CONSTRUCCIÓN DE UNA APLICACIÓN WEB PARA LA ELABORACIÓN DE DIAGRAMAS DE CLASE, ORIENTADA BAJO EL MODELO DE SERVICIOS DE INTERNET CLOUD COMPUTING  
(Proyecto de Grado Dirigido)

GABRIEL LEONARDO DÍAZ CÁRDENAS

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER  
FACULTAD DE INGENIERIAS  
PLAN DE ESTUDIOS DE INGENIERIA DE SISTEMAS  
SAN JOSÉ DE CÚCUTA  
2013

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  
Código 0152685

Trabajo de grado para ostentar el título de Ingeniero de Sistemas de la Universidad Francisco de Paula Santander

DIRECTOR  
MARCO ANTONIO ADARME JAIMES  
INGENIERO DE SISTEMAS  
MAGISTER EN CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER  
FACULTAD DE INGENIERIAS  
PLAN DE ESTUDIOS DE INGENIERIA DE SISTEMAS  
SAN JOSÉ DE CÚCUTA  
2013

**TABLA DE CONTENIDO**

[LISTADO DE ILUSTRACIONES 7](#_Toc365828648)

[LISTADO DE TABLAS 8](#_Toc365828649)

[INTRODUCCIÓN 9](#_Toc365828650)

[1. PRESENTACIÓN DEL PROYECTO 11](#_Toc365828651)

[1.1 TITULO 11](#_Toc365828652)

[1.2 PLANTEAMIENTO DEL PROBLEMA 11](#_Toc365828653)

[1.3 JUSTIFICACIÓN 12](#_Toc365828654)

[1.4 OBJETIVOS 13](#_Toc365828655)

[1.4.1 Objetivo General 13](#_Toc365828656)

[1.4.2 Objetivos Específicos 14](#_Toc365828657)

[1.5 ALCANCES Y DELIMITACIONES 14](#_Toc365828658)

[1.5.1 Alcance 14](#_Toc365828659)

[1.5.2 Delimitaciones 15](#_Toc365828660)

[2. MARCO REFERENCIAL 17](#_Toc365828661)

[2.1 MARCO DE ANTECEDENTES 17](#_Toc365828662)

[2.1.1 Draw.io 17](#_Toc365828663)

[2.1.2 Creately 17](#_Toc365828664)

[2.1.3 GWTUML 18](#_Toc365828665)

[2.1.4 JoinJS 19](#_Toc365828666)

[2.2 MARCO CONCEPTUAL 19](#_Toc365828667)

[2.2.1 Herramientas CASE 19](#_Toc365828668)

[2.2.2 UML 21](#_Toc365828669)

[2.2.3 Diagramas de Clase 23](#_Toc365828670)

[2.2.4 Meta-Programación 23](#_Toc365828671)

[2.2.5 Cloud Computing 25](#_Toc365828672)

[2.3 MARCO TEORICO 28](#_Toc365828673)

[2.3.1 Java 28](#_Toc365828674)

[2.3.2 Frameworks 29](#_Toc365828675)

[2.3.3 Servidor de Aplicaciones 31](#_Toc365828676)

[2.3.4 Extreme Programming (XP) 32](#_Toc365828677)

[2.3.5 SVG 34](#_Toc365828678)

[2.3.6 HTML, JavaScript y CSS 35](#_Toc365828679)

[2.4 MARCO DE LEGAL 36](#_Toc365828680)

[2.4.1 Acuerdos de uso 36](#_Toc365828681)

[2.4.2 Contrato de licencia de código binario, SUN MICROSYSTEMS 37](#_Toc365828682)

[2.4.3 GNU General Public Licence 37](#_Toc365828683)

[2.4.4 Propiedad Intelectual 38](#_Toc365828684)

[3. DISEÑO METODOLÓGICO 39](#_Toc365828685)

[3.1 TIPO DE INVESTIGACIÓN 39](#_Toc365828686)

[3.2 FUENTES DE INFORMACIÓN 39](#_Toc365828687)

[3.2.1 Fuentes de información primaria 39](#_Toc365828688)

[3.2.2 Fuentes de información secundaria 40](#_Toc365828689)

[3.3 RECOLECCIÓN Y ANALISIS DE INFORMACIÓN 40](#_Toc365828690)

[4. ANALISIS DE LAS HERRAMIENTAS CASE 42](#_Toc365828691)

[4.1 ENTERPRISE ARCHITECT 42](#_Toc365828692)

[4.1.1 Historia 43](#_Toc365828693)

[4.1.2 Características Principales 44](#_Toc365828694)

[4.1.3 Funciones para Diagramas de Clase 46](#_Toc365828695)

[4.2 STARUML 62](#_Toc365828696)

[4.2.1 Historia 62](#_Toc365828697)

[4.2.2 Características Principales 63](#_Toc365828698)

[4.2.3 Funciones para Diagramas de Clase 64](#_Toc365828699)

[4.3 DRAW.IO 64](#_Toc365828700)

[5. CONCEPTOS DE DIAGRAMAS DE CLASES 65](#_Toc365828701)

[6. CLASIFICACIÓN DE LA INFORMACIÓN 66](#_Toc365828702)

[6.1 MATRIZ DE CARACTERISTICAS 66](#_Toc365828703)

[6.2 CARACTERISTICAS PROPIAS 67](#_Toc365828704)

[6.3 LISTA DE FUNCIONALIDADES 67](#_Toc365828705)

[7. METODOLOGÍA DE DESARROLLO 69](#_Toc365828706)

[7.1 EXPLORACIÓN 69](#_Toc365828707)

[7.1.1 Identificación de Roles del Proyecto 69](#_Toc365828708)

[7.1.2 Identificación de Actores del Sistema 70](#_Toc365828709)

[7.1.3 Historias de Usuario 74](#_Toc365828710)

[7.2 PLANIFICACIÓN 85](#_Toc365828711)

[7.2.1 Priorización de Historias de Usuario 85](#_Toc365828712)

[7.2.2 Estimación de Historias de Usuario 87](#_Toc365828713)

[7.2.3 Definición de Iteraciones 88](#_Toc365828714)

[7.2.4 Metáfora del Sistema 89](#_Toc365828715)

[7.3 DESCRIPCIÓN DE LAS ITERACIONES 89](#_Toc365828716)

[7.3.1 Iteración 1 89](#_Toc365828717)

[7.3.2 Iteración 2 90](#_Toc365828718)

[7.3.3 Iteración 3 90](#_Toc365828719)

[7.3.4 Iteración 4 90](#_Toc365828720)

[7.3.5 Iteración 5 90](#_Toc365828721)

[7.3.6 Iteración 6 90](#_Toc365828722)

[7.3.7 Iteración 7 90](#_Toc365828723)

[7.3.8 Iteración 8 90](#_Toc365828724)

[7.3.9 Iteración 9 90](#_Toc365828725)

[7.4 PLANIFICACIÓN DE LAS PRUEBAS 90](#_Toc365828726)

[7.4.1 Casos de Prueba 90](#_Toc365828727)

[7.4.2 Pruebas Unitarias 90](#_Toc365828728)

[8. IMPLEMENTACIÓN 91](#_Toc365828729)

[8.1 DEFINICIÓN DE LA ARQUITECTURA 91](#_Toc365828730)

[8.2 DEFINICIÓN DE LA TECNOLOGIA 93](#_Toc365828731)

[8.3 PATRONES DE DESARROLLO 94](#_Toc365828732)

[8.4 BASES DE DATOS 94](#_Toc365828733)

[8.5 CONFIGURACIÓN 94](#_Toc365828734)

[8.6 ESPECIFICACIÓN DE FUNCIONALIDADES 94](#_Toc365828735)

[8.7 DESPLIEGUE 94](#_Toc365828736)

[9. ASEGURAMIENTO DE LA CALIDAD 95](#_Toc365828737)

[9.1 CASOS DE PRUEBA 95](#_Toc365828738)

[9.2 PRUEBAS DE UNIDAD 95](#_Toc365828739)

[9.3 ANALISIS ESTATICO DE CODIGO 95](#_Toc365828740)

[CONCLUSIONES 96](#_Toc365828741)

[REFERENCIAS BIBLIOGRÁFICAS 97](#_Toc365828742)

# LISTADO DE ILUSTRACIONES

[Ilustración 1. Jerarquía de los diagramas UML [3] 23](#_Toc365828743)

[Ilustración 2. Modelo de servicios de Cloud Computing 26](#_Toc365828744)

[Ilustración 3. Ciclo de vida de Extreme Programming 34](#_Toc365828745)

[Ilustración 4. Ejemplo de diagrama de clase Enterprise Architect 46](#_Toc365828746)

[Ilustración 5. Caja de herramientas para diagramas de clase Enterprise Architect 47](#_Toc365828747)

[Ilustración 6. Explorador de proyecto Enterprise Architect 48](#_Toc365828748)

[Ilustración 7. Panel de propiedades de Enterprise Architect 49](#_Toc365828749)

[Ilustración 8. Elementos de diagramas de clase Enterprise Architect 50](#_Toc365828750)

[Ilustración 9. Menú contextual de una Clase en Enterprise Architect 51](#_Toc365828751)

[Ilustración 10. Representación de una Clase en Enterprise Architect 52](#_Toc365828752)

[Ilustración 11. Ventana de creación de atributos Enterprise Architect 52](#_Toc365828753)

[Ilustración 12. Ventana de operaciones Enterprise Architect 53](#_Toc365828754)

[Ilustración 13. Conectores para diagramas de clase Enterprise Architect 54](#_Toc365828755)

[Ilustración 14. Ventana de generación de código Enterprise Architect 55](#_Toc365828756)

[Ilustración 15. Ventana de Opciones de disposición de diagrama Enterprise Architect 58](#_Toc365828757)

[Ilustración 16. Ventana de importación de XMI Enterprise Architect 60](#_Toc365828758)

[Ilustración 17. Ventana de exportación de XMI Enterprise Architect 61](#_Toc365828759)

[Ilustración 18. Diagrama de Actores 73](#_Toc365828760)

[Ilustración 19. Arquitectura de la aplicación 91](#_Toc365828761)

# LISTADO DE TABLAS

[Tabla 1. Características principales de Enterprise Architect 45](#_Toc365828762)

[Tabla 2. Características principales de StarUML 64](#_Toc365828763)

[Tabla 3. Especificación del actor Visitante 71](#_Toc365828764)

[Tabla 4. Especificación del actor Usuario 71](#_Toc365828765)

[Tabla 5. Especificación del actor Invitado 72](#_Toc365828766)

[Tabla 6. Especificación del actor Diagramador 72](#_Toc365828767)

[Tabla 7. Formato de historia de usuario H1. 75](#_Toc365828768)

[Tabla 8. Formato de historia de usuario H2 75](#_Toc365828769)

[Tabla 9. Formato de historia de usuario H3. 76](#_Toc365828770)

[Tabla 10. Formato de historia de usuario H4 77](#_Toc365828771)

[Tabla 11. Formato de historia de usuario H5 77](#_Toc365828772)

[Tabla 12. Formato de historia de usuario H6 78](#_Toc365828773)

