**CONSTRUCCIÓN DE UNA APLICACIÓN WEB PARA LA ELABORACIÓN DE DIAGRAMAS DE CLASE, ORIENTADA BAJO EL MODELO DE SERVICIOS DE INTERNET CLOUD COMPUTING**

**Gabriel Leonardo Díaz Cárdenas**1, 2, **Marco Antonio Adarme Jaimes**1, 3

1 Programa de Ingeniería de Sistemas, Universidad Francisco de Paula Santander. Avenida Gran Colombia No. 12E-96B Colsag. San José de Cúcuta, Norte de Santander, Colombia.

2 Estudiante, gabrielleonardodc@ufps.edu.co.

3 Magister en Computación, Director del Proyecto, madarme@ufps.edu.co.

**RESUMEN**

El presente artículo expone el proceso de construcción de la herramienta CLASS Modeler; se trata de una aplicación web que permite diseñar diagramas de clase UML de manera online, así como la generación de código fuente Java a partir de ellos. Esta aplicación fue el resultado del proyecto de grado titulado: CONSTRUCCIÓN DE UNA APLICACIÓN WEB PARA LA ELABORACIÓN DE DIAGRAMAS DE CLASE, ORIENTADA BAJO EL MODELO DE SERVICIOS DE INTERNET CLOUD; cuyo enfoque gira en torno al desarrollo de una herramienta tecnológica que sirva de apoyo al proceso educativo de los estudiantes del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander. En este caso el tema de investigación son las herramientas CASE, que constituyen un aspecto importante en el desarrollo de software y resultan de gran ayuda al momento de poner en práctica conceptos como abstracción y modelamiento de sistemas.

CLASS Modeler fue desarrollada usando lenguaje Java a nivel empresarial; el estándar J2EE constituyó el ambiente principal de desarrollo de la herramienta, incorporando tecnologías avanzadas en la construcción de aplicaciones con arquitectura basada en capas. Adicionalmente, se fundamenta en la especificación formal del lenguaje UML en su versión 2.4.1 definida por el Object Management Group [1], utilizando una implementación en lenguaje Java del metamodelo definido por el estándar, lo cual permitió la representación del diagrama en forma de un modelo de objetos en dicho lenguaje. A lo largo del artículo se explica de manera muy general el proceso de desarrollo de la aplicación, abordando temas técnicos de arquitectura y tecnología utilizadas, así como la aplicación de un proceso híbrido entre dos metodologías de desarrollo de software muy conocidas, Programación Extrema XP y El Proceso Unificado UP.

**Palabras Clave:** Aplicación Web, Diagramas de Clase, UML, Herramientas CASE, Desarrollo de Software, Java.

**ABSTRACT**

This article shows the development process of a tool called CLASS Modeler; it is a web application that allows drawing UML class diagrams on internet and generating Java source code from them. This application was the result of the graduation project titled: BUILDING A WEB APPLICATION FOR DRAWING CLASS DIAGRAMS, ORIENTED UNDER THE CLOUD COMPUTING SERVICES MODEL; which is focused around the development of a technological tool that will support the educational process of students of Systems Engineering Program of Francisco de Paula Santander University. In this case the research topic is CASE Tools, which constitute an important aspect of software development and are helpful when practicing concepts such as abstraction and systems modeling.

CLASS Modeler was developed using Java language Enterprise version; J2EE standard was the main development environment tool, incorporating advanced technologies in building layer-based applications. Additionally, it is based on the formal specification of UML in version 2.4.1, defined by the Object Management Group [1]; using a Java implementation of the meta-model defined by this standard, which allowed to represent the diagrams in form of an objects model in such programming language. Throughout this article we will explain in general terms the development process of the application, addressing technical issues like architecture and technology used, as well as the hybrid process between two well-known software development methodologies, Extreme Programming XP and the Unified Process UP.

