias que quieran utilizar ese comportamiento deberán implementar dicha interfaz. Un ejemplo de la utilización de este patrón se ve en la implementación de la interfaz IConnector.

```
public interface IConnector
{
    string GetConnectionString();
    void OpenConnection();
    void CloseConnection();
    DataTable ExecuteQuery(string sql);
}
```

Figura 14. Patrón interfaz

- ✓ **Singleton:** Asegura que sólo se pueda crear una instancia de la clase y ofrece un punto global de acceso a esta instancia. El uso de este patrón permite que los servicios creen una instancia de los objetos sólo una vez.

```
public class XmlManager
{
   private static XmlManager _instance = null;

    private XmlManager(){ }

    public static XmlManager GetXmlManager()
    {
        if (_instance == null)
            _instance = new XmlManager();
        return _instance;
    }
```

Figura 15. Patrón singleton

3.5 Estándares de diseño

Una interfaz bien definida facilita el trabajo de los usuarios, por esa razón se establecen reglas de diseño que guían al diseñador a tomar decisiones correctas en sus diseños con el fin de conseguir productos usables. A continuación se describen las reglas de diseño que se deben adoptar para la realización de la aplicación.

1. **Pauta cromática**

Uno de los elementos a tener en cuenta en el diseño es la pauta cromática, a partir de la cual se especifican los colores a usar en la realización de la aplicación.
Se definió que se usaría el tema *start* de las librerías de jQuery UI.

2. **Acciones**

Menús:

- ✓ Menús desplegables.
- ✓ Menús en cascada.

Cajas de diálogo:

- ✓ Presentan información.
- ✓ Ventana móvil de tamaño fijo.
- ✓ Aparece durante el procesamiento de una acción del usuario, cuando se requiere información para completarla.
- ✓ No usan menús. Usan botones para llamar a las acciones.
- ✓ Botones: aceptar, cancelar.

Tipos de cajas de diálogo:

- ✓ **No modal:** Permite a los usuarios continuar con su trabajo sin completar el diálogo.
- ✓ **Modal:** Requiere que los usuarios completen la caja de diálogo antes de continuar.

Caja de mensajes: Es un tipo especial de caja de diálogo que se utiliza exclusivamente para mostrar mensajes a los usuarios.

3. **Interacción**

- ✓ **Selección de objetos:** Los usuarios apuntan a un objeto que desean manipular y lo seleccionan de manera visible.
- ✓ **Ejecución de la acción:** Se selecciona una opción de menú y si es preciso se completa con una caja de diálogo. La ejecución de la acción debe ser visualizada.

Apuntar y seleccionar:

- ✓ Los usuarios interaccionan con los componentes de la interfaz.
- ✓ Apuntan a lo que desean manipular y lo seleccionan.
- ✓ Se utiliza solamente el ratón (el puntero indica la posición del ratón).
- ✓ El ratón tiene una indicación visual para indicar al usuario dónde se encuentra.

4. Componentes

Se seleccionaron varios componentes para facilitar la implementación y uniformidad de las funcionalidades de la aplicación. Algunos de estos son:

- ✓ Campo de texto: Componente que crea un control de entrada de una sola línea.
- ✓ Área de texto: Componente que crea un control de entrada de varias líneas.
- ✓ Botones de comprobación: Componente para listar opciones, permitiendo la elección de más de un componente.
- ✓ Botones de radio: Componente para listar opciones. Se puede seleccionar un componente solamente.
- ✓ Botones pulsables: Componente para ejecutar una acción.

3.6 Estándares de codificación

Las convenciones o estándares de codificación son un conjunto de directrices que especifican cómo debe escribirse el código fuente. Algunos ejemplos de directrices que podemos encontrar son aquellas que nos pautan el nombrado de variables, clases y paquetes, cómo escribir los bucles y estructuras de control o incluso qué información incluir en los comentarios. Su importancia radica en que facilitan la compresión del código y, por tanto, permiten a los desarrolladores tratar cada porción del mismo como si de su propio código se tratara (Herranz, 2010).

Un estándar de codificación posibilita crear un patrón de referencia a la hora de implementar un sistema. Los estándares de código se definen según el estilo del programador, del lenguaje que se vaya a utilizar y el tipo de programa que se desea implementar.

Para el desarrollo de la aplicación se establecieron estándares de codificación que garantizaran la legibilidad en el código y la organización del mismo para su posterior mantenimiento. A continuación se especifican varios de los estándares de código presentes en el desarrollo de la aplicación.

Comentarios de funciones primarias

- ✓ Objetivo de la función y no descripción del procedimiento.
- ✓ Comentarios de apoyo a variables, llamadas a función o inclusión de archivos que no sean obvios al proceso.
- ✓ Explicación de uso de parámetros no obvios.
- ✓ Explicación de uso de valores de retorno.

Nombres de identificadores

Se considera como identificador a los nombres de variables (arreglos, matrices, apuntadores), funciones, así como cualquier tipo de dato definido por el usuario (estructura, clase).

- ✓ Deberán tener un nombre significativo para que por su simple lectura, pueda conocerse su función, sin tener que consultar manuales o hacer demasiados comentarios.
- ✓ Para nombres que se usen con frecuencia o para términos largos, se deben usar abreviaturas estándar para que éstos tengan una longitud razonable.
- ✓ Los identificadores de variables deberán regirse por la nomenclatura *CamelCase* [29] siempre comienzan con letra minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con letra mayúscula.
- ✓ Los identificadores de funciones deberán comenzar siempre con letra mayúscula y para distinguir palabras dentro del nombre deberá emplearse una letra mayúscula.

Organización visual

- ✓ Declarar las variables en líneas separadas.
- ✓ Añadir comentarios descriptivos junto a cada declaración de variables, si es necesario.
- ✓ Las declaraciones de datos dentro de una función, deberán ir al inicio y separadas de las instrucciones ejecutables de la función por medio de una línea en blanco.
- ✓ Los operadores unarios deben ponerse junto a sus operandos, sin espacios intermedios.
- ✓ Para hacer más clara una expresión, es aceptable agregarle paréntesis innecesarios. Dichos paréntesis se llaman paréntesis redundantes.

Declaraciones de clases e interfaces

- ✓ No hay espacio entre el nombre del método, el paréntesis y la lista de parámetros.
- ✓ La llave de apertura del método se pone al final de la línea de declaración o en la línea continua.
- ✓ La llave de cierre debe aparecer en una línea aparte.

[29] *CamelCase* es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en *CamelCase* se asemejan a las jorobas de un camello.

3.7 Tratamiento de errores

En la construcción de un *software* se debe tener en cuenta el tratamiento de errores, que constituye la acción de identificar, localizar, analizar y eliminar los errores que se producen en el proceso de creación del mismo.

El tratamiento de errores de los sistemas automatizados debe estar encaminado a prevenir los errores que puedan ser provocados por los usuarios a la hora de introducir datos como los que puedan ser generados por el comportamiento incorrecto de los componentes internos, por ejemplo las consultas a bases de datos.

Los errores durante la ejecución de un programa son inevitables, por lo que un manejo elegante de los mismos es muy importante.

1. Al intentar ejecutar un bloque de código se debe decidir qué hacer si se produce una circunstancia excepcional durante su ejecución.
2. Si se produce la circunstancia se "lanza" una excepción, en caso contrario el programa sigue su curso normal.
3. La ejecución del programa es desviada a un sitio específico donde la excepción es "capturada" y se decide qué hacer al respecto.

En la solución, los datos que introduce el usuario son validados en los formularios mediante JavaScript, mostrando mensajes de error encima de los componentes que indican al usuario cuáles datos están incorrectos o cuáles son obligatorios.

El tratamiento de errores en los componentes internos se implementa en los métodos y cuando se detecta una posible excepción se muestra un mensaje de error.

3.8 Diagrama de despliegue

El diagrama de despliegue es un modelo que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño (Jabconson, et al., 2000).

Un diagrama de despliegue muestra cómo y dónde se desplegará el sistema. Las máquinas físicas y los procesadores se representan como nodos, y la construcción interna puede ser representada por nodos o

artefactos embebidos. Como los artefactos se ubican en los nodos para modelar el despliegue del sistema, la ubicación es guiada por el uso de las especificaciones de despliegue.

A continuación se muestra el diagrama de despliegue para la propuesta de solución, el cual está formado por las máquinas clientes, el servidor de aplicaciones y los servidores de base de datos externos a la solución pero necesarios en el proceso de navegación.

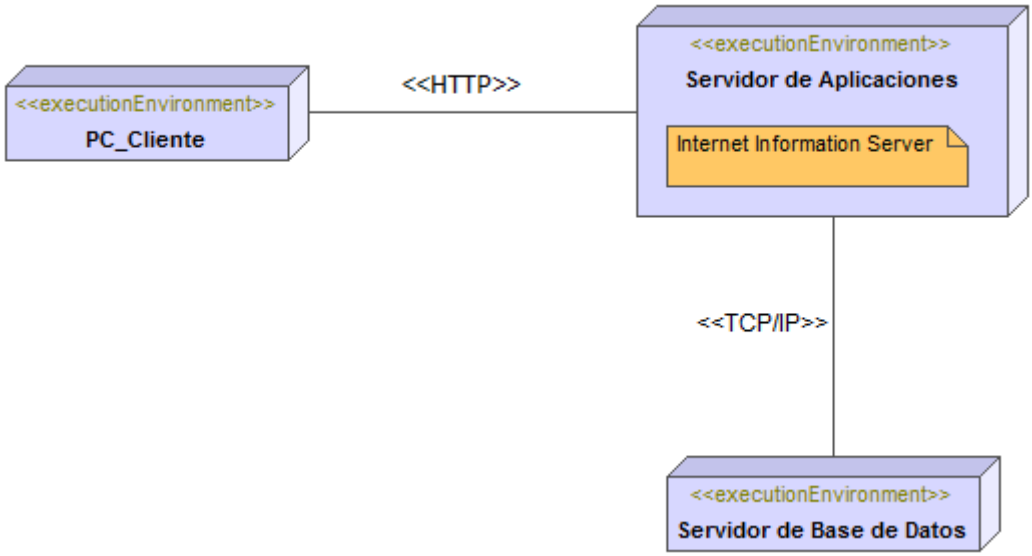


Figura 16. Modelo de despliegue

3.9 Validación

Cuando se construye un *software* la calidad no se puede probar, sino que se va construyendo a lo largo del desarrollo del mismo. Entre más errores se encuentren en las revisiones, más se podrá asegurar la calidad de la propuesta de solución.

En la investigación, al utilizar un proceso de desarrollo con enfoque ágil basado en la metodología XP se deben aplicar validaciones a las historias de usuario y realizar pruebas durante el ciclo de vida del producto. A continuación se especifican la estrategia, nivel, métodos y técnicas utilizadas para validar la solución.

3.9.1 Estrategia de prueba

Según Pressman una estrategia de prueba del *software* integra los métodos de diseño de casos de prueba en una serie bien planeada de pasos que desembocará en la eficaz construcción del *software*. La estrategia proporciona un mapa que describe los pasos, además de cuánto esfuerzo, tiempo y recursos consumirán (Pressman, 2005).