[Tabla 13. Formato de historia de usuario H7 78](#_Toc365828774)

[Tabla 14. Formato de historia de usuario H8 79](#_Toc365828775)

[Tabla 15. Formato de historia de usuario H9 79](#_Toc365828776)

[Tabla 16. Formato de historia de usuario H10 80](#_Toc365828777)

[Tabla 17. Formato de historia de usuario H11 80](#_Toc365828778)

[Tabla 18. Formato de historia de usuario H12 81](#_Toc365828779)

[Tabla 19. Formato de historia de usuario H13 81](#_Toc365828780)

[Tabla 20. Formato de historia de usuario H14 82](#_Toc365828781)

[Tabla 21. Formato de historia de usuario H15 83](#_Toc365828782)

[Tabla 22. Formato de historia de usuario H16 83](#_Toc365828783)

[Tabla 23. Formato de historia de usuario H17 84](#_Toc365828784)

[Tabla 24. Formato de historia de usuario H18 84](#_Toc365828785)

[Tabla 25. Formato de historia de usuario H19 84](#_Toc365828786)

[Tabla 26. Formato de historia de usuario H20 85](#_Toc365828787)

[Tabla 27. Formato de historia de usuario 21 85](#_Toc365828788)

[Tabla 28. Priorización de Historias de Usuario 87](#_Toc365828789)

[Tabla 29. Estimación de tiempo para Historias de Usuario 88](#_Toc365828790)

[Tabla 30. Plan de iteraciones 88](#_Toc365828791)

# INTRODUCCIÓN

En el desarrollo de software es común usar herramientas que faciliten las tareas complejas y agilicen el trabajo manual; la mayoría o casi todas las empresas dedicadas al desarrollo de software incorporan aplicaciones a su quehacer diario, permitiendo a sus trabajadores llevar a cabo su labor de manera más eficiente, conjunta y automatizada.

Este grupo de aplicaciones son llamadas *Herramientas CASE*, debido a las siglas tomadas del nombre en inglés ***“C****omputer* ***A****ided* ***S****oftware* ***E****ngineering”*, traducido al español como *Ingeniería de Software asistida por computadora*. Estas herramientas son la base para la creación de todas las aplicaciones software que vemos hoy en día y cada una se especializa en diferentes aspectos, dependiendo de la o las fases del proceso de desarrollo que apoyan. Algunas con un grado de complejidad bastante alto, permiten controlar prácticamente todo el ciclo de vida de una aplicación desde el análisis de requerimientos hasta el despliegue y mantenimiento.

Tras un largo camino en la evolución del software y en metodologías de desarrollo, se han incorporado nuevos modelos y tendencias para la construcción de aplicaciones. Debido al alto impacto que tuvo el internet, ha surgido un modelo de negocio en donde se rompe el concepto tradicional de las aplicaciones que residen estáticas en la máquina del usuario y este a su vez compra los derechos para usar en todo momento determinada aplicación; hoy por hoy hablamos de un esquema en el que las aplicaciones se alojan en lugares remotos y los usuarios hacen uso de ellas bajo demanda mediante la red global Internet y pagando únicamente el consumo que se hace de estas (en el caso del Software Propietario). Desde hace varios años llamado *Cloud Computing o Computación en la Nube*, este modelo ha cambiado la concepción que se tenía anteriormente del uso de aplicaciones software.

Con el desarrollo de este proyecto se pretende incorporar los conceptos mencionados anteriormente en una herramienta software, aprovechando las bondades que brindan las aplicaciones en la nube y resolviendo una necesidad particular de nuestro ambiente universitario. De este modo los estudiantes podrán disponer de una herramienta básica para su formación y que posea las características necesarias para ser considerada una aplicación en la nube.

El objetivo principal del proyecto es el desarrollo de una herramienta para el modelado de Diagramas de Clase UML, que le permita a los usuarios crear y manipular de manera online los componentes básicos de este tipo de diagrama y llegando hasta obtener archivos de código fuente generados a partir de los elementos del mismo.

Llamada *CLASS Modeler* (Modelador de Clases) debido a su naturaleza y enfoque, esta herramienta pretende solventar una necesidad detectada, además de servir como apoyo al proceso de aprendizaje de los estudiantes. A lo largo de este documento se expondrán todas las técnicas empleadas para su construcción y la metodología usada como guía del proceso de desarrollo.

# PRESENTACIÓN DEL PROYECTO

## TITULO

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

## PLANTEAMIENTO DEL PROBLEMA

En estudios realizados por el grupo de investigación y desarrollo de software (GIDIS) de la Universidad Francisco de Paula Santander, después convertido en un proyecto de grado [1], se ha determinado que unos de los principales factores involucrados en los altos índices de mortalidad y deserción académica en la carrera de Ingeniería de Sistemas de las universidades de Cúcuta, son la falta de dedicación al estudio por parte de los estudiantes, poca comprensión de temas como abstracción y modelamiento de sistemas, y falta de ambientes prácticos donde los estudiantes tuvieran la posibilidad de llevar a cabo actividades para afianzar los conceptos aprendidos en clase. Teniendo esto en cuenta se hace evidente que se necesitan mecanismos para que los estudiantes ejerciten sus habilidades y de esta manera se disminuya esta problemática que ha afectado considerablemente al programa académico.

Dentro del programa de Ingeniería de Sistemas de la UFPS se tiene un clara tendencia e inclinación hacia el desarrollo de software, es claro para muchos estudiantes que este es uno de los puntos fuertes de la carrera y es por ello que muchos continúan su vida profesional por ese camino; por tal motivo desde los primeros semestres se trata de cultivar estas habilidades en ellos; considerando que el modelamiento y abstracción de sistemas son temas de vital importancia, se hace necesario disponer de herramientas automáticas (software) que permitan adquirir competencias en estas áreas y reforzar conceptos aprendidos en clase, pero que solo se consiguen mediante la práctica; además que permitan al estudiante forjar unos cimientos importantes para su vida profesional en aspectos tales como UML, Diagramas de clase, CASE Tools[[1]](#footnote-1) y razonamiento abstracto.

Por otro lado, se es consciente que actualmente ya existen herramientas muy completas que permiten realizar estas labores y que el programa académico ha venido utilizando desde hace algún tiempo (por ejemplo Enterprise Architect[[2]](#footnote-2), Netbeans UML, StarUML, ArgoUML), además que la universidad dispone de licencias de uso para algunas de ellas (por ejemplo, Enterprise Architect que es software privado), y otras son software libre de modo que pueden ser utilizados de cualquier forma; sin embargo, la mayoría de ellas, por no decir todas, son software que se instala de manera nativa, de modo que carecen de portabilidad y facilidad de acceso, aspectos que son muy importantes y atractivos hoy en día en los software de aplicación. Otras requieren ciertas características mínimas de hardware que aunque no es muy común, no siempre están disponibles y son suficientes para soportar que estas aplicaciones se ejecuten de manera adecuada.

Adicional pero no menos importante, se ha notado mediante observaciones indirectas y manifestaciones explicitas de algunos estudiantes, sobre todo en primeros semestres, que muchos de ellos aún tienen dificultades en la configuración de estas aplicaciones, otros expresan no tener computadoras propias o simplemente que los recursos hardware son insuficientes para el funcionamiento adecuado de las aplicaciones, aspectos que evidencian la necesidad planteada anteriormente e impulsan a diseñar modelos de servicios accesibles a través internet, los cuales podrían brindar ciertas flexibilidades y beneficios extras para los usuarios que no tienen las aplicaciones nativas, beneficios tales como facilidad de acceso, portabilidad, autoconfiguración, capacidad de compartir recursos e información.

## JUSTIFICACIÓN

El programa académico de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, tiene como objeto de estudio la “Construcción y Gestión de Sistemas de Información”[[3]](#footnote-3), por esta razón dentro del pensum existen muchos cursos que se enfocan en programación e ingeniería de software; esto significa que hay una clara tendencia a tratar temas como el modelado de aplicaciones, procesos de desarrollo y otros conceptos que hacen parte de esta área de la ingeniería de sistemas; de esta manera consideramos que se necesita que los estudiantes pongan en práctica técnicas y métodos para desarrollar y mantener software de calidad, teniendo siempre en mente que la tecnología existe para ser usada como herramienta facilitadora del trabajo.

Lo anterior sugiere la necesidad de disponer de una herramienta que permita realizar prácticas de laboratorio, más específicamente prácticas de abstracción de diagramas de clase, de modo que se puedan aplicar todos los conceptos aprendidos, bajo un ambiente real. Es por ello que nos hemos enfocado específicamente en este aspecto, de tal manera que se pueda obtener una herramienta adicional que sirva de apoyo al proceso de aprendizaje y permita adquirir con seguridad conceptos vitales y que serán de gran ayuda en la vida profesional de los estudiantes.

De igual manera sería ideal que los docentes puedan disponer de un mecanismo de control y supervisión de las actividades realizadas por sus estudiantes, permitiendo de manera fácil y cómoda la revisión de los trabajos de clase o evaluaciones parciales; por tal motivo se pretende construir una aplicación orientada bajo el modelo de servicios “Cloud Computing”[[4]](#footnote-4), e incorporar comportamientos como los desarrollados por “Google Drive”[[5]](#footnote-5), en donde los usuarios pueden compartir información con otros de manera fácil y cómoda a través de un clic. Aplicando este concepto a la aplicación en construcción, se espera que los estudiantes puedan compartir con su docente los diagramas realizados de modo que el docente pueda consultarlos y corregirlos a través de la aplicación. Además es significativo incentivar el autoaprendizaje, permitiendo a los estudiantes el intercambio de conocimientos e información, de manera que puedan reforzar los conceptos, obviamente mediante algunos mecanismos de control que garanticen un proceso limpio y transparente.

Adicionalmente una motivación importante para el desarrollo de este proyecto es mejorar la infraestructura de servicios y de tecnologías de información disponibles para los estudiantes de la UFPS, mediante la construcción de herramientas innovadoras y que generen conocimientos sobre el desarrollo de aplicaciones web.

Por último, otro aspecto importante es la necesidad de impulsar e innovar en el desarrollo de tecnologías, el compromiso que tiene el plan de estudios de Ingeniería de Sistemas con la universidad y con la comunidad en general, para desarrollar alternativas de solución a problemáticas mediante el uso de la tecnología y generar conocimiento en base a investigación e innovación nos ha llevado a esta iniciativa, esperando propulsar el desarrollo de nuevos productos y herramientas que presten servicios dentro y fuera de la universidad, y que sean realizadas por los mismos estudiantes.

## OBJETIVOS

### Objetivo General

Implementar una aplicación web basada en el paradigma orientado a servicios de Cloud Computing, que permita el diseño y construcción de diagramas de clase UML y que sirva como medio de apoyo para proyectos de software realizados por los estudiantes de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander.