**Index Terms:** Web Application, Class Diagrams, UML, CASE Tools, Software Development, Java.

1. **INTRODUCCIÓN**

En proyectos de investigación realizados anteriormente [2], se determinó que uno de los principales factores de deserción y mortalidad académica en el Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, es la falta de práctica y comprensión de conceptos como abstracción y modelamiento de sistemas, así como la falta de ambientes para realizar actividades de afianzamiento y refuerzo.

Considerando lo anterior, se hace notoria la necesidad de disponer de mecanismos que ayuden a reforzar los conceptos aprendidos en clase y así disminuir esta problemática que afecta el programa. Es por ello que se planteó la posibilidad de diseñar una nueva herramienta enfocada en las necesidades particulares del ambiente académico.

Si bien se conocía de antemano la existencia software especializado en el diseño de diagramas UML, se determinó también, en base a observaciones y manifestaciones de estudiantes y docentes, que la tecnología existente no era suficiente en algunos casos. Debido que la mayoría es software que se instala de manera nativa, varias de estas aplicaciones requieren de configuración y actualización manual, lo cual resulta en un esfuerzo para adecuar ambientes de trabajo para grupos de estudiantes; por no mencionar la complejidad implícita que representa el uso de estas herramientas por parte de estudiantes que están iniciando sus estudios universitarios.

Se planteó el desarrollo de una aplicación RIA[[1]](#footnote-1), que realizara las funciones mínimas de una herramienta CASE para el diseño de diagramas de clase UML, y que proporcionara características propias de una aplicación web como, portabilidad, facilidad de acceso y facilidad de uso.

CLASS Modeler, nombre que se le dio a la aplicación, se clasifica dentro del modelo de servicios SaaS establecido por Cloud Computing[[2]](#footnote-2); el cual define el software como un servicio que puede ser accedido bajo demanda desde cualquier lugar y en cualquier momento a través de internet.

La herramienta permite únicamente el diseño de diagramas de clase UML y genera código fuente en lenguaje Java; esto debido a que el lenguaje ha sido adoptado por el Programa de Ingeniería de Sistemas para la enseñanza de programación desde el inicio de la carrera.

1. **DESARROLLO DEL PROYECTO**
   1. **¿QUE HACE CLASS MODELER?**

CLASS Modeler dispone de un conjunto de características destinadas a la construcción de diagramas de clase, a continuación se mencionan las características principales de la herramienta:

* Generar de código fuente.
* Generar de imágenes PNG del diagrama.
* Compartir diagramas con otros usuarios.
* Diseñar diagramas de manera concurrente por varios usuarios.
* Generar métodos *get* y *set* de un atributo.
* Generar método *constructor* de una clase.

Adicionalmente se encuentran otras características secundarias enfocadas en el diseño de diagramas.

* Portapapeles
* Deshacer y rehacer.
* Zoom.
* Vista en miniatura.
* Ordenar elementos.
  1. **METODOLOGIA**

En términos de metodología de desarrollo, se optó por realizar un hibrido entre dos metodologías, XP y UP. Esto permitió enfocar el desarrollo de manera iterativa y adaptable a los cambios, sin dejar de lado los lineamientos un proceso robusto y bien documentado.

La metodología estuvo constituida por las siguientes fases:

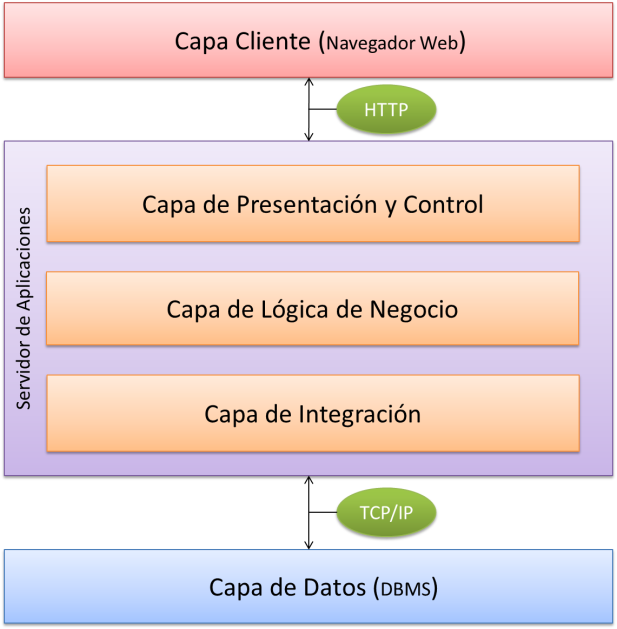
* **Exploración**: Inicio del proyecto, en donde se definieron aspectos importantes que otorgaron los lineamientos para su correcta ejecución. Se destaca la definición de las historias de usuario, los roles del proyecto, los actores del sistema, la arquitectura y la tecnología a utilizar.
* **Planificación**: Se priorizaron las historias de usuario y se determinaron las iteraciones, así como las pruebas unitarias por cada iteración. En la sección 2.5 se habla en detalle de las pruebas de unidad.
* **Construcción**: Incluyó el análisis y diseño de cada historia de usuario. Se diseñó el diagrama de clases y el modelo de datos de cada iteración.
* **Producción**: En esta fase se realizó la liberación de una versión prototipo por cada iteración terminada.
* **Mantenimiento**: Se verificaron de las funcionalidades del sistema, las pruebas de funcionamiento y se corrigieron errores encontrados.
* **Muerte**: Se realizó el manual de usuario y la documentación de funcionalidades.

Las historias de usuario contenían la descripción de las funcionalidades a implementar, las cuales a su vez, fueron producto de un análisis de las herramientas CASE más utilizadas por estudiantes y docentes del programa académico.

* 1. **ARQUITECTURA**

La arquitectura del proyecto está basada en capas, en donde cada capa tiene una labor específica y se comunica con las demás mediante interfaces bien definidas.

La siguiente figura muestra la distribución de las capas.



Se diferencian claramente 3 ambientes de ejecución:

* Navegador web: Representa la interfaz gráfica con la cual interactúan los usuarios.
* Servidor de aplicaciones: Contiene tanto la lógica de negocio como el acceso a los datos, además procesa las peticiones del cliente. Para este caso se utilizó el servidor *Glassfish*.
* Servidor de Base de Datos: Almacena la información persistente. Se utilizó *MySQL*.
  1. **TECNOLOGIAS UTILIZADAS**

Las tecnologías utilizadas se clasifican dentro de cada capa de la arquitectura del sistema, es así como cada tecnología cumple con su labor específica dentro de su ambiente de ejecución.

La siguiente tabla menciona las tecnologías utilizadas para el desarrollo de CLASS Modeler.

|  |  |
| --- | --- |
| **Capa** | **Tecnología** |
| Cliente | jQuery / Javascript |
| CSS |
| HTML |
| XML |
| mxGraph |
| Presentación y Control | JSF [3] |
| PrimeFaces |
| Facelets |
| XHTML |
| Servlets |
| XML Parser |
| Lógica de Negocio | Enterprise JavaBeans |
| UML2/EMF |
| JavaMail |
| StringTemplate |
| Integración | JPA [4] |
| Datos | MySQL |

Cabe destacar dentro de las tecnologías utilizadas, una herramienta que significó una vital importancia en el desarrollo de la capa cliente, se trata de la librería JavaScript mxGraph [5]. Esta librería permite la representación de gráficos en un navegador web mediante el uso de SVG[[3]](#footnote-3).

Si bien mxGraph es una herramienta comercial, puede ser usada de manera gratuita en proyectos educativos, otorgando los créditos pertinentes.

Por otro lado, destaca también el uso de diferentes patrones de diseño, tales como: Model-Vista-Controlador, Data Access Object, Singleton y Front Controller [6].