Con la estrategia de prueba se pretende que el producto se pruebe desde la parte más pequeña del *software* hasta la más grande en forma de espiral usando los niveles, tipos, técnicas y herramientas.

Los niveles de prueba a utilizar son las pruebas de unidad y las pruebas de aceptación.

3.9.2 Pruebas de unidad

Las pruebas de unidad se centran en la verificación de los elementos más pequeños del *software* que se puedan probar. Estas se aplican a componentes representados en el modelo de implementación para verificar que se cubren los flujos de control y los flujos de datos y que funcionan como se esperaba. Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa que está siendo probada.

Cuando un programador tiene que modificar código que ha sido desarrollado por otra persona se corre un alto riesgo de que introduzca errores. Las pruebas unitarias reducen este riesgo dado que avisarán al instante si algo deja de funcionar. Para ello debe adquirirse el hábito de ejecutar todas las pruebas del sistema tras hacer un cambio (Robles, et al., 2002).

Método de caja negra:

Las pruebas de caja negra parten de los requisitos funcionales para diseñar pruebas que se aplican sobre el sistema sin necesidad de conocer como está construido por dentro. Estas permiten determinar si la función está siendo desempeñada correctamente por el sistema bajo prueba, aplicando sobre el mismo un conjunto de datos de entrada y observando las salidas que se producen. Estas pruebas permiten encontrar funciones incorrectas o ausentes, errores de interfaz y errores en estructuras de datos o en accesos a las base de datos externas (Pressman, 2005).

En la propuesta de solución se aplicó el método de caja negra con la técnica de casos de prueba en donde se intenta demoler el sistema con todas las posibles combinaciones de datos y su resultado radica en tener una alta probabilidad de mostrar un error no descubierto hasta el momento.

Para la aplicación de esta técnica se generaron los artefactos de "Diseño de casos de prueba basados en HU". Por cada historia de usuario se realizó un documento en donde se recogen todos los datos necesarios para probar la interfaz.

En la siguiente tabla se muestra un ejemplo de diseño de casos de prueba, específicamente de la HU "Gestionar consulta". El resto se pueden encontrar en el Anexo 6.

Secciones a probar en la Historia de Usuario:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Adicionar consulta a concepto.	EC 1.1: Adicionar consulta a concepto.	Adiciona una consulta a un concepto.
	EC 1.2: Error en los datos.	Muestra mensaje de error.
	EC 1.3: Campos vacíos.	Muestra mensaje de error.
SC 2: Modificar consulta a concepto.	EC 2.1: Modificar consulta a concepto.	Modifica una consulta a un concepto.
	EC 2.2: Error en los datos.	Muestra mensaje de error.
	EC 2.3: Campos vacíos.	Muestra mensaje de error.
SC 3: Adicionar columna a relación.	EC 3.1: Adicionar columna a relación.	Adiciona una columna a una relación.
	EC 3.2: Error en los datos.	Muestra mensaje de error.
	EC 3.3: Campos vacíos.	Muestra mensaje de error.

Tabla 13. Caso de prueba: HU04

Descripción de variables:

No.	Nombre de campo	Clasificación	Valor nulo	Descripción
[1]	Consulta.	Campo de texto.	No.	Admite los caracteres a-z, 0-9, *, [], (), operadores, funciones.
[2]	Columna	Campo de texto.	No.	Admite letras solamente.

Tabla 14. Descripción de variables

Matriz de datos:

SC 1: Adicionar consulta a concepto

Escenario	Consulta	Columna	Respuesta del sistema	Resultado	Flujo central
EC 1.1: Adicionar consulta a concepto.	V/Persona	N/A	La consulta será adicionada.	Satisfactorio.	1. Seleccionar la opción de descripción 2. Doble clic sobre el concepto 3. Introducir la consulta. 4. Seleccionar la opción “Crear consulta”.
EC 1.2: Error en los datos.	I/Consulta	N/A	Mostrará un mensaje de error indicando que la consulta es inválida.	Satisfactorio.	
EC 1.3: Campos vacíos.	Vacío/	N/A	Mostrará un mensaje de error indicando que la consulta no debe estar vacía.	Satisfactorio.	

Tabla 15. Matriz de datos: SC Adicionar consulta

SC 2: Modificar consulta a concepto

Escenario	Consulta	Columna	Respuesta del sistema	Resultado	Flujo central
EC 2.1: Modificar consulta a concepto.	V/Persona	N/A	La consulta será modificada.	Satisfactorio.	1. Seleccionar la opción de descripción. 2. Doble clic sobre el concepto. 3. Seleccionar la opción “Ver consultas”.
EC 2.2: Error en los datos.	I/Consulta	N/A	Mostrará un mensaje de error indicando que la consulta no es válida.	Satisfactorio.	

EC 2.3: Campos vacíos.	Vacía/	N/A	Mostrará un mensaje de error indicando que la consulta no debe estar vacía.	Satisfactorio.	4. El sistema muestra las consultas creadas. 5. Seleccionar de la consulta que se desea modificar la opción “Modificar”. 6. Introducir los datos a modificar en la consulta. 7. Seleccionar la opción “Crear consulta”.

Tabla 16. Matriz de datos: SC Modificar consulta

SC 3: Adicionar columna a relación

Escenario	Consulta	Columna	Respuesta del sistema	Resultado	Flujo central
EC 3.1: Adicionar columna a relación.	N/A	V/nombre	La columna será adicionada.	Satisfactorio.	1. Seleccionar la opción de descripción. 2. Doble clic sobre la relación. 3. Introducir el nombre de la columna 4. Seleccionar la opción “Crear”.
EC 3.2: Error en los datos	N/A	I/1er nombre	Mostrará un mensaje de error indicando que el nombre de la columna es inválido.	Satisfactorio.	
EC 3.3: Campos vacíos.	N/A	Vacío/	Mostrará un mensaje de error indicando que el campo no puede estar vacío.	Satisfactorio.	

Tabla 17.. Matriz de datos: SC Adicionar columna

Registro de no conformidades:

Elemento	No.	NC	Aspecto corresp.	Etapa de detección	Significativa	Estado NC

Columna	1	El campo columna al adicionar la descripción de la relación dice que no puede ser nulo sin embargo no muestra mensaje de error.	Al adicionar las columnas para la descripción de la relación.	Iteración 3.	Validación.	RA

Tabla 18. Registro de no conformidades

Método de caja blanca:

Las pruebas de caja blanca conocidas también como pruebas de caja de cristal o pruebas estructurales se centran en los detalles procedimentales del *software*, por lo que su diseño está de gran manera ligado al código fuente. Para la realización de este tipo de pruebas, el probador escoge diferentes valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados. Aunque las pruebas de caja blanca son aplicables a varios niveles como son unidad, integración y sistema, habitualmente se aplican a las unidades de *software* (Laurie, 2006).

Las pruebas de caja blanca realizadas al sistema fueron automatizadas gracias a la herramienta Visual Studio. A continuación se muestra un ejemplo de la prueba unitaria realizada a los métodos de añadir y actualizar concepto y de añadir campos a un concepto, las demás se pueden encontrar en el Anexo 7.

```
public void AddConceptTest()
{
    var target = new ConceptOntology();
    const string name = "Persona";
    target.AddConcept(name);
}
```

Figura 17. Prueba unitaria al método AddConcept

```
public void UpdateConceptTest()
{
    var target = new ConceptOntology();
    const string oldName = "Persona";
    const string newName = "Casa";
    const string description = "Es una casa";
    const string label = "CISED";
    target.UpdateConcept(oldName, newName, description, label);
}
```

Figura 18. Prueba unitaria al método UpdateConcept

```
public void AddFieldsTest()
{
    var target = new ConceptOntology();
    const string name = "Persona";
    const string description = "Es una persona";
    const string fieldName = "nombre";
    const DataType fieldType = DataType.String;
    const bool isProtected = false;
    target.AddFields(name, description, fieldName, fieldType, isProtected);
}
```

Figura 19. Prueba unitaria al método AddFields

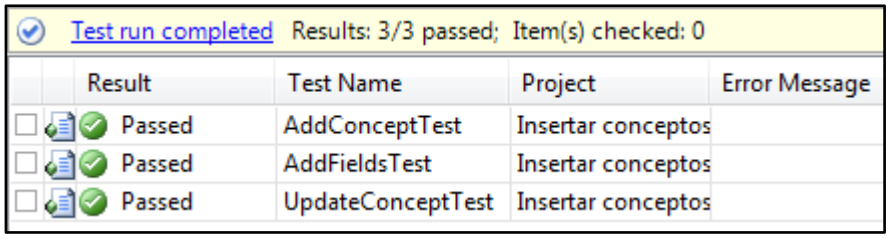


Figura 20. Resultado de la prueba unitaria

3.9.3 Pruebas de aceptación

Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a testear cuando una historia de usuario ha sido correctamente implementada. Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema (Fernández, 2002).

A continuación se muestra la prueba de aceptación para la Historia de Usuario Gestionar concepto, las demás se encuentran en el Anexo 8.

Caso de prueba de Aceptación	
Código Caso de Prueba: HU2-P1	**Nombre de Historia de Usuario:** Gestionar concepto
Nombre de la persona que realiza la prueba: Yisel Correa Rodríguez	

Descripción de la Prueba: Prueba de funcionalidad Adicionar concepto.
Condiciones de Ejecución: El usuario debe estar autenticado y con los permisos pertinentes. El usuario debe haber accedido a la página del Editor de Contextos.
Entradas/Pasos de Ejecución: 1. Seleccionar la opción “Adicionar concepto”. 2. Dar clic sobre el área de trabajo para adicionar le concepto.
Resultado Esperado: El concepto debe ser adicionado correctamente.
Evaluación de la Prueba: Satisfactoria

Tabla 19. Prueba de aceptación: HU2-P1 Adicionar concepto

Caso de prueba de Aceptación	
Código Caso de Prueba: HU2-P2	**Nombre de Historia de Usuario:** Gestionar concepto
Nombre de la persona que realiza la prueba: Yisel Correa Rodríguez	
Descripción de la Prueba: Prueba de funcionalidad Modificar concepto	
Condiciones de Ejecución: El concepto a modificar debe existir y haber sido seleccionado.	
Entradas/Pasos de Ejecución: 3. Doble clic sobre el concepto. 4. Introducir los campos (nombre del concepto, descripción del concepto, nombre del campo, descripción del campo, tipo de dato del campo, etiquetas). 5. Seleccionar la opción “Aceptar”.	
Resultado Esperado: Los campos del concepto deben ser modificados correctamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 20. Prueba de aceptación: HU2-P2 Modificar concepto

Caso de prueba de Aceptación	
Código Caso de Prueba: HU2-P3	**Nombre de Historia de Usuario:** Gestionar concepto

Nombre de la persona que realiza la prueba: Yisel Correa Rodríguez
Descripción de la Prueba: Prueba de funcionalidad Eliminar concepto
Condiciones de Ejecución: El concepto a eliminar debe existir.
Entradas/Pasos de Ejecución: 1. Seleccionar la opción “Eliminar concepto”. 2. Dar clic sobre el concepto que se desea eliminar.
Resultado Esperado: El concepto debe ser eliminado correctamente.
Evaluación de la Prueba: Satisfactoria

Tabla 21. Prueba de aceptación: HU2-P3 Eliminar concepto

Caso de prueba de Aceptación	
Código Caso de Prueba: HU2-P4	**Nombre de Historia de Usuario:** Gestionar concepto
Nombre de la persona que realiza la prueba: Yisel Correa Rodríguez	
Descripción de la Prueba: Prueba de funcionalidad Buscar concepto	
Condiciones de Ejecución: El concepto a buscar debe existir.	
Entradas/Pasos de Ejecución: 1. Seleccionar la opción “Buscar”. 2. Dar doble clic sobre el nombre del concepto que se desea buscar dentro del área de trabajo.	
Resultado Esperado: El concepto debe ser localizado dentro del área de trabajo correctamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 22. Prueba de aceptación: HU2-P4 Buscar concepto

3.10 Resultados de las pruebas unitarias

En las cinco iteraciones de desarrollo se realizaron un total de cinco iteraciones de pruebas en las cuales se encontraron y corrigieron errores llegando a un resultado positivo en todas las funcionalidades. El resultado de las pruebas se muestra a continuación.