### Objetivos Específicos

* Realizar un estudio de las principales características que poseen las herramientas CASE utilizadas por los estudiantes del programa de Ingeniería de Sistemas, para determinar los servicios que prestan en cuanto al diseño de diagramas de clase.
* Definir las funcionalidades que el software debe poseer, determinando las características específicas, los servicios que debe prestar a los usuarios y los procedimientos que debe llevar a cabo.
* Realizar el modelado del sistema, determinando una solución óptima a los procedimientos definidos, de modo que no se vea afectado el rendimiento y la estabilidad del software a construir.
* Realizar pruebas de funcionamiento que garanticen el comportamiento adecuado del software y que permitan verificar que se cumplan los requerimientos definidos.
* Documentar las pruebas realizadas junto con los procedimientos llevados a cabo y los resultados obtenidos.
* Desplegar la aplicación en un servidor de modo que pueda ser accedido por estudiantes y docentes.

## ALCANCES Y DELIMITACIONES

### Alcance

La ejecución de este proyecto pretende alcanzar lo siguiente:

* Obtener un producto software que permita a los usuarios diseñar de manera visual diagramas de clase UML.
* Disponer de una plataforma online que pueda ser utilizada desde cualquier lugar y a cualquier hora a través de internet.
* Disponer de una herramienta que permita a los docentes asesorar de manera continua y cómoda a sus estudiantes en los proyectos desarrollo de software.
* Un software para realizar modelado de objetos que sirva como soporte de los proyectos de software y permita generar código fuente de manera automática, agilizando el proceso de desarrollo.
* Un documento mediante el cual se pueda conocer el funcionamiento básico del software y los servicios que este presta, que sirva como guía a los usuarios sobre la manera en la cual deben utilizar la herramienta (Manual de usuario).

### Delimitaciones

Con el propósito de determinar la viabilidad y posibilidad de desarrollo del proyecto se han establecido las delimitaciones mencionadas a continuación:

* El software será totalmente independiente del Sistema de Información Académico de la Universidad (SIA), de modo que no se validara que los usuarios registrados estén en la base de datos del mismo.
* Solo se podrá realizar diagramas de clase UML, la aplicación no abarca el diseño de otro tipo de diagrama.
* El software no dispondrá de un administrador, de modo que no será necesario disponer de usuarios que realicen configuraciones específicas y únicas en el software.
* El código fuente generado a partir de los diagramas de clase será únicamente en lenguaje Java.
* Considerando que existen varios cursos de programación e Ingeniería de Software dentro del pensum académico actual del programa de Ingeniería de Sistemas, el proyecto no estará dirigido específicamente a un curso en particular, sino que estará disponible para cualquier estudiante en cualquier curso; sin embargo, se tomara una población base de uno o dos cursos para realizar el levantamiento de información. En el diseño metodológico se especificará con más detalle este tema.
* El software no permitirá editar el código fuente generado directamente, para esta labor se hace necesario que el usuario disponga de un editor.

# [MARCO REFERENCIAL](#TablaContenido)

## MARCO DE ANTECEDENTES

Después de realizar una investigación de las soluciones existentes a la problemática planteada, se encontraron algunas aplicaciones en el mercado que ofrecen parte de las características que esperamos obtener con el software en construcción. A continuación se mencionan algunas de estas aplicaciones:

### Draw.io

Draw.io es una aplicación de diagramado online desarrollada usando la librería JavaScript mxGraph creada por la compañía JGraph Ltd. Esta empresa fue fundada en el año 2000 y está dedicada únicamente a la comercialización y soporte de componentes gráficos de visualización desde el 2004. Actualmente se ha impuesto en el mercado con su producto líder *mxGraph*, el cual comprende una librería JavaScript que permite dibujar componentes para diagramas en cualquier navegador web usando HTML, CSS, JavaScript y SVG, además que permite realizar el backend[[6]](#footnote-6) de las aplicaciones en diferentes lenguajes como Java, PHP y .NET.

Draw.io es el sitio de demostración para la tecnología desarrollada por la empresa JGraph y su producto mxGraph, el cual permite dibujar diagramas creativos en el navegador, compartir y editar con otros usuarios en tiempo real. Esta aplicación puede ser accedida directamente desde la URL: <https://www.draw.io/>.

En el capítulo 4 se analizará con más detalle las características de esta herramienta en cuanto al diseño de diagramas de clase.

### Creately

Creately es una herramienta online UML para crear diagramas de Casos de uso, Diagramas de clase, Actividad, Secuencia, Modelos Entidad Relación, entre muchos otros. Esta es una herramienta comercial la cual es propiedad de la empresa CinergixPty Ltd., es el primer producto de la empresa y se encuentra actualmente en una etapa madura; es posible acceder a él en modo prueba aunque no se pueden guardar los proyectos y las funciones son un poco limitadas; para poder tener a disposición toda la herramienta es necesario comprar una licencia de uso. Esta herramienta permite la generación de código fuente, documentación, ingeniería inversa, entre otras funciones; características que lo hacen muy llamativo aunque cuenta con la limitante de ser software propietario.

### GWTUML

GWTUML es una herramienta open source desarrollada por la empresa francesa Object Direct, dirigido por el ODLabs departament. Esta es una compañía dedicada a la consultoría y el desarrollo de tecnologías de información utilizando metodologías de desarrollo ágil, orientación a objetos y Cloud computing, brindando soporte a proyectos open source.

GWTUML es una herramienta CASE en proceso de desarrollo, la cual permite el modelado visual de diagramas de clase, diagramas de objetos y diagramas de secuencia. El desarrollo de esta herramienta se inició en enero de 2009 y la versión actual aún se encuentra en modo experimental, sin embargo, los desarrolladores han liberado algunas APIs[[7]](#footnote-7) y versiones beta de la aplicación, las cuales pueden ser descargadas y utilizadas en otros proyectos que requieran sus servicios. Adicionalmente existe alguna documentación del código fuente (Javadocs) de modo que los usuarios pueden consultarlos en caso de utilizar las APIs disponibles.

Este proyecto está basado en GWT[[8]](#footnote-8) (Google Web Toolkit) el cual es un Framework para el desarrollo de aplicaciones web en lenguaje java, liberando al desarrollador de la complejidad implícita de varios aspectos de la tecnología AJAX, que a su vez es la herramienta más utilizada hoy en día para la construcción de aplicaciones para la internet

### JoinJS

JoinJS es una librería JavaScript para crear diagramas totalmente interactivos en la web. Esta librería puede ser usada tanto para implementar herramientas de diagramado o para simplemente publicar diagramas online.

Las principales características de esta librería son:

* Permite conectar objetos vectoriales con varios tipos de líneas y flechas.
* Interacción con objetos y conectores.
* Personalizar eventos y manejadores para mouse, teclado, entre otros.
* Elementos pre construidos para diagramas comúnmente usados (Entidad relación, UML, Org Chart, entre otros).
* Serialización (formato JSON, exportación a SVG en navegadores que lo permiten).
* Soporte para la mayoría de navegadores en versiones antiguas.

Página principal de la herramienta: [http://www.jointjs.com](http://www.jointjs.com/).

## MARCO CONCEPTUAL

### Herramientas CASE

Se puede definir a las Herramientas CASE (Computer Aided Software Engineering) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software [2].

La realización de un nuevo software requiere que las tareas sean organizadas y completadas en forma correcta y eficiente. Las Herramientas CASE fueron desarrolladas para automatizar esos procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo de software.

La mejor razón para la creación de estas herramientas fue el incremento en la velocidad de desarrollo de los sistemas. Por esto, las compañías pudieron desarrollar sistemas sin encarar el problema de tener cambios en las necesidades del negocio, antes de finalizar el proceso de desarrollo.

Las herramientas CASE también permiten a los analistas tener más tiempo para el análisis y diseño y minimizar el tiempo para codificar y probar.

Con un CASE integrado, las organizaciones pueden desarrollar rápidamente sistemas de mejor calidad para soportar procesos críticos del negocio y asistir en el desarrollo y promoción intensiva de la información de productos y servicios.

Estas herramientas pueden proveer muchos beneficios en todas las etapas del proceso de desarrollo de software, algunos de estos beneficios son:

* Verificar el uso de todos los elementos en el sistema diseñado.
* Automatizar el dibujo de diagramas.
* Ayudar en la documentación del sistema.
* Ayudar en la creación de relaciones en la Base de Datos.
* Generar estructuras de código.

La introducción de las herramientas CASE para ayudar en este proceso ha permitido que los diagramas puedan ser fácilmente creados y modificados, mejorando la calidad de los diseños de software. Los diccionarios de datos, un documento muy usado que mantiene los detalles de cada tipo de dato y los procesos dentro de un sistema, son el resultado directo de la llegada del diseño de flujo de datos y análisis estructural, hecho posible a través de las mejoras en las Herramientas CASE.

Pronto se reemplazaron los paquetes gráficos por paquetes especializados que habilitan la edición, actualización e impresión en múltiples versiones de diseño. Eventualmente, las herramientas gráficas integradas con diccionarios de base de datos para producir poderosos diseños y desarrollar herramientas, podrían sostener ciclos completos de diseño de documentos. Como un paso final, la verificación de errores y generadores de casos de pruebas fueron incluidos para validar el diseño del software. Todos estos procesos pueden saberse integrados en una simple herramienta CASE que soporta todo el ciclo de desarrollo.

La primera herramienta comercial se remonta a 1982, aunque algunos especialistas indican que algunos ejemplos de herramientas para diagramación ya existían.

No fue sino hasta 1985 en que las herramientas CASE se volvieron realmente importantes en el proceso de desarrollo de software. Los proveedores prometieron a la industria que muchas actividades serían beneficiadas por la ayuda de las CASE. Estos beneficios consistían, por ejemplo, en el aumento en la productividad. El objetivo en 1985 para muchos vendedores era producir software más rápidamente. Las herramientas del CASE serían una familia de métodos favorablemente estructurados para planeamiento, análisis y diseño. Esto llevaría a la generación automática de código para desarrollo de software vía una especificación formalmente diseñada.

Las herramientas CASE, en función de las fases del ciclo de vida abarcadas, se pueden agrupar de la forma siguiente:

1. Herramientas integradas, I-CASE (Integrated CASE, CASE integrado): abarcan todas las fases del ciclo de vida del desarrollo de sistemas. Son llamadas también CASE workbench.
2. Herramientas de alto nivel, U-CASE (Upper CASE - CASE superior) o Front-End, orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo: análisis y diseño.
3. Herramientas de bajo nivel, L-CASE (Lower CASE - CASE inferior) o Back-End, dirigidas a las últimas fases del desarrollo: construcción e implantación.
4. Juegos de herramientas o Tools-Case, son el tipo más simple de herramientas CASE. Automatizan una fase dentro del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento.

### UML

El UML (Lenguaje Unificado de Modelado) es una de las herramientas más emocionantes en el mundo actual de desarrollo de sistemas. Esto se debe a que permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas [3].

La comunicación de la idea es de suma importancia. Antes del advenimiento del UML, el desarrollo de sistemas era, con frecuencia, una propuesta al azar. Los analistas de sistemas intentaban evaluar los requerimientos de sus clientes, generar un análisis de requerimientos en algún tipo de notación que ellos mismos comprendieran (aunque el cliente no lo comprendiera), dar tal análisis a uno o más programadores y esperar que el producto final cumpliese con lo que el cliente deseaba.