* 1. **UML A CÓDIGO FUENTE**

La generación de código representó uno de los aspectos más importantes de implementación del proyecto. Como se mencionó anteriormente CLASS Modeler está basado en una implementación del metamodelo definido por la OMG para el estándar UML. Este metamodelo define una estructura de clases relacionadas que conforman la vista estática del lenguaje.

Esta implementación se llama UML2 [7], la cual es una librería (JAR[[4]](#footnote-4)) que contiene un conjunto de clases representando cada elemento de los diagramas UML: Clase, Interfaz, Relación, Atributo, Método. Esta librería es parte del *Eclipse Modeling Framework,* que es una suite de componentes para diseño de diagramas para el IDE Eclipse.

En la capa cliente de CLASS Modeler, los diagramas son representados en formato XML. Cada elemento del diagrama contiene un objeto XML y los atributos representan la información básica de dicho elemento: nombre, visibilidad, tipo de dato, entre otros.

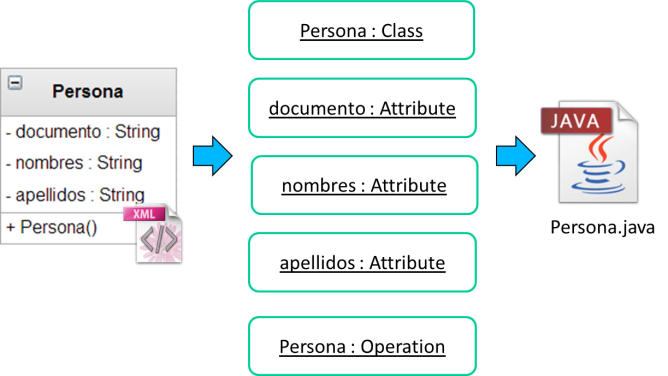
La comunicación entre el cliente y el servidor se realiza mediante el envío de mensajes AJAX transportando el XML de los elementos editados. Estos mensajes son recibidos por la capa de control hasta llegar a la lógica de negocio.

Al momento de generar el código fuente, todo el XML del diagrama es procesado mediante un *Parser* y se obtiene un modelo UML basado en objetos. La transformación que realiza dicho *Parser* es una implementación propia desarrollada para el proyecto, utilizando las clases del dominio provistas por la librería UML2.

Una vez se obtiene la representación del modelo UML, se realiza la generación del código fuente en lenguaje Java. Para esta labor se implementó un servicio EJB que permite tomar el modelo y en base a un sistema de plantillas de texto, generar cualquier tipo de archivos de texto plano.

La herramienta utilizada para la generación de los archivos de código se llama StringTemplate [8], esta librería de código abierto, permite definir plantillas de texto plano con secciones variables que pueden ser reemplazadas utilizando un modelo de objetos.

La siguiente imagen muestra el proceso de transformación que sigue la herramienta para generar el código fuente para una clase llamada *Persona*.



* 1. **PRUEBAS DE UNIDAD**

Las pruebas de unidad se definieron al inicio de cada iteración, estas pruebas se enfocaron en la verificación de los servicios EJB que soportan la lógica de negocio de toda la aplicación, incluyendo los servicios que permiten la generación de código y la interacción con la capa de datos.

Para esta labor se dispuso de un Framework para realizar pruebas automáticas, llamado TestNG [9]. Este Framework permite configurar ambientes en los cuales se pueden ejecutar pruebas de software simulando ambientes reales. El Framework se usó en conjunto con el servidor de aplicaciones Glassfish embebido. Al iniciar la ejecución de las pruebas se lanza de manera automática una instancia del servidor que se encarga de desplegar todos los servicios que serán probados.

1. **RESULTADOS**

Se obtuvo una aplicación web que permite diseñar diagramas de clase UML y generar código fuente en lenguaje Java a partir del diagrama construido. De esta manera se obtiene un mecanismo adicional que sirve como apoyo al proceso educativo de los estudiantes, permitiendo practicar y reforzar conceptos de modelamiento y abstracción de sistemas.