Iteraciones	1	2	3	4	5
HU no implementadas	9	7	5	1	0
HU que pasaron	3	3	10	11	12
HU que no pasaron	0	2	1	1	0

Tabla 23. Pruebas a HU por cada iteración

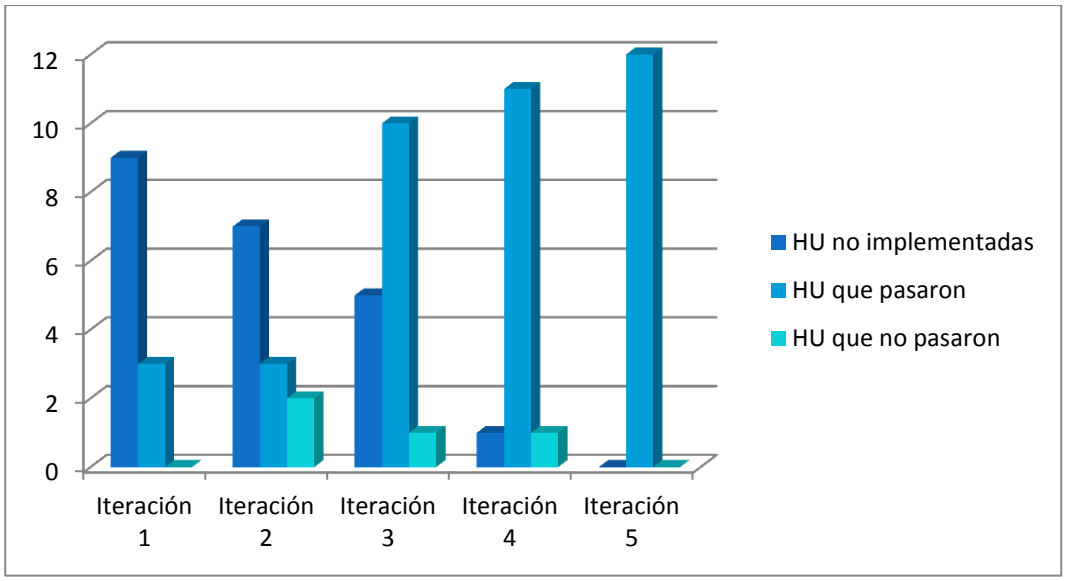


Figura 21. Gráfica de prueba a HU por cada iteración

Luego de realizar un análisis estadístico de los resultados de las pruebas en cada iteración se concluye que la solución desarrollada cumple con todas las funcionalidades y requerimientos definidos, teniendo resultados correctos.

3.11 Beneficios de la aplicación

- ✓ Versión estable de la aplicación web para la representación de datos basada en mapas conceptuales.

- ✓ Editor de contextos gráfico para la gestión de los mismos.
- ✓ Facilidad en cuanto al acceso y manipulación de la información contenida en bases de datos.
- ✓ Integración de los procesos de edición de contextos y navegación en un solo sistema.

Conclusiones

En el presente capítulo como parte de los elementos definidos en la metodología se encuentran las Tarjetas CRC que permiten que todo el equipo contribuya a la tarea del diseño y que representan las clases que se deben desarrollar con sus respectivas responsabilidades y colaboradores. Se describe la arquitectura, definida en un modelo de N-capas dividida en cuatro niveles. Para cada capa se realizó una descripción y se explicó la relación entre las mismas. Se utilizaron los patrones de diseño GRASP y GOF los cuales permitieron reutilizar las experiencias de los desarrolladores. También se definieron los estándares de diseño y codificación lo que facilitó una guía a los desarrolladores para la implementación de la solución. Algunos estándares de diseño definidos fueron las pautas cromáticas, las acciones, la interacción y los componentes. Los estándares de codificación comprenden los comentarios, el nombre de identificadores y la declaración de clases e interfaces. Luego de la realización de las pruebas unitarias se comprobó el correcto funcionamiento de procedimientos internos de la solución. Con el uso de las pruebas de aceptación se validó que las funciones a nivel de interfaces de usuario fueron operativas.

Conclusiones

Como conclusiones generales de la investigación se tiene que:

- ✓ Luego de realizar un estudio de las soluciones existentes para la creación de mapas conceptuales y de la herramienta de referencia, se logró adquirir mayor visión y entendimiento en cuanto a los procesos de navegación y edición de contextos.
- ✓ Las herramientas establecidas resultaron idóneas para la realización de la solución.
- ✓ La arquitectura definida logró un excelente nivel de organización y relación entre las capas especificadas.
- ✓ Las pruebas realizadas permitieron comprobar las funcionalidades del sistema, brindando resultados satisfactorios que otorgaron validez a la investigación.
- ✓ Se obtuvo una versión estable de la aplicación web para la representación de datos basada en mapas conceptuales y un editor de contextos gráfico para la gestión de los mismos.
- ✓ La solución web brinda mayor facilidad en cuanto al acceso y manipulación de la información contenida en bases de datos.
- ✓ Se logró la integración de los procesos de edición de contextos y navegación en un solo sistema, lo cual mejora la usabilidad del “Navegador Conceptual”.

Recomendaciones

Se recomienda:

- ✓ Hacer un módulo de administración para gestionar los proyectos elaborados por los usuarios.
- ✓ Realizar los contextos asociados a la base de datos del proyecto Identidad Cuba de manera que se pueda aplicar la solución web desarrollada en dicho proyecto.

Bibliografía

Altova. 2012. Altova. [En línea] 2012. [Citado el: 10 de Enero de 2012.] http://www.altova.com/xml-editor/.

—. 2012. Altova. [En línea] 2012. [Citado el: 10 de Enero de 2012.] http://www.altova.com/xmlspy.html.

—. 2012. Altova. [En línea] 2012. [Citado el: 10 de Enero de 2012.] http://www.altova.com/umodel.html.

Alvarez, Miguel Angel. 2001. *DesarrolloWeb.com.* [En línea] 13 de Junio de 2001. [Citado el: 1 de Diciembre de 2011.] http://www.desarrolloweb.com/articulos/449.php.

Ávila, Carlos Andrés Dorado. 2006. *CmapTools, programa para elaborar mapas conceptuales.* [En línea] Eduteka, 5 de Agosto de 2006. [Citado el: 6 de Diciembre de 2011.] http://www.eduteka.org/Cmap1.php.

basesdedatos.org. 2010. basesdedatos.org. [En línea] 2010. [Citado el: 5 de Diciembre de 2011.] http://www.basesdedatos.org/.

Booch, Grady, Rumbaugh, James y Jacobson, Ivar. 2005. *El Lenguaje Unificado de Modelado. Addison-Wesley Object Technology Series.* 2005.

Díaz Pastrana, Dainerys y Pérez Pérez, Joisel. 2008. *Modelo para la ayuda en la toma de decisiones relativas.* La Habana : s.n., 2008.

Dürsteler, Juan C. 2004. [En línea] 1 de Marzo de 2004. [Citado el: 5 de Diciembre de 2011.] http://www.infovis.net/printMag.php?num=141&lang=1.

Echeverry Tobón, Luis Miguel y Delgado Carmona, Luz Elena. 2007. *Caso práctico de la metodología ágil XP al desarrollo de software.* 2007.

Fernández Medina, Eduardo. 2007. Ingeniería de Software I. [En línea] 2007. [Citado el: 20 de Diciembre de 2011.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/.

Fernández, Escribano Gerardo. 2002. *Introducción a Extreme Programming.* 2002.

Gamma, Erich, y otros. 2005. *Design Patterns. Elements of Reusable Object-Oriented Software.* 2005.

García, Joaquín. 2005. IngenieroSoftware. [En línea] 27 de Mayo de 2005. [Citado el: 10 de Marzo de 2012.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

García, MsC. Jorge Landrian. 2008. *"Navegador Conceptual", herramienta para el acceso a la información basada en mapas conceptuales.* Caracas : s.n., 2008.

Herranz, Raúl. 2010. Utópica informática. [En línea] 23 de Diciembre de 2010. [Citado el: 03 de Marzo de 2012.] http://www.utopicainformatica.com/2010/12/convencionesestandares-de-codificacion.html.

IEEE 1471-2000.

IEEE, Standars Coordinating Committee of the Computer Society of the. 1990. *IEEE Standard Glossary of Software Engineering Terminology.* 1990.

Jabconson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El proceso unificado de desarrollo del software.* Madrid : Adison Wesley, 2000. ISBN: 84-7829-036-2.

jQuery Project. 2010. The jQuery Project. [En línea] 2010. [Citado el: 15 de Diciembre de 2011.] http://www.jqueryui.com/.

jQuery UI Team. 2012. [En línea] 2012. [Citado el: 10 de Enero de 2012.] http://jqueryui.com/.

Larman, Craig. 2004. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* La Habana : Félix Varela, 2004.

—. 2004. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* La Habana : Felix Varela, 2004. Vol. I.

Laurie, W. 2006. *White-Box Testing.* 2006.

López, Alberto Gómez. 2007. Descripción de la ontología. [En línea] 2007. [Citado el: 6 de Diciembre de 2011.] http://axel.deri.ie/teaching/ri2007/alberto.pdf.

maestrosdelweb. 2010. *El HTML, una idea en evolución.* [En línea] 2010. [Citado el: 4 de Diciembre de 2011.] http://www.maestrosdelweb.com/editorial/htmlhis/.

Microsoft. 2011. [En línea] 2011. [Citado el: 1 de Diciembre de 2011.] http://www.microsoft.com/spain/visualstudio.

—. 2005. Microsoft. [En línea] Noviembre de 2005. [Citado el: 9 de Diciembre de 2011.] http://msdn.microsoft.com/es-es/library/z1zx9t92%28v=VS.80%29.aspx.

—. 2006. Microsoft. [En línea] Julio de 2006. [Citado el: 15 de Diciembre de 2011.] www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf.