Dado que el desarrollo de sistemas es una actividad humana, hay muchas posibilidades de cometer errores en cualquier etapa del proceso, por ejemplo, el analista pudo haber malentendido al cliente, es decir, probablemente produjo un documento que el cliente no pudo comprender. Tal vez ese documento tampoco fue comprendido por los programadores quienes, por ende, pudieron generar un programa difícil de utilizar y no generar una solución al problema original del cliente.

Hoy en día, es necesario contar con un plan bien organizado. Un cliente tiene que comprender que es lo que hará un equipo de desarrolladores; además tiene que ser capaz de señalar cambios si no se han captado claramente sus necesidades (o si cambia de opinión durante el proceso). A su vez, el desarrollo es un esfuerzo orientado a equipos, por lo que cada uno de sus miembros tiene que saber qué lugar toma su trabajo en la solución final (así como saber cuál es la solución general).

La clave está en organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. UML proporciona tal organización.

El UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson. Estos caballeros, apodados recientemente “Los tres amigos”, trabajaban en empresas distintas durante la década de los años ochenta y principios de los noventa y cada uno diseño su propia metodología para el análisis y diseño orientado a objetos. Sus metodologías predominaron sobre la de sus competidores. A mediados de los años noventa empezaron a intercambiar ideas entre si y decidieron desarrollar su trabajo en conjunto.

Los anteproyectos de UML empezaron a circular en la industria del software y las reacciones resultantes trajeron consigo considerables modificaciones. Conforme diversos corporativos vieron que el UML era útil a sus propósitos, se conformó un consorcio de UML. Entre los miembros se encuentran DEC, Hewlett-Packard, Intellicorp, Microsoft, Oracle, Texas Instruments y Rational. En 1997 el consorcio produjo la versión 1.0 del UML.

UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos entre los cuales encontramos: diagrama de clases, diagrama de objetos, diagrama de casos de uso, diagrama de estados, diagrama de secuencias, diagrama de actividades, diagrama de colaboraciones, diagrama de componentes y diagrama de distribución.

El estándar ofrece una variedad de diagramas, cada uno representa una vista del sistema en construcción, con diferente nivel de abstracción. La figura 2 muestra los diagramas que hacen parte de UML y la jerarquía o relación que existen entre cada uno de ellos.



Ilustración . Jerarquía de los diagramas UML [3]

### Diagramas de Clase

Los diagramas de clases muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema. Principalmente, esto incluye modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas. Los diagramas de clases también son la base para un par de diagramas relacionados, los diagramas de componentes y los diagramas de despliegue. Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino que también para construir sistemas ejecutables aplicando ingeniería directa e inversa.

Un diagrama de clases es un diagrama que muestra un conjunto de clases, interfaces, colaboraciones y sus relaciones. Al igual que otros diagramas los diagramas de clases pueden contener notas y restricciones. También pueden contener paquetes o subsistemas, los cuales se usan para agrupar los elementos de un modelo en partes más grandes. A veces se colocarán instancias en los diagramas de clases, especialmente cuando se quiera mostrar el tipo (posiblemente dinámico) de una instancia. El capítulo 5 está dedicado a explicar completamente tanto conceptos básicos como avanzados relacionados a los diagramas de clase.

### Meta-Programación

Meta-programación es el proceso o la práctica por la cual escribimos programas que generan, manipulan o utilizan otros programas. Por ejemplo, un compilador se puede pensar como un programa que genera otro programa, un formateador de código es un programa que manipula otro programa, o una herramienta como Javadoc utiliza nuestro programa para generar su documentación.

En general la Meta-programación se utiliza más fuertemente en el desarrollo de Frameworks, simplemente porque un Framework debe resolver cierta problemática de una aplicación, pero no va a estar diseñado para ninguna aplicación en particular. Es decir, la idea de Framework es que se pueda aplicar y utilizar en diferentes dominios desconocidos para el creador del mismo, entonces estos Frameworks manipulan objetos sin necesidad de conocerlos de antemano. Un ejemplo básico de un Framework es un ORM[[9]](#footnote-9) como Hibernate, el cual se encarga de persistir en base de datos, instancias de nuestras clases sin siquiera conocerlas de antemano.

Así como todo programa construye un modelo para describir su dominio, el dominio de un meta-programa es otro programa denominado programa objeto o base, y este tendrá a su vez, un modelo que describe dicho programa al que llamamos meta-modelo.

**Reflexión Computacional**

Reflexión es un caso particular de meta-programación, en donde “meta-programamos” en el mismo lenguaje en que están escritos (o vamos a escribir) los programas. Es decir todo desde el mismo lenguaje. Inicialmente el lenguaje pionero en cuanto a reflexión fue LISP.

El ejemplo más visible de esto es el caso de SmallTalk, donde no existe una diferenciación entre IDE y nuestro programa. Ambos están hechos en SmallTalk y de hecho viven en un mismo ambiente. Ambos construidos con objetos y pueden interactuar entre sí.

**Tipos de Reflexión:**

Reflexión también abarca los siguientes ítems que vamos a mencionar:

* Introspección: Se refiere a la capacidad de un sistema, de analizarse a sí mismo. Algo así como la introspección humana, pero en términos de software. Para eso, el lenguaje debe proveer ciertas herramientas que le permitan al mismo programa ver o reflejar cada uno de sus componentes.
* Auto modificación: Es la capacidad de un programa de modificarse a sí mismo. Nuevamente esto requiere cierto soporte del lenguaje y las limitaciones van a depender de este soporte.
* Intercesión: Es la capacidad de modificar la semántica del modelo que estamos manipulando desde el mismo lenguaje.

### Cloud Computing

La nube (Cloud Computing) es el procesamiento masivo de datos y almacenamiento de información en servidores, ubicados en cualquier parte del mundo,  conectados a una conexión de internet y a los que se puede acceder desde cualquier dispositivo: computador, Smartphone, tabletas, etc.

En los años 70’s John McCarthy, famoso científico e informático ganador del Premio Turing, expresó: “algún día la computación podrá ser organizada como un servicio público”. El primer uso académico de la expresión “computación en nube”, fue durante una conferencia de RamnathChellappa (actual profesor de la EmoryUniversity) en 1997. Desde ese momento el famoso concepto de la nube ha venido trascendiendo y hoy en día estamos entiendo con más claridad a qué se referían estos visionarios de la tecnología.

EL NIST (National Institute of Standards and Technology) de EE.UU lo define como: “Cloud Computing es un modelo para permitir el acceso adecuado y bajo demanda a un conjunto de recursos de cómputo configurables (ejemplos: redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente provistos y puestos a disposición del cliente con un mínimo esfuerzo de gestión y de interacción con el proveedor del servicio”.

Sin embargo, para algunos el concepto de nube sigue siendo confuso. Lo cierto es que muchas personas  ya realizan actividades diarias en la nube y lo desconocen. Cada vez que alguien consulta  su correo electrónico,  hace una transacción bancaria por Internet,  ve un video de su artista favorito en YouTube, sube fotos de una fiesta en Flickr, escucha música en  Grooveshark o  envía documentos por Dropbox, está  viviendo en la nube.

Para los usuarios finales, el cómputo cloud significa que no existen los costos de adquisición de hardware, ni el manejo de las licencias de software o de actualizaciones, ni nuevos empleados o consultores que contratar, ni instalaciones que rentar, ni costos de capital de ninguna clase — ni costos ocultos. Sólo una tarifa medida por uso o una cuota fija de subscripción. Use sólo lo que quiera, pague sólo lo que usa.

La siguiente figura muestra el modelo de servicios que pretende ofrecer Cloud Computing.



Ilustración . Modelo de servicios de Cloud Computing

**Modelos de servicios**

* Software como servicio: El software como servicio (en inglés software as a service, SaaS) se encuentra en la capa más alta y caracteriza una aplicación completa ofrecida como un servicio, en demanda, vía multi-tendencia que significa una sola instancia del software que corre en la infraestructura del proveedor y sirve a múltiples organizaciones de clientes. Un ejemplo es la plataforma MS Office como servicio SaaS con su denominación de Microsoft Office 365, que incluye versiones online de la mayoría de las aplicaciones de esta suite ofimática de Microsoft.
* Plataforma como servicio: La capa del medio, que es la plataforma como servicio (en inglés platform as a service, PaaS), es la encapsulación de una abstracción de un ambiente de desarrollo y el empaquetamiento de una serie de módulos o complementos que proporcionan, normalmente, una funcionalidad horizontal (persistencia de datos, autenticación, mensajería, etc.). De esta forma, un arquetipo de plataforma como servicio podría consistir en un entorno conteniendo una pila básica de sistemas, componentes o APIs pres configurados y listos para integrarse sobre una tecnología concreta de desarrollo (por ejemplo, un sistema Linux, un servidor web, y un ambiente de programación como Perl o Ruby). Un ejemplo de este tipo de nube es Google App Engine que sirven aplicaciones de Google y Microsoft.
* Infraestructura como servicio: La infraestructura como servicio (infrastructure as a service, IaaS) -también llamado en algunos casos hardware as a service, HaaS)6 se encuentra en la capa inferior y es un medio de entregar almacenamiento básico y capacidades de cómputo como servicios estandarizados en la red. Servidores, sistemas de almacenamiento, conexiones, enrutadores, y otros sistemas se concentran (por ejemplo a través de la tecnología de virtualización) para manejar tipos específicos de cargas de trabajo, desde procesamiento en lotes (batch) hasta aumento de servidor/almacenamiento durante las cargas pico.

**Tipos de nubes**

* Las nubes públicas se manejan por terceras partes, y los trabajos de muchos clientes diferentes pueden estar mezclados en los servidores, los sistemas de almacenamiento y otras infraestructuras de la nube. Los usuarios finales no conocen qué trabajos de otros clientes pueden estar corriendo en el mismo servidor, red, discos como los suyos propios.
* Las nubes privadas son una buena opción para las compañías que necesitan alta protección de datos y ediciones a nivel de servicio. Las nubes privadas están en una infraestructura en-demanda manejada por un solo cliente que controla qué aplicaciones debe correr y dónde. Son propietarios del servidor, red, y disco y pueden decidir qué usuarios están autorizados a utilizar la infraestructura.
* Las nubes híbridas combinan los modelos de nubes públicas y privadas. Usted es propietario de unas partes y comparte otras, aunque de una manera controlada. Las nubes híbridas ofrecen la promesa del escalado aprovisionada externamente, en-demanda, pero añaden la complejidad de determinar cómo distribuir las aplicaciones a través de estos ambientes diferentes. Las empresas pueden sentir cierta atracción por la promesa de una nube híbrida, pero esta opción, al menos inicialmente, estará probablemente reservada a aplicaciones simples sin condicionantes, que no requieran de ninguna sincronización o necesiten bases de datos complejas.