1. **CONCLUSIONES**

Con el desarrollo de CLASS Modeler, se logró construir una herramienta de apoyo para el proceso de aprendizaje de los estudiantes del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander; muy útil para la realización de prácticas sobre el modelado de diagramas de clase, promoviendo aspectos importantes como el trabajo en equipo y la colaboración.

CLASS Modeler proporciona beneficios que resultan atractivos a la hora de utilizar una herramienta de modelado UML, tales como la facilidad de acceso, inmediatez a la hora de diseñar diagramas y portabilidad de la información. CLASS Modeler solo necesita una conexión a internet y un navegador web para ser usado, y no requiere instalación de complementos como Flash o Shockwave.

CLASS Modeler se puede clasificar dentro del modelo de servicios SaaS, donde el soporte lógico y los datos de la aplicación se encuentran en un servidor de TI y los clientes acceden bajo demanda mediante un navegador web.

Un aspecto muy importante del proyecto fue la capacidad de convertir el metamodelo de clases de UML a código fuente en lenguaje Java. Esto representó un logro bastante importante debido a la complejidad implícita del proceso y la diferencia que existe entre ambos lenguajes. En algunos casos fue necesario extender el metamodelo para soportar conceptos propios de Java, como son los Tipos Genéricos, utilizados a la hora de crear colecciones de objetos.

**REFERENCIAS**

1. **OMG Object Management Group.** Unified Modeling Language Superstructure Definition. [En línea] 6 de Agosto de 2011. http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/. formal/2011-08-06.

2. **Amaya Torrado, Yegny Karina y Herrera Angarita, Lady Torcoroma.** *Identificación de las causas que generan problemas en el aprendizaje de fundamentos de programación de computadores en las facultades de ingenieira de sistemas de las universidades de la ciudad de Cúcuta.* Cúcuta N. de S. : Universidad Francisco de Paula Santander, Biblioteca Eduardo Cote Lamus, 2003. TIS 371.334A489i.

3. **Geary, David y Horstmann, Cay.** *Core Java Server Faces.* Boston : Prentince Hall, 2010. ISBN 9780137012893.

4. **Keith, Mike y Schincariol, Merrick.** *Pro JPA 2 Mastering the Java Persistence API.* New York : Apress, 2009. ISBN 9781430219569.

5. **Benson, David y Alder, Gaudenz.** mxGraph JavaScript HTML 5 diagramming library. [En línea] JGraph Company, 26 de 12 de 2012. [Citado el: 23 de 01 de 2013.] http://www.jgraph.com/mxgraph.html.

6. **Gamma, Erich, y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1994. 0-201-63361-2.

7. **Eclipse Org.** UML2/MDT Documentation. [En línea] http://wiki.eclipse.org/MDT-UML2.

8. **Parr, Terence.** Documentación de la libreria StringTemplate. [En línea] 2013. https://theantlrguy.atlassian.net/wiki/display/ST4/StringTemplate+4+Documentation.

9. **Beust, Cédric y Suleiman, Hani.** *Next Generation Java Testing: TestNG and Advanced Concepts.* Boston : Addison Wesley, 2008. 978-0-321-50310-7.

1. Rich Internet Applications, son aplicaciones web que tienen la mayoría de características de las aplicaciones de escritorio normales. [↑](#footnote-ref-1)
2. Cloud Computing, es un paradigma que permite ofrecer servicios de computación a través de internet. [↑](#footnote-ref-2)
3. Scalable Vector Graphics, son una especificación para describir gráficos vectoriales

   bidimensionales, tanto estáticos como animados en formato XML. [↑](#footnote-ref-3)
4. JAR es un tipo de archivo que permite ejecutar aplicaciones escritas en el lenguaje Java. [↑](#footnote-ref-4)