—. 2012. Microsoft. [En línea] 2012. [Citado el: 15 de Diciembre de 2011.] http://www.microsoft.com/visualstudio/es-es.

—. 2012. Microsoft. [En línea] 2012. [Citado el: 5 de Enero de 2012.] http://msdn.microsoft.com/es-ES/library/dd547188.aspx.

—. 2012. Microsoft. [En línea] 2012. [Citado el: 10 de Enero de 2015.] http://msdn.microsoft.com/es-es/library/gg589525%28v=vs.85%29.aspx.

—. 2005. MSF for CMMI Process Improvement. [En línea] 2005. [Citado el: 5 de Diciembre de 2011.] http://guides.brucejmack.biz/MSF%20for%20CMMI%20Process%20Improvement/Process%20Guidanc.

Novak, Joseph D. 2006. The Theory Underlying Concept Maps and How to Construct and Use Them. [En línea] 2006. [Citado el: 22 de Noviembre de 2011.] www.stanford.edu/dept/SUSE/projects/ireport/articles/concept_maps/The%20Theory%20Underlying%20Concept%20Maps.pdf.

Ojeda Cabrera, Lic. Angela, y otros. 2007. [En línea] ACIMED, 2007. [Citado el: 6 de Diciembre de 2011.] http://bvs.sld.cu/revistas/aci/vol15_05_07/aci09507.htm.

Pelaez , Juan Carlos. 2009. Geeks.ms. [En línea] 29 de Mayo de 2009. [Citado el: 15 de Diciembre de 2011.] http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx.

Peñalver, G., Meneses, A. y S., García. 2010. *SXP, Metodología ágil para el desarrollo de software.* Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba : s.n., 2010.

Pérez, Javier Eguíluz. 2010. *Introducción a CSS.* [En línea] 2010. [Citado el: 1 de Diciembre de 2011.] http://www.librosweb.es/css/capitulo1.html.

—. 2010. *Introducción a JavaScript.* [En línea] 2010. [Citado el: 1 de Diciembre de 2011.] http://www.librosweb.es/javascript/capitulo1.html.

Pressman, Roger S. 2005. *La Ingeniería de Software, un enfoque práctico.* 2005.

Real Academia Española. 2010. Real Academia Española. *Real Academia Española.* [En línea] 2010. [Citado el: 12 de Enero de 2012.] http://buscon.rae.es/draeI/.

Robles, Gregorio y Ferrer, Jorge. 2002. *Programación eXtrema y Software Libre.* 2002.

Rovira, Cristòfol. 2005. *El editor de mapas conceptuales DigiDocMap y la norma Topic Maps.* [En línea] 2005. [Citado el: 4 de Diciembre de 2011.] http://www.upf.edu/hipertextnet/numero-3/digidocmap.html.

Sanz, Ismael y Jiménez Ruiz, Ernesto. 2007. *Ontologías en Informática.* [En línea] Mayo de 2007. [Citado el: 6 de Diciembre de 2011.] http://krono.act.uji.es/publications/techrep/Book-Chapter-Protege-report2007.pdf.

Simón, Alfredo, y otros. 2006. *GECOSOFT: Un Entorno Colaborativo para la Gestón del Conocimiento con Mapas Conceptuales.* La Habana : s.n., 2006.

Villate, Jaime E. 2001. *Introducción al XML.* [En línea] Universidad de Oporto, 5 de Mayo de 2001. [Citado el: 6 de Diciembre de 2011.] http://fisica.fe.up.pt/~villate/documentos/cursoXML.pdf.

W3C. 2012. W3C. [En línea] 24 de Enero de 2012. [Citado el: 25 de Enero de 2012.] http://www.w3.org/MarkUp/.

w3schools.com. 2012. w3schools.com. [En línea] 2012. [Citado el: 15 de Enero de 2012.] http://www.w3schools.com/ajax/ajax_intro.asp.

Anexos

Anexo 1. Historias de usuario

Historia de Usuario	
Código: HU01	**Nombre Historia de Usuario:** Autenticar usuario.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: RF1.	
Prioridad: Baja.	**Iteración Asignada:** 5
Riesgo en Desarrollo: Medio.	**Puntos Estimados:** 1 semana.
Asignado a: Yisel Correa Rodríguez.	**Puntos Reales:** 1 semana.
Descripción: El sistema debe brindar la posibilidad al usuario de autenticarse para acceder a las opciones del sistema que le corresponden según su nivel de acceso. Se mostrará un formulario de autenticación en la página principal para que el usuario se autentifique. Según el rol que posea el usuario, podrá acceder o no a los recursos de la aplicación.	
Observaciones: Si el usuario o la contraseña no son correctos, se mostrará un mensaje de error.	

Tabla 24. Historia de usuario: Autenticar usuario

Historia de Usuario	
Código: HU03	**Nombre Historia de Usuario:** Gestionar relación.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: RF3.	
Prioridad: Media.	**Iteración Asignada:** 2
Riesgo en Desarrollo: Medio.	**Puntos Estimados:** 3 semanas.
Asignado a: Yoslandy López Cristóbal.	**Puntos Reales:** 3 semanas.
Descripción: Al seleccionar la opción “Adicionar relación” el sistema debe brindar la posibilidad de	

adicionar una relación (dueño, relacionado, descriptor, descripción, etiquetas y restricciones) entre dos conceptos contenidos dentro del área de trabajo. Al dar doble clic sobre la relación, el sistema debe permitir modificar su información. Al seleccionar la opción “Eliminar relación” y dar clic sobre una relación el sistema debe eliminarla.
Observaciones: Una relación sólo puede tener dos restricciones. El listado de etiquetas que se pueden añadir a la relación contiene todas las creadas para los conceptos. Para cada par de conceptos sólo puede existir una relación.

Tabla 25. Historia de usuario: Gestionar relación

Historia de Usuario	
Código: HU04	**Nombre Historia de Usuario:** Gestionar consulta.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: RF4.	
Prioridad: Media.	**Iteración Asignada:** 3
Riesgo en Desarrollo: Medio.	**Puntos Estimados:** 2 semanas.
Asignado a: Yoslandy López Cristóbal.	**Puntos Reales:** 2 semanas.
Descripción: En el caso de los conceptos el sistema debe brindar la posibilidad de adicionar una representación de la consulta de la base de datos y validar que sea correcta. La misma debe poder ser modificada y eliminada. Para las relaciones se debe tener en cuenta si son simples, es decir, si corresponde a una relación de 1-N en una base de datos o si tienen una tabla intermedia (N-M), estas últimas son extensiones de las primeras. Si la relación es simple se especifican los campos LeftTableColumns y RightTableColumns y si es de N-M se incluyen además de los anteriores los campos InnerTableLeftColumns, InnerTableRightColumns e InnerTableSQL. El sistema debe permitir modificar estos campos y eliminarlos.	

Observaciones: Debe seleccionar el concepto o la relación a la cual se le desea hacer la consulta y haber seleccionado en el menú la opción de descripción. Es necesario que se haya establecido la conexión con la base de datos anteriormente.

Tabla 26. Historia de usuario: Gestionar consulta

Historia de Usuario	
Código: HU05	**Nombre Historia de Usuario:** Gestionar operador.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: RF5.	
Prioridad: Media.	**Iteración Asignada:** 3
Riesgo en Desarrollo: Medio.	**Puntos Estimados:** 2 semanas.
Asignado a: Yoslandy López Cristóbal.	**Puntos Reales:** 2 semanas.
Descripción: El sistema debe brindar la posibilidad de adicionar un operador (operación, tipo de operación, cantidad de miembros, precedencia, tipo de retorno, formato de traducción y las expresiones). Dicho operador debe poder ser modificado y eliminado.	
Observaciones:	

Tabla 27. Historia de usuario: Gestionar operador

Historia de Usuario	
Código: HU06	**Nombre Historia de Usuario:** Guardar proyecto.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: RF6.	
Prioridad: Media.	**Iteración Asignada:** 4
Riesgo en Desarrollo: Medio.	**Puntos Estimados:** 1 semana.

Asignado a: Yisel Correa Rodríguez.	**Puntos Reales:** 1 semana.
Descripción: El sistema debe permitir guardar el proyecto (se debe especificar el nombre) en la carpeta personal del usuario cuando el mismo lo desee. El proyecto consiste en los ficheros XML de ontología, sintaxis, descripción y el fichero de configuración.	
Observaciones: Esta opción no validará los ficheros XML pues se asume que no se han terminado.	

Tabla 28. Historia de usuario: Guardar proyecto

Historia de Usuario	
Código: HU07	**Nombre Historia de Usuario:** Cargar proyecto.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: FR7.	
Prioridad: Media.	**Iteración Asignada:** 4
Riesgo en Desarrollo: Medio.	**Puntos Estimados:** 2 semanas.
Asignado a: Yisel Correa Rodríguez.	**Puntos Reales:** 2 semanas.
Descripción: El sistema debe brindar al usuario la posibilidad de cargar un proyecto existente en su carpeta desde la página del “Editor”. Para eso se mostrarán cuatro campos de entrada (cada uno perteneciente a un fichero determinado) y cuatro botones de búsqueda para ubicar el fichero en el campo de entrada que le corresponde. Si se selecciona la opción cargar desde la página del “Navegador” el sistema debe permitir cargar los ficheros publicados.	
Observaciones: En el caso del “Editor” los archivos cargados deben pertenecer al mismo proyecto y la dirección de cada fichero debe coincidir con el campo de entrada en el que se encuentre.	

Tabla 29. Historia de usuario: Cargar proyecto

Historia de Usuario	
Código: HU08	**Nombre Historia de Usuario:** Generar ficheros XML.
Modificación de Historia de Usuario Número: ninguna.	

Referencia: RF8.	
Prioridad: Alta.	**Iteración Asignada:** 1
Riesgo en Desarrollo: Alto.	**Puntos Estimados:** 1 semana.
Asignado a: Yisel Correa Rodríguez.	**Puntos Reales:** 1 semana.
Descripción: El sistema debe permitir generar los ficheros XML de ontología, descripción y sintaxis como parte de un proyecto al cual el usuario debe especificarle un nombre. Debe dar la posibilidad de compartir esos ficheros, es decir, guardarlos en una carpeta de acceso público para que todos los usuarios puedan navegar haciendo uso de esos contextos.	
Observaciones: Los ficheros deben ser validados a partir de sus respectivos xsd y si son correctos se crean. Para que el fichero sintaxis y descripción sean correctos es necesario especificar su definición.	

Tabla 30. Historia de usuario: Generar ficheros XML

Historia de Usuario	
Código: HU09	**Nombre Historia de Usuario:** Establecer conexión.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: RF9.	
Prioridad: Media.	**Iteración Asignada:** 4
Riesgo en Desarrollo: Medio.	**Puntos Estimados:** 3 semanas.
Asignado a: Yisel Correa Rodríguez.	**Puntos reales:** 3 semanas.
Descripción: El sistema debe permitir a los usuarios establecer los parámetros para conectarse a la base de datos por la cual desean navegar y especificar los contextos con los cuales van a realizar la navegación. Los contextos se encuentran en las carpetas de proyectos del usuario autenticado y en la carpeta de proyectos publicados.	