## MARCO TEORICO

### Java

El significado de Java tal y como se le conoce en la actualidad es el de un lenguaje de programación y un entorno para ejecución de programas escritos en el lenguaje Java. AI contrario que los compiladores tradicionales, que convierten el código fuente en instrucciones a nivel de máquina, el compilador Java traduce el código fuente Java en instrucciones que son interpretadas por la Máquina Virtual Java (JVM, *Java Virtual Machine).* A diferencia de los lenguajes C y C++ en los que está inspirado, Java es un lenguaje interpretado. Aunque hoy en día Java es por excelencia el lenguaje de programación para Internet y la World Wide Web en particular, Java no comenzó como proyecto Internet y por esta circunstancia es idóneo para tareas de programación de propósito general y, de hecho, muchas de las herramientas Java están escritas en Java[[10]](#footnote-10).

La independencia de plataforma es una de las razones por las que Java es interesante para Internet, ya que muchas personas deben tener acceso con ordenadores distintos. Pero no se queda ahí, Java está desarrollándose incluso para distintos tipos de dispositivos además del ordenador como móviles, agendas y en general para cualquier cosa que se le ocurra a la industria.

**Java Virtual Machine (JVM)**

La JVM es una de las piezas fundamentales de la plataforma Java. Básicamente se sitúa en un nivel superior al Hardware del sistema sobre el que se pretende ejecutar la aplicación, y este actúa como un puente que entiende tanto el bytecode, como el sistema sobre el que se pretende ejecutar. Así, cuando se escribe una aplicación Java, se hace pensando que será ejecutada en una máquina virtual Java en concreto, siendo ésta la que en última instancia convierte de código bytecode a código nativo del dispositivo final.

La gran ventaja de la máquina virtual java es aportar portabilidad al lenguaje de manera que desde Sun Microsystems se han creado diferentes máquinas virtuales java para diferentes arquitecturas y así un programa .class escrito en un Windows puede ser interpretado en un entorno Linux. Tan solo es necesario disponer de dicha máquina virtual para dichos entornos.

**Componentes Java**

*J2EE* es un grupo de especificaciones diseñadas por Sun que permiten la creación de aplicaciones empresariales, esto sería: acceso a base de datos (JDBC), utilización de directorios distribuidos (JNDI), acceso a métodos remotos (RMI/CORBA), funciones de correo electrónico (Java Mail). Java EE también configura algunas especificaciones únicas para componentes Java EE, estas incluyen: Enterprise JavaBeans EJB, Servlets, Portlets (siguiendo la especificación de Portlets Java), Java Server Pages y varias tecnologías de servicios web. Ello permite al desarrollador crear aplicaciones empresariales portables entre plataformas y escalables, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel[[11]](#footnote-11).

*J2SE* es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java. La Plataforma Java 2 Enterprise Edition incluye todas las clases en el Java SE, además de algunas de las cuales son útiles para programas que se ejecutan en servidores sobre estaciones de trabajo.

*J2ME* es la versión de Java orientada a los dispositivos móviles. Debido a que los dispositivos móviles tienen una potencia de cálculo baja e interfaces de usuario pobres, es necesaria una versión específica de Java destinada a estos dispositivos, ya que el resto de versiones de Java, J2SE o J2EE, no encajan dentro de este esquema. J2ME es por tanto, una versión “reducida” de J2SE.

### Frameworks

En el desarrollo de software, un Framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Muchas veces a los Frameworks se les tiende a dar el mismo nombre de Patrón de diseño, pensando que hacen referencia al mismo concepto, sin embargo esto no es cierto y de hecho tienen significados diferentes; un Framework representa código implementado, mientras que un patrón de diseño representa conocimiento y experiencia.

Con el crecimiento exponencial de Internet en los últimos años, las aplicaciones Web se han convertido en una parte básica y común dentro del desarrollo de software; estas han acaparado la atención no solo de las empresas que desean formar parte de este nuevo mundo, sino también de aquellas que se han dedicado a las herramientas de desarrollo de software.

Dentro de este tipo de aplicaciones, Java juega un papel muy importante actualmente ya que este es uno de los usos más comunes que se le da a este lenguaje de programación, además de que representa uno de los mejores medios para construir dichas aplicaciones.

Aunque no existe una clasificación estándar y formal que se aplique a los Frameworks para aplicaciones web, existen algunos aspectos que se pueden considerar para diferenciar unos de otros.

De esta manera y teniendo en cuenta el enfoque de los Frameworks, se pueden mencionar dos tipos para el lenguaje Java:

* **Frameworks de aplicación**: Los Frameworks de aplicación son aquellos que están basados en los request HTTP, es decir hacen uso del API Servlet. Asi mismo, estos Frameworks no se preocupan acerca de los detalles de cómo se renderiza la interfaz de usuario y no hacen distinción entre acciones que solo afectan a la interfaz de usuario y aquellas que requieren ser procesadas por código de la aplicación.
* **Frameworks de Interfaz de usuario**: Estos Frameworks están basados en componentes, se enfocan en los detalles de la interfaz de usuario y no se interesan en cómo es implementado el resto de la aplicación. De igual forma, definen un API detallado para los componentes que formaran la interfaz de usuario con los objetivos de ligar a estos con la lógica de la aplicación adecuada, determinar qué acciones de los usuarios resultaran eventos de la interfaz y como serán manejados estos últimos, entre otras finalidades.

De entre todos los Frameworks para el desarrollo en Java, sobresalen los siguientes:

* **Struts**: Framework de aplicación open source, este fue el primer Framework en salir para el desarrollo web en Java.
* **Java Server Faces (JSF)**: Framework de interfaz de usuario, es el estándar de Java actualmente para el desarrollo web.
* **Spring**: Framework de aplicación muy popular últimamente.

### Servidor de Aplicaciones

En informática, se denomina servidor de aplicaciones a un servidor en una red de computadores que ejecuta ciertas aplicaciones. Usualmente se trata de un dispositivo de software que proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

También conocidos como Middleware, el cual es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos. Éste simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones que son necesarias en los sistemas distribuidos. De esta forma se provee una solución que mejora la calidad de servicio, seguridad, envío de mensajes, directorio de servicio, etc.

Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red). El middleware abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API para la fácil programación y manejo de aplicaciones distribuidas. Dependiendo del problema a resolver y de las funciones necesarias, serán útiles diferentes tipos de servicios de middleware. Por lo general el middleware del lado cliente está implementado por el Sistema Operativo, el cual posee las bibliotecas que ejecutan todas las funcionalidades para la comunicación a través de la red.

En Java estos son llamados contenedores JEE, los cuales proveen un entorno de ejecución de componentes Java. Tambien brindan servicios de seguridad, transacciones, administración de ciclo de vida, caching, persistencia y comunicación en la red. Existen dos tipos de contenedores JEE:

* **Contenedores Web**: Almacenan componentes de presentación (JSPs, Servlets). Un ejemplo claro de este tipo de contenedor es Apache Tomcat, el cual permite el procesamiento de este tipo de componentes.
* **Contenedores EJB**: Adicional a los componentes permitidos por los contenedores web, los contenedores EJB permiten la ejecución de componentes EJB (Enterprise Java Beans). Dentro de este tipo de software se encuentran herramientas como Glassfish, JBoss AS y WebLogic.

### Extreme Programming (XP)

Metodología ágil basada en cuatro principios: simplicidad, comunicación, retroalimentación y valor. Además, orientada por pruebas y refactorización, se diseña e implementan las pruebas antes de programar la funcionalidad, el programador crea sus propias Pruebas de Unidad[[12]](#footnote-12).

Este método es típicamente atribuido a Kent Beck, Ron Jeffries y Ward Cinningham. El objetivo de XP son grupos pequeños y medianos de construcción de software en donde los requisitos aún son muy ambiguos, cambian rápidamente o son de alto riesgo. XP busca la satisfacción del cliente tratando de mantener durante todo el tiempo su confianza en el producto. Además, sugiere que el lugar de trabajo sea una sala amplia, si es posible sin divisiones (en el centro los programadores, en la periferia los equipos individuales). Una ventaja del espacio abierto es el incremento en la comunicación y el proporcionar una agenda dinámica en el entorno de cada proyecto [4].

**Actividades de XP**

* **Codificar**: Es necesario codificar y plasmar nuestras ideas a través del código. En programación, el código expresa la interpretación del problema, así podemos utilizar el código para comunicar, para hacer comunes las ideas, y por tanto para aprender y mejorar.
* **Hacer pruebas**: Las características del software que no pueden ser demostradas mediante pruebas simplemente no existen. Las pruebas dan la oportunidad de saber si lo implementado es lo que en realidad se tenía en mente. Las pruebas nos indican que nuestro trabajo funciona, cuando no podemos pensar en ninguna prueba que pudiese originar un fallo en nuestro sistema, entonces habremos acabado por completo.
* **Escuchar**: Si vamos a hacer pruebas tenemos que preguntar si lo obtenido es lo deseado, y tenemos que preguntar a quién necesita la información. Tenemos que escuchar a nuestros clientes cuáles son los problemas de su negocio, debemos de tener una escucha activa explicando lo que es fácil y difícil de obtener, y la realimentación entre ambos nos ayudan a todos a entender los problemas.
* **Diseñar**: El diseño crea una estructura que organiza la lógica del sistema, un buen diseño permite que el sistema crezca con cambios en un solo lugar. Los diseños deben de ser sencillos, si alguna parte del sistema es de desarrollo complejo, lo apropiado es dividirla en varias. Si hay fallos en el diseño o malos diseños, estos deben de ser corregidos cuanto antes.

**Ciclo de Vida de XP**

El ciclo de vida de XP se enfatiza en el carácter interactivo e incremental del desarrollo, Según [1] una iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas que en el caso de XP corresponden a un conjunto de historias de usuarios.

Las iteraciones son relativamente cortas ya que se piensa que entre más rápido se le entreguen desarrollos al cliente, más retroalimentación se va a obtener y esto va a representar una mejor calidad del producto a largo plazo. Existe una fase de análisis inicial orientada a programar las iteraciones de desarrollo y cada iteración incluye diseño, codificación y pruebas, fases superpuestas de tal manera que no se separen en el tiempo.

La siguiente figura muestra las fases en las que se subdivide el ciclo de vida XP:

**

Ilustración . Ciclo de vida de Extreme Programming

### SVG

Los Gráficos Vectoriales Redimensionables (del inglés Scalable Vector Graphics) o SVG son una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL), en formato XML.

SVG se convirtió en una recomendación del W3C en septiembre de 2001, por lo que ya ha sido incluido de forma nativa en el navegador web del W3C Amaya. Las versiones 1.5 y posteriores de Mozilla Firefox soportan gráficos hechos con SVG, así como el navegador Opera que desde su versión 8 ha implementado SVG 1.1 Tiny en su núcleo. Navegadores como Google Chrome, Safari e Internet Explorer 9 también son capaces de mostrar imágenes en formato SVG sin necesidad de complementos externos. Otros navegadores web, como versiones anteriores a la 9 de Internet Explorer, necesitan un conector o plug-in, para lo que se puede utilizar.