Observaciones: El sistema debe informar si hubo algún fallo en la conexión o si los ficheros XML son inválidos.

Tabla 31. Historia de usuario: Establecer conexión

Historia de Usuario	
Código: HU10	**Nombre Historia de Usuario:** Realizar consulta.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: RF10.	
Prioridad: Alta.	**Iteración Asignada:** 1
Riesgo en Desarrollo: Alto.	**Puntos Estimados:** 3 semanas.
Asignado a: Yoslandy López Cristóbal.	**Puntos reales:** 3 semanas.
Descripción: El sistema debe permitir al usuario ingresar una consulta en un lenguaje comprensible para cualquier persona que no tenga muchos conocimiento de SQL ni del negocio. La consulta se debe crear a partir de un componente visual que permita seleccionar los conceptos, relaciones, funciones, operadores y demás elementos contenidos en los ficheros XML. Debe mostrar el historial de consultas realizadas y permitir ejecutarlas nuevamente. Este historial puede ser eliminado. Las consultas deben poder ser realizadas también desde una consola. A partir de las consultas creadas en la consola se debe poder crear la equivalente en el componente visual para poder añadir nuevos filtros y condiciones.	
Observaciones: Al ejecutar la consulta debe validarse que la misma sea correcta, en caso de serlo debe mostrar el resultado.	

Tabla 32. Historia de usuario: Realizar consulta

Historia de Usuario	
Código: HU11	**Nombre Historia de Usuario:** Realizar navegación.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: RF11.	

Prioridad: Alta.	**Iteración Asignada:** 1
Riesgo en Desarrollo: Alto.	**Puntos Estimados:** 3 semanas.
Asignado a: Yoslandy López Cristóbal.	**Puntos reales:** 3 semanas.
Descripción: El sistema debe permitir a los usuarios poder navegar a través de la información contenida en la base de datos. A partir del resultado de una consulta realizada el usuario debe poder navegar por los conceptos y campos asociados al mismo y ver los resultados de dicha navegación en forma de árbol.	
Observaciones:	

Tabla 33. Historia de usuario: Realizar navegación

Historia de Usuario	
Código: HU12	**Nombre Historia de Usuario:** Gestionar investigación.
Modificación de Historia de Usuario Número: ninguna.	
Referencia: RF12.	
Prioridad: Media.	**Iteración Asignada:** 4
Riesgo en Desarrollo: Medio.	**Puntos Estimados:** 3 semanas.
Asignado a: Yoslandy López Cristóbal.	**Puntos reales:** 3 semanas.
Descripción: El sistema debe permitir al usuario adicionar un elemento a la investigación, es decir que debe crear dicho elemento como un nodo raíz del árbol y a partir de ese nodo continuar realizando la navegación. Este nodo debe poder ser eliminado.	
Observaciones:	