El SVG permite tres tipos de objetos gráficos:

* Elementos geométricos vectoriales (por ejemplo caminos consistentes en rectas y curvas, y áreas limitadas por ellos)
* Imágenes de mapa de bits /digitales
* Texto

Los objetos gráficos pueden ser agrupados, transformados y compuestos en objetos previamente renderizados, y pueden recibir un estilo común. El texto puede estar en cualquier espacio de nombres XML admitido por la aplicación, lo que mejora la posibilidad de búsqueda y la accesibilidad de los gráficos SVG. El juego de características incluye las transformaciones anidadas, los clipping paths, las máscaras alfa, los filtros de efectos, las plantillas de objetos y la extensibilidad.

El dibujado de los SVG puede ser dinámico e interactivo. El Document Object Model (DOM) para SVG, que incluye el DOM XML completo, permite animaciones de gráficos vectoriales sencillos y eficientes mediante ECMAScript o SMIL. Un juego amplio de manejadores de eventos, como "onMouseOver" y "onClick", pueden ser asignados a cualquier objeto SVG. Debido a su compatibilidad y relación con otras normas Web, características como el scripting pueden ser aplicadas a elementos SVG y a otros elementos XML desde distintos espacios de nombre XML simultáneamente dentro de la misma página web. Un ejemplo extremo de esto es un juego completo de tetris realizado como un objeto SVG.

Si el espacio de almacenamiento es un problema, las imágenes SVG pueden salvarse comprimidas con gzip, en cuyo caso pasan a ser imágenes SVGZ. Debido a la verbosidad del XML, este tiende a comprimirse muy bien, y estos ficheros pueden ser mucho más pequeños. Aun así, a menudo el fichero vectorizado original (SVG) es más pequeño que la versión de mapa de bits. A pesar de ser un lenguaje vectorial, SVG permite crear imágenes complejas.

### HTML, JavaScript y CSS

Originalmente, las páginas HTML sólo incluían información sobre sus contenidos de texto e imágenes. Con el desarrollo del estándar HTML, las páginas empezaron a incluir también información sobre el aspecto de sus contenidos: tipos de letra, colores y márgenes.

La posterior aparición de tecnologías como JavaScript, provocaron que las páginas HTML también incluyeran el código de las aplicaciones (llamadas scripts) que se utilizan para crear páginas web dinámicas.

Incluir en una misma página HTML los contenidos, el diseño y la programación complica en exceso su mantenimiento. Normalmente, los contenidos y el diseño de la página web son responsabilidad de diferentes personas, por lo que es conveniente separarlos.

CSS es el mecanismo que permite separar los contenidos definidos mediante XHTML y el aspecto que deben presentar esos contenidos.

Una ventaja añadida de la separación de los contenidos y su presentación es que los documentos XHTML creados son más flexibles, ya que se adaptan mejor a las diferentes plataformas: pantallas de ordenador, pantallas de dispositivos móviles, impresoras y dispositivos utilizados por personas discapacitadas.

De esta forma, utilizando exclusivamente XHTML se crean páginas web "feas" pero correctas. Aplicando CSS, se pueden crear páginas "bonitas" a partir de las páginas XHTML correctas.

El código JavaScript se encierra entre etiquetas <script> y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código JavaScript dentro de la cabecera del documento (dentro de la etiqueta <head>). Las instrucciones JavaScript tambien se pueden incluir en un archivo externo de tipo JavaScript que los documentos XHTML enlazan mediante la etiqueta <script>. Se pueden crear todos los archivos JavaScript que sean necesarios y cada documento XHTML puede enlazar tantos archivos JavaScript como necesite.

## MARCO DE LEGAL

### Acuerdos de uso

El software, resultado de la ejecución del proyecto, se regirá bajo los marcos legales fijados por la licencia pública Creative Common (LPCC) que dicta las siguientes condiciones de uso, encontradas más detalladamente en el documento Atribución No Comercial Compartir Igual 2.5 (Colombia) [6]:

* Se posee la libertad de compartir la obra, entendiendo compartir como la capacidad de copiar, distribuir, ejecutar y comunicar públicamente la obra.
* Además puede llevar a cabo obras derivadas de la original.
* Es obligatorio reconocer los créditos de la obra de manera especificada por el autor o el licenciante, pero no de manera que sugiera que tiene el apoyo de este último, o que apoya el uso que hacen a su obra.
* No puede ser utilizada para uso comercial.
* Si se altera o transforma, o genera un derivado a partir de esta obra, solo puede ser distribuida bajo una licencia idéntica a la presente.

### Contrato de licencia de código binario, SUN MICROSYSTEMS

Licencia de uso de software concedidos por Sun Microsystems, para la utilización de Java SE, siempre y cuando sean aceptados los términos de licencia, expuestos en el documento SUN MICROSYSTEMS, INC. CONTRATO DE LICENCIA DE CÓDIGO BINARIO [7].

“SUN MICROSYSTEMS, INC. (EN ADELANTE DENOMINADO “SUN”) LE CONCEDE LA LICENCIA DEL SOFTWARE DEFINIDO A CONTINUACIÓN ÚNICAMENTE CON LA CONDICIÓN DE QUE USTED ACEPTE TODOS LOS TÉRMINOS ESTIPULADOS EN EL PRESENTE CONTRATO DE LICENCIA DE CÓDIGO BINARIO Y TÉRMINOS DE LICENCIA ADICIONALES (EN CONJUNTO DENOMINADOS “CONTRATO”). POR FAVOR, LEA EL CONTRATO DETENIDAMENTE. EL USO DEL SOFTWARE SIGNIFICA QUE HA LEÍDO LAS CONDICIONES Y QUE LAS ACEPTA. SI ACEPTA ESTAS CONDICIONES EN REPRESENTACIÓN DE UNA COMPAÑÍA U OTRA ENTIDAD LEGAL, SIGNIFICA QUE ESTÁ EN POSESIÓN DE LA AUTORIDAD QUE LE PERMITE VINCULAR LA ENTIDAD LEGAL A ESTAS CONDICIONES. SI NO ESTÁ EN POSESIÓN DE DICHA AUTORIDAD O SI NO DESEA QUEDAR VINCULADO A ESTAS CONDICIONES, NO DEBE UTILIZAR EL SOFTWARE EN ESTA INSTALACIÓN NI EN NINGÚN OTRO SOPORTE EN EL QUE SE UBIQUE EL SOFTWARE”.

### GNU General Public Licence

La Licencia Pública General de GNU (GNU GPL, por sus siglas en inglés) es una licencia libre y gratuita con derecho de copia para software y otros tipos de obras.

Las licencias para la mayoría del software y otras obras de índole práctica están diseñadas para privarle de la libertad para distribuir y modificar las obras. Por el contrario, la Licencia Pública General de GNU garantiza la libre distribución y modificación de todas las versiones de un programa, a fin de asegurarle dicha libertad a todos los usuarios. En la Fundación para el Software Libre utilizamos la Licencia Pública General de GNU para la mayoría de nuestro software; también se aplica a cualquier otra obra publicada de esta manera por sus autores. Usted también puede aplicarla a sus programas.

Cuando hablamos de software libre, nos referimos a la libertad, no al precio. Nuestras Licencias Públicas Generales están diseñadas para garantizarle a usted la libertad de distribuir copias de software libre (y cobrar por ellas, si así lo desea), obtener el código fuente, o tener la posibilidad de obtenerlo, modificar el software o utilizar partes del mismo en nuevos programas libres, y saber que puede hacer estas cosas [8].

### Propiedad Intelectual

De acuerdo con lo estimulado en el artículo 156 del acuerdo 065 del 26 de 1996, correspondiente al estatuto estudiante de la Universidad Francisco de Paula Santander, el cual dicta de la siguiente manera: “Los trabajos de grado son propiedad intelectual de la Universidad y su uso estará sujeto a las normas que para tal fin estén vigentes”[[13]](#footnote-13).

# [DISEÑO METODOLÓGICO](#TablaContenido)

El diseño metodológico del proyecto está enmarcado en la investigación aplicada y se fundamentará en la ingeniería de software, de ella se tomarán los conceptos, metodologías y técnicas propias para la construcción de la aplicación.

## TIPO DE INVESTIGACIÓN

Según la problemática planteada, el proyecto pretende dar solución a una necesidad vista dentro del programa de Ingeniería de Sistemas de la UFPS, de tal manera que se englobará el proyecto como una INVESTIGACIÓN APLICADA en términos del desarrollo de software que permitan solventar necesidades existentes dentro de la organización.

## FUENTES DE INFORMACIÓN

### Fuentes de información primaria

Teniendo en cuenta que la problemática afecta a la comunidad perteneciente al programa de Ingeniería de Sistemas, se consideran las siguientes fuentes de información primaria:

* Estudiantes que están cursando las asignaturas, “Programación Orientada a Objetos” y “Análisis y diseño de Sistemas”, las cuales se enfocan principalmente en el modelado de aplicaciones. Estos son cursos de 2º y 6º semestre respectivamente según el pensum 115 [9]. No obstante, a manera de observación indirecta se tomaran otros cursos que puedan usar el producto en construcción.
* Docentes del programa Ingeniería de Sistemas, los cuales son fuente de información de vital importancia, ya que ellos son los que acompañan el proceso de aprendizaje de los estudiantes.

### Fuentes de información secundaria

Estas fuentes hacen referencia a estudios, técnicas, teorías, documentos y toda clase de información existente sobre el tema en particular de modo que se establecen las siguientes:

* Herramientas CASE existentes, comerciales o de código abierto, que permitan el diseño de diagramas de clase. Se tomará en cuenta las herramientas utilizadas actualmente por los estudiantes, de modo que se pueda determinar que funcionalidad o que servicios ofrecen.
* Documento de Especificación del estándar UML para Diagramas de Clase.
* Documentación de proyectos de software online que permitan diseñar diagramas de clase.

Libros enfocados en temas como: Cloud computing, Ingeniería de Software, Lenguaje Unificado de Modelado, Procesos de desarrollo de software, desarrollo de aplicaciones Web, desarrollo de aplicaciones con J2EE, AJAX.

## RECOLECCIÓN Y ANALISIS DE INFORMACIÓN

El desarrollo del proyecto requiere obtener información general sobre la construcción de diagramas de clase, esto implica conocer a fondo conceptos tales como: Clases, Interfaces, Relaciones, Navegabilidad, entre otros; además de todos los elementos de los cuales dispone UML para el diseño de este tipo de diagramas. Para esto se realizará una investigación profunda sobre el tema, consultado bibliografía existente y diferentes autores.

Adicionalmente se necesita determinar las funcionalidades básicas que poseen las herramientas CASE utilizadas actualmente por los estudiantes de la UFPS, de manera que se pueda conocer qué procedimientos, servicios y características poseen con respecto al diseño de diagramas de clase; esto se hace con el objetivo de determinar los requerimientos de funcionalidad mínimos que debe poseer el software en construir y los servicios que este debe ofrecer. De cierta manera, mediante observación indirecta y a manera de tradición se conocen las principales herramientas CASE utilizadas por estudiantes y docentes de Ingeniería de Sistemas de la UFPS (Enterprise Architect, ArgoUML, StarUML), y considerando que el programa académico dispone de algunas licencias de uso para este tipo de herramientas (como es el caso de Enterprise Architect), se procederá a realizar un análisis concreto de estas herramientas específicamente. De igual manera y como se mencionó anteriormente, este análisis servirá como punto de partida del proyecto y además ayudará a tener un idea clara de la aplicación que se desea llegar a construir.