Tabla 34. Historia de usuario: Gestionar investigación

Anexo 2. Interfaces

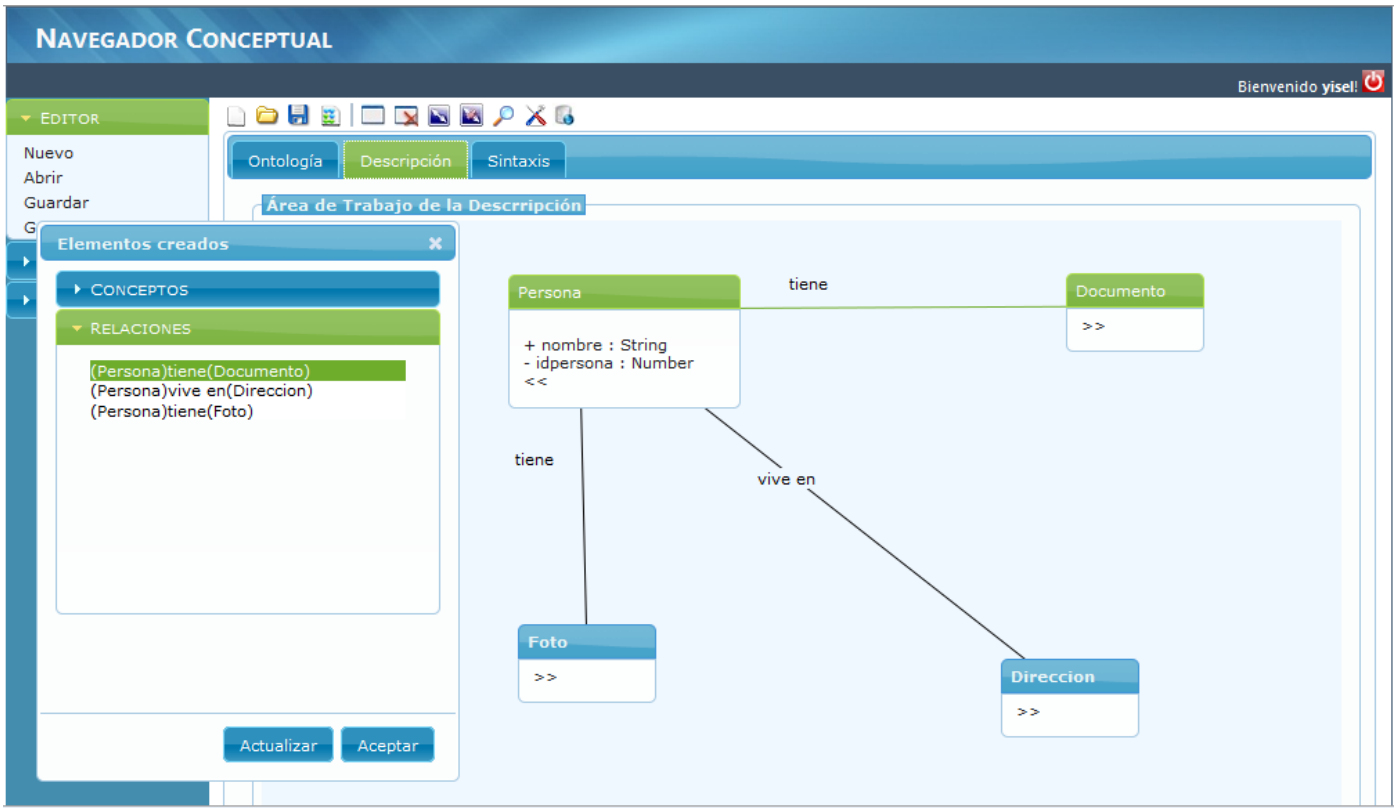


Figura 22. Interfaz del Editor

Figura 23. Interfaz HU2: Modificar concepto

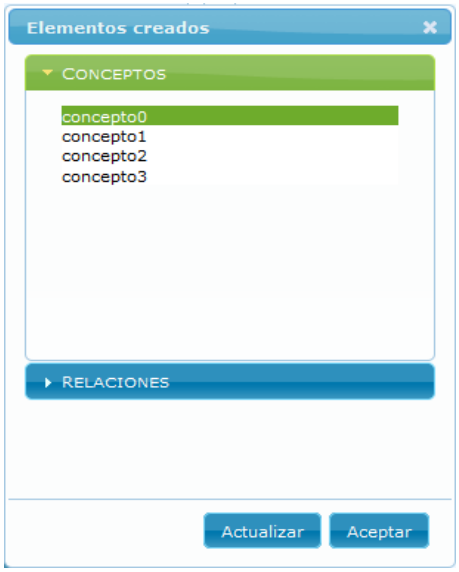


Figura 24. Interfaz HU2: Buscar concepto

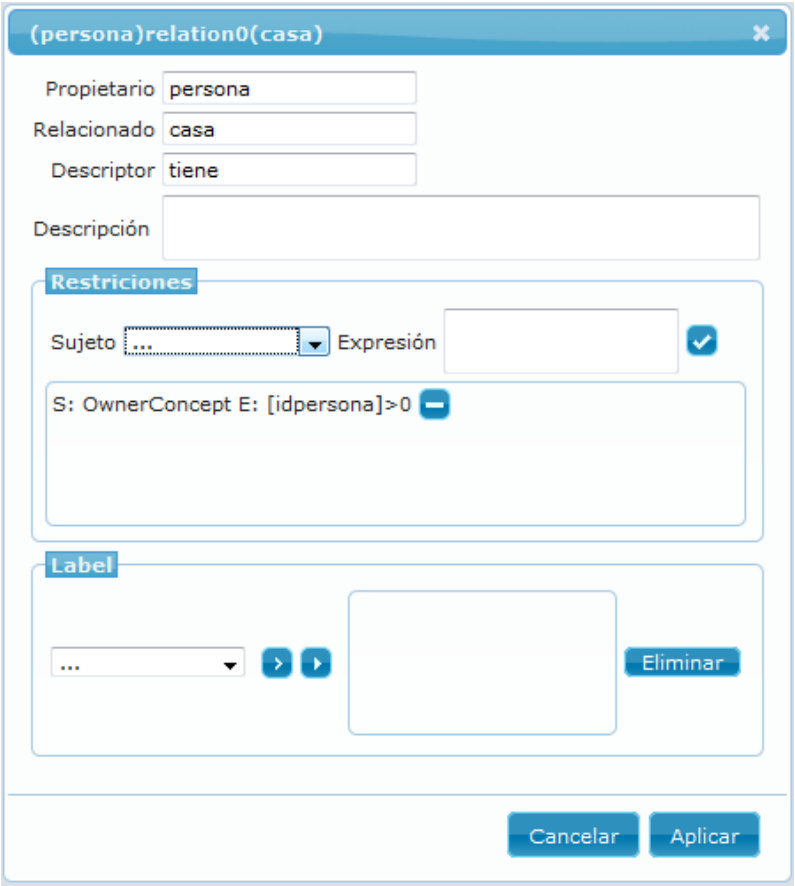


Figura 25. Interfaz HU3: Modificar relación

Figura 26. Interfaz HU9: Establecer conexión

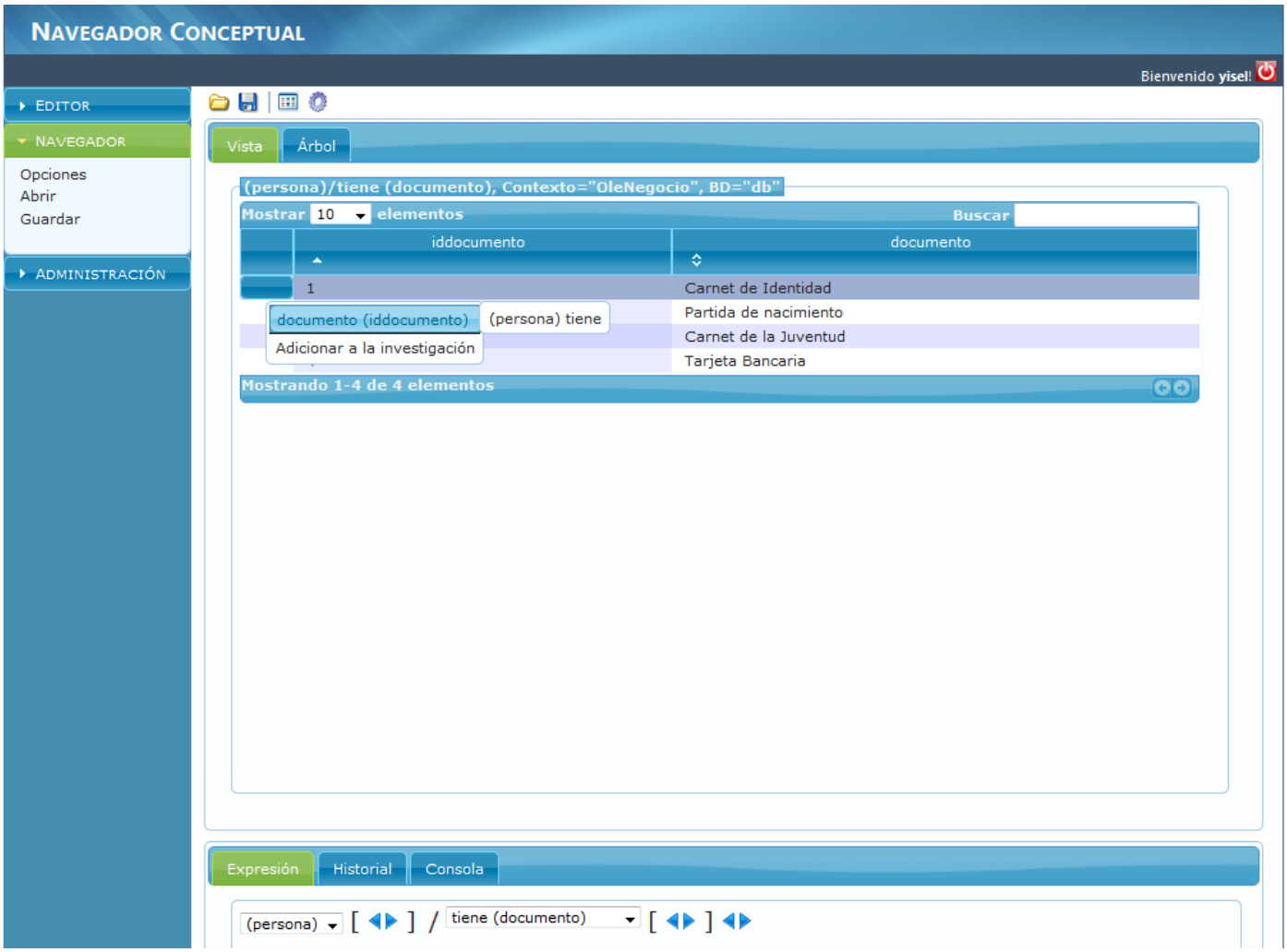


Figura 27. Interfaz del Navegador

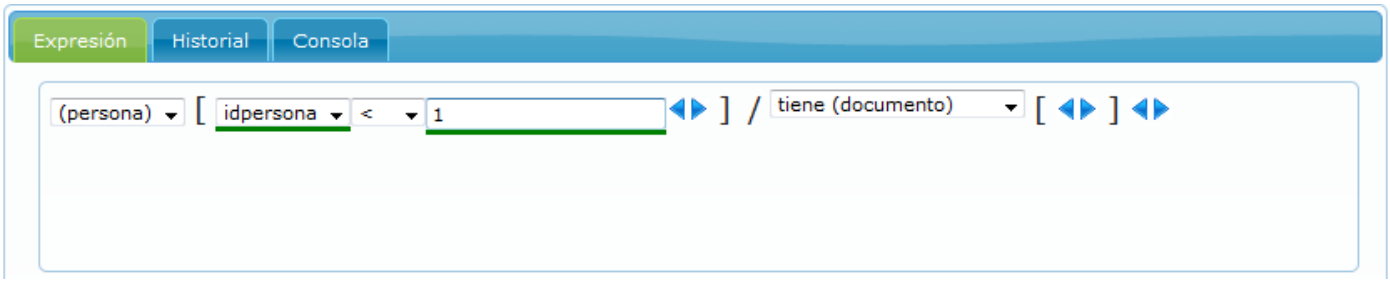


Figura 28. Interfaz HU10: Realizar consulta

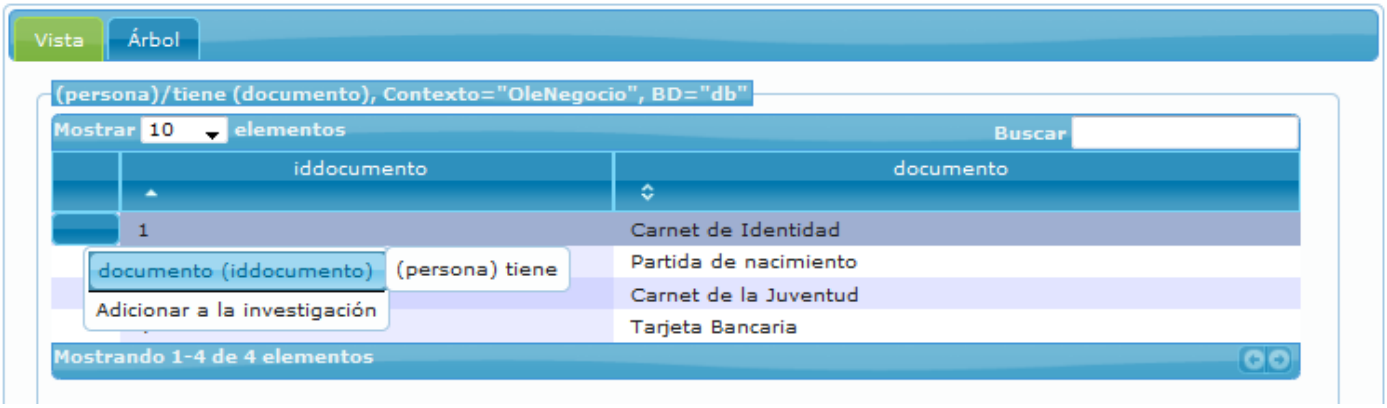


Figura 29. Interfaz HU11: Realizar navegación

Anexo 3. Tareas de ingeniería

Iteración 1:

Tarea de ingeniería	
Número Tarea: 1	**Número de Historia de Usuario:** 8
Nombre Tarea: Generar ficheros XML.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Enero.	**Fecha Fin:** Enero.
Programador Responsable: Yisel Correa Rodríguez.	
Descripción: El sistema guardará un proyecto con los ficheros XML de ontología, descripción y sintaxis generados a partir de la información contenida en la página. Estos serán validados antes de guardarse.	

Tabla 35. Tarea de ingeniería: Gestionar ficheros XML

Tarea de ingeniería	
Número Tarea: 2	**Número de Historia de Usuario:** 10
Nombre Tarea: Crear componente visual para realizar una consulta.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1

Fecha Inicio: Enero.	**Fecha Fin:** Enero.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: Crear el componente visual que permita introducir una consulta a partir de los conceptos, relaciones, operadores, funciones y demás elementos contenidos en los ficheros XML .	

Tabla 36. Tarea de ingeniería: Crear componente visual para realizar una consulta

Tarea de ingeniería	
Número Tarea: 3	**Número de Historia de Usuario:** 10
Nombre Tarea: Guardar historial de consultas realizadas.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Enero.	**Fecha Fin:** Enero.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: Crear un historial de las consultas creadas para que el usuario pueda ejecutarlas o modificarlas en otro momento. Las consultas guardadas en el historial pueden ser eliminadas.	

Tabla 37. Tarea de ingeniería: Guardar historial de consultas realizadas

Tarea de ingeniería	
Número Tarea: 4	**Número de Historia de Usuario:** 10
Nombre Tarea: Validar consulta.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Enero.	**Fecha Fin:** Enero.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe validar antes de crear la consulta SQL si la estructura de la misma en el componente visual es correcta.	

Tabla 38. Tarea de ingeniería: Validar consulta

Tarea de ingeniería	
Número Tarea: 5	**Número de Historia de Usuario:** 10
Nombre Tarea: Realizar consulta.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Enero.	**Fecha Fin:** Enero.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe crear la consulta SQL a partir de la introducida en el componente visual y mostrar la información solicitada.	

Tabla 39. Tarea de ingeniería: Realizar consulta

Tarea de ingeniería	
Número Tarea: 6	**Número de Historia de Usuario:** 11
Nombre Tarea: Realizar navegación.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Enero.	**Fecha Fin:** Enero.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe mostrar un menú para poder navegar por la información relacionada a un concepto o un campo y mostrar la misma en forma de árbol, donde el nodo raíz será el concepto o campo a partir del cual se inició la navegación.	

Tabla 40. Tarea de ingeniería: Realizar navegación

Iteración 2:

Tarea de ingeniería	
Número Tarea: 11	**Número de Historia de Usuario:** 3

Nombre Tarea: Adicionar relación.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Febrero.	**Fecha Fin:** Febrero.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe permitir adicionar una relación (dueño, relacionado, descriptor, descripción, restricciones, etiquetas) a la ontología.	

Tabla 41. Tarea de ingeniería: Adicionar relación

	Tarea de ingeniería
Número Tarea: 12	**Número de Historia de Usuario:** 3
Nombre Tarea: Modificar relación.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Febrero.	**Fecha Fin:** Febrero.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe permitir modificar los campos (dueño, relacionado, descriptor, descripción, restricciones, etiquetas) de una relación seleccionada.	

Tabla 42. Tarea de ingeniería: Modificar relación

	Tarea de ingeniería
Número Tarea: 13	**Número de Historia de Usuario:** 3
Nombre Tarea: Eliminar relación.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Febrero.	**Fecha Fin:** Febrero.

Programador Responsable: Yoslandy López Cristóbal.
Descripción: El sistema debe permitir eliminar una relación seleccionada.

Tabla 43. Tarea de ingeniería: Eliminar relación

Iteración 3:

	Tarea de ingeniería
Número Tarea: 14	**Número de Historia de Usuario:** 4
Nombre Tarea: Adicionar consulta a un concepto.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Marzo.	**Fecha Fin:** Marzo.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe permitir crear una consulta a un concepto específico y validar que la misma sea correcta	

Tabla 44. Tarea de ingeniería: Adicionar consulta a un concepto

	Tarea de ingeniería
Número Tarea: 15	**Número de Historia de Usuario:** 4
Nombre Tarea: Modificar consulta de un concepto.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Marzo.	**Fecha Fin:** Marzo.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe permitir modificar una consulta de un concepto específico y validar que la misma sea correcta	

Tabla 45. Tarea de ingeniería: Modificar consulta de un concepto

Tarea de ingeniería	
Número Tarea: 16	**Número de Historia de Usuario:** 4
Nombre Tarea: Eliminar consulta de un concepto.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Marzo.	**Fecha Fin:** Marzo.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe permitir eliminar una consulta de un concepto específico.	

Tabla 46. Tarea de ingeniería: Eliminar consulta de un concepto

Tarea de ingeniería	
Número Tarea: 17	**Número de Historia de Usuario:** 4
Nombre Tarea: Adicionar columnas a una relación	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Marzo.	**Fecha Fin:** Marzo.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe permitir adicionar las columnas para la descripción de las relaciones. Si es una relación 1-N los campos a llenar serían LeftTableColumns y RigthTableColumns y si es una relación de N-M se deben incluir además los campos InnerTableLeftColumns, InnerTableRightColumns e InnerTableSQL.	

Tabla 47. Tarea de ingeniería: Adicionar columnas a una relación

Tarea de ingeniería	
Número Tarea: 18	**Número de Historia de Usuario:** 4
Nombre Tarea: Eliminar columnas de una relación	

Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Marzo.	**Fecha Fin:** Marzo.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe permitir eliminar las columnas para la descripción de las relaciones.	

Tabla 48. Tarea de ingeniería: Eliminar columnas de una relación

		Tarea de ingeniería
Número Tarea: 19	**Número de Historia de Usuario:** 5	
Nombre Tarea: Adicionar operador.		
Tipo Tarea: Desarrollo.		**Puntos Estimados:** 1
Fecha Inicio: Marzo.		**Fecha Fin:** Marzo.
Programador Responsable: Yoslandy López Cristóbal.		
Descripción: El sistema debe permitir adicionar un operador (operación, tipo de operación, cantidad de miembros, precedencia, tipo de retorno, formato de traducción, y expresiones) a la sintaxis.		

Tabla 49. Tarea de ingeniería: Adicionar operador

		Tarea de ingeniería
Número Tarea: 20	**Número de Historia de Usuario:** 5	
Nombre Tarea: Modificar operador.		
Tipo Tarea: Desarrollo.		**Puntos Estimados:** 1
Fecha Inicio: Marzo.		**Fecha Fin:** Marzo.
Programador Responsable: Yoslandy López Cristóbal.		

Descripción: El sistema debe permitir modificar los campos (operación, tipo de operación, cantidad de miembros, precedencia, tipo de retorno, formato de traducción, y expresiones) de un operador seleccionado.

Tabla 50. Tarea de ingeniería: Modificar operador

Tarea de ingeniería	
Número Tarea: 21	**Número de Historia de Usuario:** 5
Nombre Tarea: Eliminar operador.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Marzo.	**Fecha Fin:** Marzo.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe permitir eliminar un operador seleccionado.	

Tabla 51. Tarea de ingeniería: Eliminar operador

Iteración 4:

Tarea de ingeniería	
Número Tarea: 22	**Número de Historia de Usuario:** 9
Nombre Tarea: Establecer conexión.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Abril.	**Fecha Fin:** Abril.
Programador Responsable: Yisel Correa Rodríguez.	
Descripción: El sistema debe permitir establecer una conexión con la base de datos seleccionada y deserializar los ficheros XML seleccionados.	

Tabla 52. Tarea de ingeniería: Establecer conexión

Tarea de ingeniería

Número Tarea: 23	**Número de Historia de Usuario:** 7
Nombre Tarea: Guardar proyecto.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Abril.	**Fecha Fin:** Abril.
Programador Responsable: Yisel Correa Rodríguez.	
Descripción: El sistema debe permitir guardar un proyecto con los ficheros XML de ontología, sintaxis y descripción y el fichero de configuración en cualquier momento. Los contextos no serán validados pues se asume que el proyecto no ha sido terminado.	

Tabla 53. Tarea de ingeniería: Guardar proyecto

	Tarea de ingeniería
Número Tarea: 24	**Número de Historia de Usuario:** 6
Nombre Tarea: Cargar proyecto.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Abril.	**Fecha Fin:** Abril.
Programador Responsable: Yisel Correa Rodríguez.	
Descripción: El sistema debe mostrar una interfaz con campos de entrada que se llenarán con las direcciones de sus respectivos ficheros, los cuales se localizarán con un botón de búsqueda que mostrará los proyectos existentes en la carpeta del usuario autenticado. Los ficheros deben pertenecer al mismo proyecto.	

Tabla 54. Tarea de ingeniería: Cargar proyecto

	Tarea de ingeniería
Número Tarea: 25	**Número de Historia de Usuario:** 12

Nombre Tarea: Adicionar a investigación.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Abril.	**Fecha Fin:** Abril.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe permitir adicionar un concepto o campo a la investigación, lo que lo colocará como un nodo raíz para realizar la navegación a partir del mismo.	

Tabla 55. Tarea de ingeniería: Adicionar a investigación

	Tarea de ingeniería
Número Tarea: 26	**Número de Historia de Usuario:** 12
Nombre Tarea: Eliminar nodo de la investigación.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Abril.	**Fecha Fin:** Abril.
Programador Responsable: Yoslandy López Cristóbal.	
Descripción: El sistema debe permitir eliminar un nodo del árbol de la investigación.	

Tabla 56. Tarea de ingeniería: Eliminar nodo de la investigación

Iteración 5:

	Tarea de ingeniería
Número Tarea: 27	**Número de Historia de Usuario:** 1
Nombre Tarea: Autenticar usuario.	
Tipo Tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: Abril.	**Fecha Fin:** Abril.

Programador Responsable: Yisel Correa Rodríguez.
Descripción: Crear los formularios correspondientes para el inicio de sesión, validando los permisos para los roles correspondientes.

Tabla 57. Tarea de ingeniería: Autenticar usuario

Anexo 4. Tarjetas CRC

Tarjeta CRC	
Clase: OntologyConcept	
Responsabilidades	**Colaboraciones**
Almacenar toda la información referente a un concepto. Controlar la gestión de los campos pertenecientes al concepto. Devolver la información asociada al concepto.	ConceptField Description

Tabla 58. Tarjeta CRC de la clase OntologyConcept

Tarjeta CRC	
Clase: ConceptField	
Responsabilidades	**Colaboraciones**
Almacenar toda la información referente a un campo. Mostrar todos los elementos del campo.	DataType Description

Tabla 59. Tarjeta CRC de la clase ConceptField

Tarjeta CRC	
Clase: OntologyRelation	
Responsabilidades	**Colaboraciones**
Almacenar toda la información referente a una relación.	RelationRestriction

Devolver la información asociada a la relación.	

Tabla 60. Tarjeta CRC de la clase OntologyRelation

Tarjeta CRC	
Clase: RDBConceptualDescription	
Responsabilidades	**Colaboraciones**
Almacenar toda la información referente a la descripción. Controlar la gestión de las consultas para los conceptos y las relaciones. Controlar los elementos para la definición del gestor. Brindar la información necesaria a los elementos representados en el editor de contextos.	DescriptionConcept DescriptionRelation DescriptionGestor

Tabla 61. Tarjeta CRC de la clase RDBConceptualDescription

Tarjeta CRC	
Clase: DescriptionGestor	
Responsabilidades	**Colaboraciones**
Almacenar los elementos pertenecientes al gestor.	

Tabla 62. Tarjeta CRC de la clase DescriptionGestor

Tarjeta CRC	
Clase: DescriptionConcept	
Responsabilidades	**Colaboraciones**
Almacenar toda la información de un concepto referente a la descripción.	

Tabla 63. Tarjeta CRC de la clase DescriptionConcept

Tarjeta CRC	
Clase: DescriptionRelation	
Responsabilidades	**Colaboraciones**
Almacenar toda la información de una relación referente a la descripción.	

Tabla 64. Tarjeta CRC de la clase DescriptionRelation

Tarjeta CRC	
Clase: ConceptQuerySintaxis	
Responsabilidades	**Colaboraciones**
Almacenar toda la información referente a la sintaxis. Controlar la gestión de los operadores. Controlar los elementos para la definición del gestor. Brindar la información necesaria para la gestión de los operadores.	OperationDefinition

Tabla 65. Tarjeta CRC de la clase ConceptQuerySintaxis

Tarjeta CRC	
Clase: OperationDefinition	
Responsabilidades	**Colaboraciones**
Almacenar toda la información referente a un operador.	ExpressionType

Tabla 66. Tarjeta CRC de la clase OperationDefinition

Anexo 5. Diagramas de clases

Figura 30. Diagrama de clases del paquete Ontología extendido

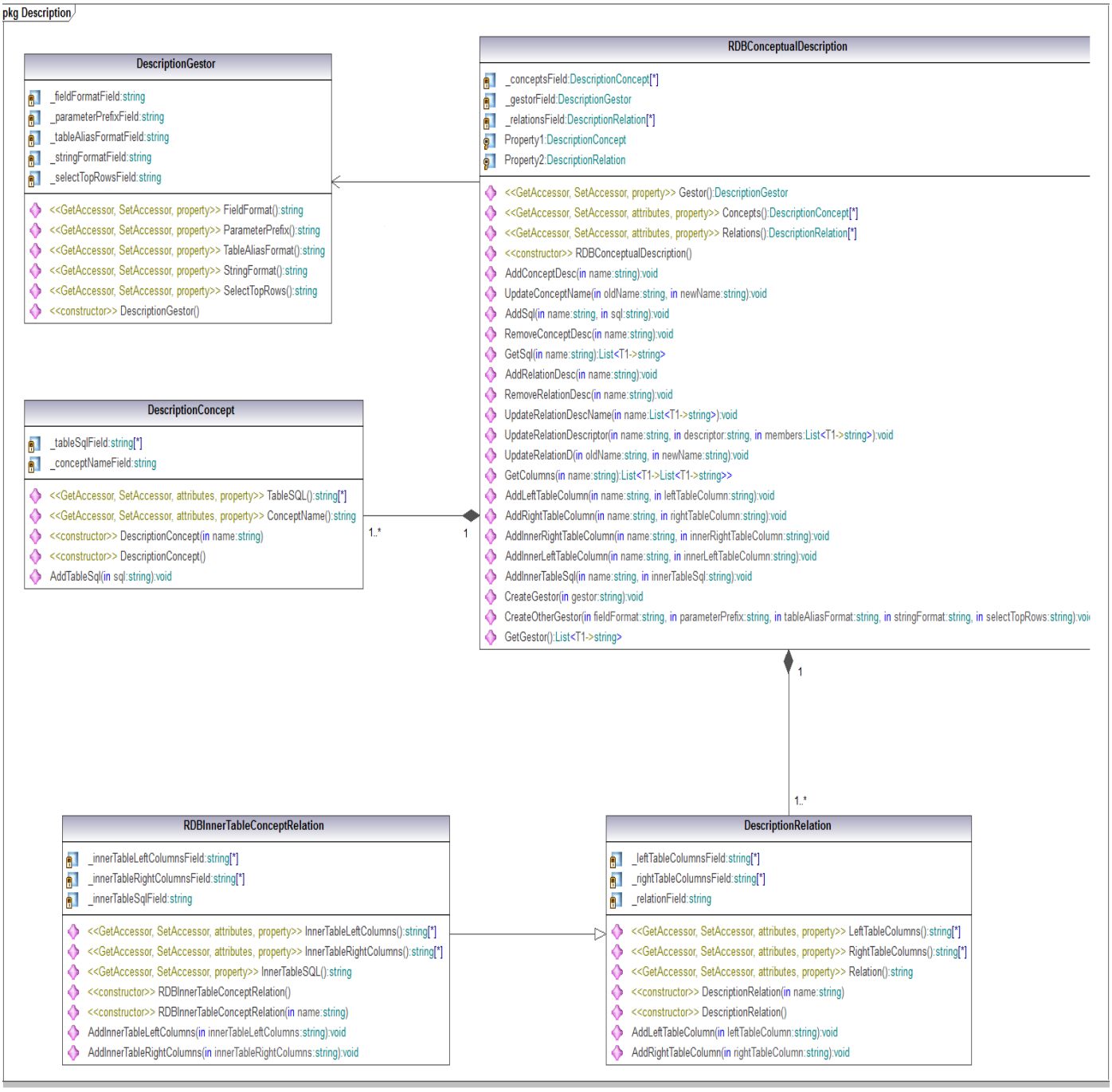


Figura 31. Diagrama de clases del paquete Descripción extendido

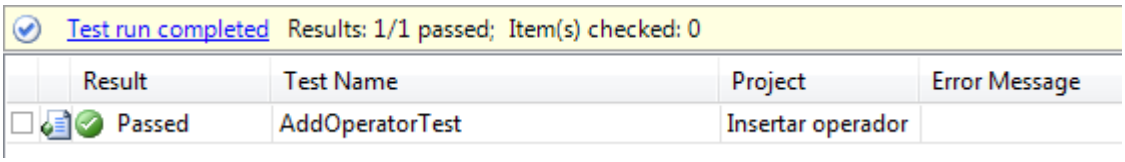
Figura 32. Diagrama de clases del paquete Sintaxis extendido