El procedimiento utilizado para la recolección de información en este caso será de manera literaria, mediante consultas e investigaciones de material bibliográfico, logrando una apropiación de conceptos y técnicas importantes para la construcción del proyecto.

Por último es necesario aclarar que este proyecto no pretende dar respuestas a incógnitas o hipótesis que se tengan sobre algún fenómeno académico o social dentro de la Universidad, sino que se encargara de suplir una necesidad detectada dentro del establecimiento.

Para el análisis de la información se procederá a clasificar los datos según su naturaleza, determinando los aspectos importantes para cada fase del desarrollo y la información necesaria para construir los componentes en cada una de ellas.

El análisis de la información consistirá en evaluar los datos obtenidos de las fuentes mencionadas, documentación de las herramientas CASE existentes, manuales de usuario, análisis del funcionamiento de las aplicaciones utilizadas actualmente, literatura especifica del caso, y en general, fuentes que sean cercanas al tema. Se realizará un proceso de clasificación y agrupamiento de los datos, de manera que se pueda mediante ellos, apoyar la siguiente fase del desarrollo y que sean las directrices de todo el proyecto en general, de modo que mediante estos se puedan tomar decisiones importantes para el desarrollo.

# ANALISIS DE LAS HERRAMIENTAS CASE

Con el objetivo de obtener información importante para la realización del proyecto y con el ánimo de iniciar el proceso de levantamiento de requerimientos se ha realizado un pequeño análisis de algunas de las herramientas utilizadas dentro del plan de estudios de Ingeniería de Sistemas para realizar modelado UML, de modo que se pueda establecer cuáles son las principales características y servicios que estas ofrecen a los usuarios en cuanto al diseño de *Diagramas de Clase* y que puedan ser incluidos en las funcionalidades del software en construcción.

De modo aleatorio se han seleccionado dos de las herramientas más conocidas y que a manera de observación se ha determinado que son ampliamente utilizadas por estudiantes y docentes del plan de estudios, adicionalmente se ha seleccionado una herramienta similar a la que está en construcción, de modo que podamos conocer qué características ofrece a los usuarios y cuales son imprescindibles en una herramienta de modelamiento UML. Las herramientas seleccionadas son: *Enterprise Architect* que es un software comercial conocido a nivel mundial por su gran calidad, *StarUML* que a su vez es open source pero ampliamente utilizado, y por ultimo una herramienta de modelado online llamada *Draw.IO* la cual se asemeja mucho a lo que estamos tratando de lograr.

Al momento de realizar este análisis se tomó la última versión de las aplicaciones que se instalan de manera nativa, Enterprise Architect en su versión 9.0.9 con un periodo de prueba de 30 días y StarUML en su versión 5.0. En el caso de Draw.IO la librería mxGraph usada para la construcción de esta aplicación se encuentra en la versión 1.10.4.0.

A continuación se describirá cada uno de ellos, resaltando en detalle cada una de sus características:

## ENTERPRISE ARCHITECT

Enterprise Architect de Sparx Systems es una herramienta CASE (Computer Aided Software Engineering) para el diseño y construcción de sistemas de software, para el modelado de procesos de negocios, y para objetivos de modelado más generalizados.

EA es una herramienta de análisis y diseño UML comprensivo, que cubre el desarrollo de software desde la obtención de los requisitos, diseño del modelo, pruebas, cambio de control y mantenimiento para la implementación, con completa trazabilidad. EA combina el poder de la última especificación UML 2.4.1 con alta performance, una interfaz intuitiva, para brindar un modelado avanzado a todo el equipo de desarrollo. Es una herramienta visual con multi-usuario con un gran establecimiento de características ayudando a los analistas, testers, administradores de proyectos, personal del control de calidad y desarrolladores alrededor del mundo a construir y documentar, software sostenible.

### Historia

Sparx Systems es una empresa australiana dedicada a la creación de herramientas para el modelado UML, fundada en 1996 por Geoffrey Sparx y situada en Creswick una pequeña población de Victoria, Australia. La primera versión de Enterprise Architect fue liberada en el año 2000 y fue originalmente diseñada como una herramienta de modelado UML para la versión 1.1 del estándar, de allí en adelante el producto ha evolucionado para soportar otras especificaciones OMG[[14]](#footnote-14) UML como 1.3, 2.0, 2.1, 2.3 y 2.4.1.

Durante el tiempo de vida de la aplicación se han liberado alrededor de 20 versiones, desde la versión 1.0 hasta la más reciente 9.3 liberada en Agosto de 2012, cada una con mejoras y características importantes según los cambios que ha sufrido el estándar UML y la adopción de algunas tecnologías

Actualmente la cantidad de estándares para el diseño y modelado de software y sistemas de negocios que son soportados por Enterprise Architect es considerable, teniendo en cuenta la adopción de UML 2.4.1 como principal herramienta para modelar aplicaciones e incluyendo algunos otros BPMN, WSDL, BPEL y XSD, ha permitido que la herramienta se imponga como uno de los líderes del mercado y que su aceptación por parte de la industria del software haya sido grandiosa.

Enterprise Architect se encuentra disponible en diferentes ediciones, según las necesidades del usuario o de las organizaciones que usan la aplicación, de modo que se pueden adquirir diferentes tipos de licencias para diferentes usos. Actualmente existen dos ediciones y dentro de las cuales se pueden comprar varias versiones del producto:

* Ediciones base
* Corporativa
* Profesional
* Escritorio
* Ediciones Suite
* Ultimate
* Ingeniería de Sistemas
* Ingeniería de Software y negocios

Los precios del producto dependen de la versión adquirida y de la cantidad de licencias que se obtengan de cada uno, siendo la versión de Escritorio la más simple hasta la versión Ultimate que a su vez posee la capacidad de trabajo en múltiples dominios de manera remota y un soporte completo de ingeniería de software y de negocios.

### Características Principales

Enterprise Architect posee un gran abanico de características, todas encaminadas al desarrollo y mantenimiento de proyectos de software, aunque también brinda soporte para otro tipo de proyectos.

La información aquí consignada fue tomada de la documentación explicita de Enterprise Architect, disponible en la página web, además se usó parte de la información del “Contenido de Ayuda” que viene incluido en la versión del software.

|  |  |
| --- | --- |
| **Característica** | **Descripción** |
| Plataforma | Diseñado para ser usado bajo sistemas operativos Windows, sin embargo puede ser instalado y utilizado mediante emuladores como CrossOver en sistemas Linux o MAC. CrossOver es una herramienta propietaria basada en una tecnología open source llamada Wine, el cual es una API Win32 re-desarrollada para sistemas operativos basados en Unix como Mac OS X o Linux. |
| Categoría de Software | Software propietario distribuido bajo licencias comerciales, el costo de estas depende de la versión del software y la cantidad de licencias adquiridas. |
| Versión de UML | UML 2.4.1 Liberada en agosto de 2011. |
| Diagramas UML | Soporte para todos los tipos de diagramas UML tanto estructurales como comportamentales. Algunos diagramas adicionales no UML: Análisis, Requerimientos, Bases de Datos, Mantenimiento, Interfaces de Usuario, Interacción y modelos del negocio. |
| Tecnologías MDG | Soporta tecnologías como ICONIX, ArcGIS, BPMN, SMOF, SoaML, SysML. |
| Fases del desarrollo | Modelamiento del Negocio, Ingeniería de Requerimientos, Análisis, Diseño, Implementación, Pruebas, Mantenimiento, Gestión de la Calidad. |
| Estándar XMI | Soporte completo del estándar en su versión 2.1, permite exportar e importar proyectos en archivos .XMI de modo que puedan ser llevados a otras herramientas de modelado. |
| Proyectos y equipos de trabajo | Permite generar repositorios de servidor DBMS , de modo que se pueda trabajar de manera concurrente en proyectos de gran envergadura y con gran cantidad de usuarios. Adicionalmente permite integración con repositorios de control de versiones como Subversion, CVS y SCC. |
| Importación / Exportación | Exportación: Existen varias formas de exportar proyectos, diagramas o cualquier otro tipo de artefacto creado dentro del software, archivos XML, XMI, Código fuente, imágenes planas de los diagramas en formato PNG, JPG, GIF, entre otros. Importación: Desde XML, XMI, archivos CSV, desde código fuente en varios lenguajes, desde archivos binarios como JARs o ensamblados .NET. |
| Ingeniería Directa e Inversa | Enterprise Architect soporta la generación directa e ingeniería reversa de código fuente para muchos lenguajes populares incluyendo C++, C#, Java, Delphi, VB.Net, Visual Basic, ActionScript, Python y PHP. Con un editor de código fuente “resaltador de sintaxis” integrado. Enterprise Architect le permite rápidamente navegar y explorar su código fuente del modelo en el mismo ambiente. Las plantillas de generación de código le permiten personalizar el código fuente generado de acuerdo a las especificaciones de su compañía. |
| Documentación generada | Soporte completo de WYSIWYG manejador de plantillas para generar archivos RTF. |
| Soporte de Versiones | Soporte para servidores Subversion SVN. |

Tabla . Características principales de Enterprise Architect

### Funciones para Diagramas de Clase

El diagrama de Clases captura la estructura lógica del sistema - las clases y cosas que constituyen el modelo -. Es un modelo estático, describiendo lo que existe y qué atributos y comportamiento tiene, más que cómo se hace algo. Los diagramas de Clases son los más útiles para ilustrar las relaciones entre las clases e interfaces. Las generalizaciones, las agregaciones y las asociaciones son todas valiosas para reflejar la herencia, la composición o el uso y las conexiones respectivamente.

El diagrama de abajo ilustra las relaciones de agregación entre clases. La agregación con la punta de flecha en color claro indica que la clase Cuenta es usada por LibroDeDirecciones, pero que no está contenida necesariamente. La agregación con la punta de flecha en color oscuro indica la posesión o la contención de las clases destino (en el extremo del rombo) por las clases origen.



Ilustración . Ejemplo de diagrama de clase Enterprise Architect

**Caja de herramientas**

La Caja de herramientas de UML de Enterprise Architect se usa para crear elementos y conectores en un diagrama. Dentro de la Caja de herramientas, los elementos relativos de UML y conectores están organizados en páginas, cada página conteniendo los elementos o conectores para un tipo de diagrama en particular. Los diagramas incluyen los diagramas estándares de UML, los diagramas extendidos de Enterprise Architect, y cualquier Tecnología MDG o Perfil UML que usted haya agregado a Enterprise Architect.

Los componentes en la caja de herramientas se pueden arrastrar y soltar (Drag & Drop) al lienzo principal del diseñador. En cuanto a diagramas de clase la caja de herramientas muestra el listado de componentes específico para este tipo de diagrama. La siguiente imagen muestra la caja de herramientas para diagramas de clase.