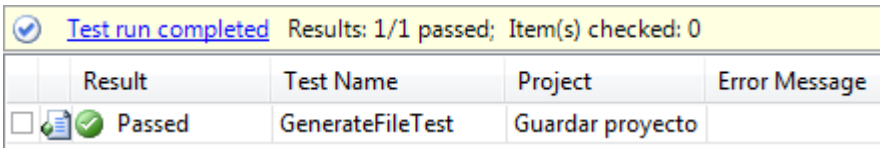
Anexo 6. Diseño de casos de prueba

Diseño de casos de prueba basado en HU

Anexo 7. Pruebas de caja blanca

```
public void AddOperatorTest()
{
    var target = new ConceptQuerySintaxis();
    const string operation = "+";
    const string operationType = "Operation";
    const string precedence = "2";
    const ExpressionType returningType = ExpressionType.Number;
    const string translationFormat = "{0}+{1}";
    var parameters = new ExpressionType[] { ExpressionType.Number, ExpressionType.Number, ExpressionType.Number };
    target.AddOperator(operation, operationType, precedence, returningType, translationFormat, parameters);
}
```





Anexo 8. Pruebas de aceptación

Caso de prueba de Aceptación	
Código Caso de Prueba: HU3-P1	**Nombre de Historia de Usuario:** Gestionar relación
Nombre de la persona que realiza la prueba: Yoslandy López Cristóbal	
Descripción de la Prueba: Prueba de funcionalidad Adicionar relación.	
Condiciones de Ejecución: El usuario debe estar autenticado y con los permisos pertinentes. El usuario debe haber accedido a la página del Editor de Contextos.	
Entradas/Pasos de Ejecución: 1. Seleccionar la opción “Adicionar relación”. 2. Dar clic sobre el concepto dueño de la relación y luego sobre el relacionado.	
Resultado Esperado: La relación debe ser adicionada correctamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 67. Prueba de aceptación: HU3-P1 Adicionar relación

Caso de prueba de Aceptación	
Código Caso de Prueba: HU3-P2	**Nombre de Historia de Usuario:** Gestionar relación
Nombre de la persona que realiza la prueba: Yoslandy López Cristóbal	

Descripción de la Prueba: Prueba de funcionalidad Modificar relación
Condiciones de Ejecución: La relación a modificar debe existir y haber sido seleccionada.
Entradas/Pasos de Ejecución: 1. Doble clic sobre la relación. 2. Introducir los campos (descriptor, descripción de la relación, restricciones). 3. Seleccionar la opción “Aceptar”.
Resultado Esperado: Los campos de la relación deben ser modificados correctamente.
Evaluación de la Prueba: Satisfactoria

Tabla 68. Prueba de aceptación: HU3-P2 Modificar relación

Caso de prueba de Aceptación	
Código Caso de Prueba: HU3-P3	**Nombre de Historia de Usuario:** Gestionar relación
Nombre de la persona que realiza la prueba: Yoslandy López Cristóbal	
Descripción de la Prueba: Prueba de funcionalidad Eliminar relación	
Condiciones de Ejecución: La relación a eliminar debe existir.	
Entradas/Pasos de Ejecución: 1. Seleccionar la opción “Eliminar relación”. 2. Dar clic sobre la relación que se desea eliminar.	
Resultado Esperado: La relación debe ser eliminada correctamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 69. Prueba de aceptación: HU3-P3 Eliminar relación

Caso de prueba de Aceptación	
Código Caso de Prueba: HU3-P4	**Nombre de Historia de Usuario:** Gestionar relación
Nombre de la persona que realiza la prueba: Yoslandy López Cristóbal	

Descripción de la Prueba: Prueba de funcionalidad Buscar relación
Condiciones de Ejecución: La relación a buscar debe existir.
Entradas/Pasos de Ejecución: 1. Seleccionar la opción “Buscar”. 2. Dar doble clic sobre el nombre de la relación que se desea buscar dentro del área de trabajo.
Resultado Esperado: La relación debe ser localizada dentro del área de trabajo correctamente.
Evaluación de la Prueba: Satisfactoria

Tabla 70. Prueba de aceptación: HU3-P4 Buscar relación

Caso de prueba de Aceptación	
Código Caso de Prueba: HU4-P1	**Nombre de Historia de Usuario:** Gestionar consulta
Nombre de la persona que realiza la prueba: Yoslandy López Cristóbal	
Descripción de la Prueba: Prueba de funcionalidad Adicionar consulta	
Condiciones de Ejecución: Se debe haber seleccionado el área de trabajo de la Descripción.	
Entradas/Pasos de Ejecución: 1. Doble clic sobre el concepto/relación. 2. Introducir la consulta. Si es una relación especificar las columnas.	
Resultado Esperado: La consulta debe ser adicionada correctamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 71. Prueba de aceptación: HU4-P1 Adicionar consulta

Caso de prueba de Aceptación	
Código Caso de Prueba: HU4-P2	**Nombre de Historia de Usuario:** Gestionar consulta
Nombre de la persona que realiza la prueba: Yoslandy López Cristóbal	

Descripción de la Prueba: Prueba de funcionalidad Modificar consulta
Condiciones de Ejecución: La consulta a modificar debe existir y hacer sido seleccionada.
Entradas/Pasos de Ejecución: 1. Doble clic sobre el concepto/relación. 2. Seleccionar la opción “Ver consultas creadas” o “Ver columnas creadas” si es una relación. 3. Seleccionar la opción “Modificar” correspondiente a la consulta y modificarla. Si es una relación se debe eliminar la columna y adicionar la nueva. 4. Seleccionar la opción “Aceptar”.
Resultado Esperado: La consulta debe ser modificada correctamente.
Evaluación de la Prueba: Satisfactoria

Tabla 72. Prueba de aceptación: HU4-P2 Modificar consulta

Caso de prueba de Aceptación	
Código Caso de Prueba: HU4-P3	**Nombre de Historia de Usuario:** Gestionar consulta
Nombre de la persona que realiza la prueba: Yoslandy López Cristóbal	
Descripción de la Prueba: Prueba de funcionalidad Eliminar consulta	
Condiciones de Ejecución: La consulta a eliminar debe existir.	
Entradas/Pasos de Ejecución: 1. Doble clic sobre el concepto/relación. 2. Seleccionar la opción “Ver consultas creadas” o “Ver columnas creadas” si es una relación. 3. Seleccionar la opción “Eliminar” correspondiente a la consulta. Si es una relación se eliminar la columna dando clic sobre ella. 4. Seleccionar la opción “Aceptar”.	

Resultado Esperado: La consulta debe ser eliminada correctamente.
Evaluación de la Prueba: Satisfactoria

Tabla 73. Prueba de aceptación: HU4-P3 Eliminar consulta

Caso de prueba de Aceptación	
Código Caso de Prueba: HU5-P1	**Nombre de Historia de Usuario:** Gestionar operador
Nombre de la persona que realiza la prueba: Yoslandy López Cristóbal	
Descripción de la Prueba: Prueba de funcionalidad Adicionar operador	
Condiciones de Ejecución: Se debe haber seleccionado el área de trabajo de la Sintaxis.	
Entradas/Pasos de Ejecución: 1. Introducir los datos del nuevo operador (operación, tipo de operación, miembros, precedencia, tipo de retorno, formato de traducción, tipo de expresión). 2. Seleccionar la opción “Aceptar”.	
Resultado Esperado: El operador debe ser adicionado correctamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 74. Prueba de aceptación: HU5-P1 Adicionar operador

Caso de prueba de Aceptación	
Código Caso de Prueba: HU5-P2	**Nombre de Historia de Usuario:** Gestionar operador
Nombre de la persona que realiza la prueba: Yoslandy López Cristóbal	
Descripción de la Prueba: Prueba de funcionalidad Modificar operador	
Condiciones de Ejecución: El operador a modificar debe existir.	
Entradas/Pasos de Ejecución: 1. Dar clic sobre el operador a modificar. 2. Modificar los campos.	

3. Seleccionar la opción “Aceptar”.
Resultado Esperado: El operador debe ser modificado correctamente.
Evaluación de la Prueba: Satisfactoria

Tabla 75. Prueba de aceptación: HU5-P2 Modificar operador

Caso de prueba de Aceptación	
Código Caso de Prueba: HU5-P3	**Nombre de Historia de Usuario:** Gestionar operador
Nombre de la persona que realiza la prueba: Yoslandy López Cristóbal	
Descripción de la Prueba: Prueba de funcionalidad Eliminar operador	
Condiciones de Ejecución: El operador a eliminar debe existir.	
Entradas/Pasos de Ejecución: 1. Dar clic sobre el operador a eliminar. 2. Seleccionar la opción “Cancelar”.	
Resultado Esperado: El operador debe ser eliminado correctamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 76. Prueba de aceptación: HU5-P3 Eliminar operador

Caso de prueba de Aceptación	
Código Caso de Prueba: HU9-P1	**Nombre de Historia de Usuario:** Establecer conexión
Nombre de la persona que realiza la prueba: Yisel Correa Rodríguez	
Descripción de la Prueba: Prueba de funcionalidad Establecer conexión	
Condiciones de Ejecución: El usuario debe estar autenticado con los permisos requeridos	
Entradas/Pasos de Ejecución: 1. Seleccionar la opción “Base de datos”.	

2. Seleccionar el gestor de base de datos al que se va a conectar. 3. Introducir los campos para la conexión. 4. Seleccionar la opción “Aceptar”.
Resultado Esperado: La conexión debe ser establecida correctamente.
Evaluación de la Prueba: Satisfactoria

Tabla 77. Prueba de aceptación: HU9-P1 Establecer conexión