Ilustración . Caja de herramientas para diagramas de clase Enterprise Architect

**Explorador de proyecto**

La ventana del Explorador del proyecto permite navegar a través del espacio de proyectos de Enterprise Architect. Muestra paquetes, diagramas, elementos y propiedades de los elementos.

Los proyectos pueden contener diferentes tipos de modelos y a su vez estos contienen diagramas, los diagramas son agrupaciones de elementos los cuales contienen propiedades.

Puede arrastrar y soltar elementos entre carpetas, o incluso soltar elementos desde el Explorador del proyecto directamente dentro del diagrama actual.

Si hace clic con el botón derecho del mouse sobre un ítem en el Explorador del Proyecto, puede llevar a cabo acciones adicionales, tales como agregar nuevos paquetes, crear diagramas, renombrar ítems, crear documentación y otros reportes y eliminar elementos del modelo.



Ilustración . Explorador de proyecto Enterprise Architect

**Panel de propiedades**

La ventana Propiedades provee una forma conveniente de ver (y en algunos casos editar) las propiedades comunes de los elementos. Cuando se selecciona un elemento, la pestaña de Propiedades mostrará el nombre, estereotipo, versión, autor, fechas, y otra información pertinente del elemento.



Ilustración . Panel de propiedades de Enterprise Architect

**Trabajar con elementos**

Los Modelos UML se construyen desde los elementos, cada uno de los cuales tiene su propio significado, reglas y notación. Los elementos se pueden usar en diferentes estados del proceso de diseño para distintos propósitos.

Los elementos básicos para diagramas de clase UML 2.1 se describen a continuación:



Ilustración . Elementos de diagramas de clase Enterprise Architect

La anterior imagen contiene los diferentes elementos que Enterprise Architect permite para el diseño de diagramas de clase. Esta contiene los elementos: Paquete, Clase, Interfaz, Objeto, Tabla y Asociación.

Haciendo clic con el botón derecho del mouse sobre un elemento del diagrama se abre el menú contextual del mismo. Si selecciona dos o más elementos, se mostrará un menú contextual de selección múltiple.

El menú contextual del elemento está dividido en un número secciones y submenús:

* Propiedades
* Agregar
* Buscar
* Transforme - permite que Transforme el elemento seleccionado de un dominio a otro.
* Elementos embebidos
* Características
* Generar DDL - Genera DDL para una tabla, procedimiento o Clase
* Ingeniería de código
* Apariencia
* Ayuda UML - abre el tema de ayuda de Enterprise Architect para el tipo de elemento UML.
* Eliminar - puede eliminar el elemento desde la opción menú.



Ilustración . Menú contextual de una Clase en Enterprise Architect

De este menú contextual podemos resaltar las opciones de Atributos, Operaciones, Generación de Código, Sincronizar con código y Ver código, estas opciones permiten al diagramador trabajar fácilmente y de manera síncrona entre el diseño visual de la clase y con el código fuente que esta representa.

Los atributos son características de una clase u otro elemento que representan las propiedades o elementos de información interna de ese elemento. Para una Clase como Usuario, nombreDeUsuario y direcciónDeUsuario pueden ser atributos. Los atributos tienen varias características importantes como el tipo, alcance (visibilidad), estática, derivada y notas.



Ilustración . Representación de una Clase en Enterprise Architect

Enterprise Architect provee un mecanismo para agregar, editar o eliminar los atributos a una Clase, Interfaz o Enumeración. Mediante el dialogo de edición de elementos (en este caso una Clase) se puede acceder a esta opción.



Ilustración . Ventana de creación de atributos Enterprise Architect

Otro aspecto importante a la hora de trabajar con elementos de diagramas de clase, son las operaciones o métodos. Estos hacen parte de Clases e Interfaces. Las operaciones son características de una clase u otro elemento que representan el comportamiento o los servicios que soportan un elemento. Para una clase Cliente, actualizarNombreDelCliente y obtenerDomicilioDelCliente pueden ser Operaciones. Las Operaciones tienen varias características importantes, tales como tipo, alcance (visibilidad), estática, abstracta y notas.



Ilustración . Ventana de operaciones Enterprise Architect

**Trabajar con conectores**

Las conexiones UML, junto con los elementos, forman la base de los modelos UML. Las conexiones relacionan elementos entre sí, para marcar algún tipo de relación lógica o funcional entre ellos. Cada conector tiene su propio propósito, significado y notación y se utilizan en tipos específicos de diagramas UML.

Enterprise Architect proporciona los siguientes tipos de conectores para diagramas de clase.

|  |  |
| --- | --- |
| **Conector de diagrama de clase** | **Descripción** |
|  | Relación básica entre dos elementos. |
|  | Herencia entre dos clases. |
|  | Relación de composición. |
|  | Relación de agregación. |
|  | Clase asociación entre dos clases más. |
|  | Relación de unión entre 2 interfaces. |
|  | Implementación de una interfaz por una clase. |
|  | Un elemento se anida dentro de otro. Usado para representar clases internas de otras. |
|  | Incluye paquetes importados y anidados. |
|  | Importa paquetes de uno a otro. |

Ilustración . Conectores para diagramas de clase Enterprise Architect

**Ingeniería de código**

Ingeniería de Código es un proceso que incluye generación automática de código, ingeniería inversa de código fuente y sincronización entre el código y el modelo.

*Generación de Código*

Generación de código en también conocida como Ingeniería de Código Directa. Enterprise Architect le permite generar código fuente de los elementos de los modelos UML, creando un código fuente equivalente de la clase o interface para futura elaboración y compilación. En particular puede generar código fuente en C, C++, C#, Delphi, Java, PHP, Python, ActionScript, Visual Basic and VB.NET. El código fuente generado incluye definiciones de clases, variables y funciones para cada atributo y método en la clase UML. Puede usar el Visor de código fuente para cualquier código fuente que usted esté abriendo.



Ilustración . Ventana de generación de código Enterprise Architect

*Ingeniería Inversa*

Ingeniería Inversa es la importación de código fuente existente en elementos del modelo, mapeando las estructuras de código fuente en sus representaciones UML. Esto le permite examinar código antiguo y la funcionalidad de las librerías de código para reusar, o para actualizar el modelo UML con el código. Puede realizar ingeniería inversa en los mismos lenguajes mientras realiza generación de código con Enterprise Architect. Enterprise Architect también permite realizar ingeniería inversa en algunos tipos de archivos binarios: Java .jar y .NET PE.

*Sincronización*

Sincronización es cuando los cambios en el modelo son exportados a la fuente y los cambios de la fuente son importados en el modelo. Esto le permite mantener su modelo y su código actualizados mientras el proyecto avanza.

*Ingeniería Completa*

Ingeniería completa ocurre como una combinación de generación de código inversa y directa y debe incluir sincronización entre el código fuente en todos los proyectos de ingeniería de código. Para obtener el mayor provecho de esta ingeniería en Enterprise Architect, usted debe estar familiarizado con las convenciones de modelado usadas cuando genera e ingeniería inversa de los lenguajes que usted usa.

**Plantillas de código fuente**

El Code Template Framework (CTF) le permite adaptar la manera en que Enterprise Architect genera código fuente y también le permite generar en lenguajes que Enterprise Architect no soporta específicamente ayudándole a definir las plantillas de generación de código para ese lenguaje (esto es discutido en "Enterprise Architect Software Developers' Kit" (SDK).

Las plantillas de código de EA especifican la transformación desde los elementos UML a las varias partes de un lenguaje de programación dado. Las plantillas están escritas como texto plano con una sintaxis que comparte algunos aspectos de lenguajes de marcas y de scripting. Un ejemplo simple de una plantilla usada por EA es la "plantilla de clase". Se usa para generar código fuente desde una clase UML:

%ClassNotes%   
%ClassDeclaration%   
%ClassBody%

La plantilla de arriba simplemente se refiere a otras tres plantillas, que se llaman ClassNotes, ClassDeclaration y ClassBody. Los signos delimitadores porcentaje (%) indican una "macro." Las Plantillas de Código consisten de varios tipos de macros, cada una de ellas resultantes en una sustitución en la salida que se genera. Para un lenguaje tal como C++, el resultado de procesar la plantilla de arriba podría ser:

/\*\*   
 \* Este es una nota de clase de ejemplo que se generó usando las plantillas de código  
 \* @author Sparx Systems  
 \*/   
class ClassA : public ClassB

{ .. }

**Generación de imágenes planas**

El usuario puede generar una imagen plana del diagrama y guardarla en disco, la imagen puede ser generada en los siguientes formatos:

* Windows bitmap (256 color bitmap)
* GIF image
* Windows Enhanced Metafile (standard metafile)
* Windows Placeable Metafile (older style metafile)
* PNG format
* JPG
* TGA

**Distribución automática de elementos en diagrama**

Enterprise Architect provee la facilidad de distribuir los elementos del diagrama de manera automática basada en una estructura de árbol. Algunos diagramas que debido a su estructura compleja deben ser distribuidos de manera manual por el usuario.

Esta opción está disponible para diagramas estructurales y diagramas adicionales, pero no para diagramas comportamentales de UML. Sin embargo, esta opción está disponible para diagramas de secuencia generados automáticamente por Enterprise Architect.

Si la distribución automática hecha por la herramienta no es del agrado del usuario, este tiene la posibilidad de revertir los cambios con la opción Ctrl+Z.

Existen varios “Opciones de distribución o disposicion” y cada uno de ellos permite organizar el diagrama en una forma específica en base a algunas configuraciones, una vez definido la manera de organizar los elementos del diagrama, solamente es necesario utilizar la opción de distribución y esta se ejecuta automáticamente, Enterprise Architect provee una configuración por defecto. A continuación se mencionan las diferentes configuraciones de disposicion para los diagramas:



Ilustración . Ventana de Opciones de disposición de diagrama Enterprise Architect

*Panel Opciones de remoción de ciclos* - Estas configuraciones determinan el tipo de eliminación de ciclos a usarse durante la organización.

* Voraz - Seleccione para usar el algoritmo de optimización de ciclos Voraz.
* Recorrido en profundidad - Seleccione para usar el algoritmo de optimización de ciclos de Recorrido en profundidad.

*Panel Opciones de disposición* - Estas configuraciones determinan el tipo de capas que se usará durante la organización.

* Mayor camino al sumidero - Seleccione para usar el algoritmo de Opciones de disposición de Mayor camino al sumidero.
* Mayor camino a la fuente - Seleccione para usar el algoritmo de Mayor camino a la fuente.
* Longitud de vínculos óptima - Seleccione para usar el algoritmo de Longitud de vínculos óptima.

*Panel Opciones de inicialización* - Estas configuraciones determinan el tipo de inicialización de índices y columnas a usarse durante la organización.

* Sencillo - Seleccione para usar el algoritmo de Opciones de inicialización, sencillo.
* DFS hacia afuera - Seleccione para usar el algoritmo de DFS hacia afuera.
* hacia dentro - Seleccione para usar el algoritmo de DFS hacia adentro.

*Panel Opciones de Reducción de Cruces*.

* Iteraciones - Ingrese el número de iteraciones a usarse durante la eliminación de bucles.
* Agresivo - Esta opción permite especificar si se usa o no un paso de reducción de cruces agresiva (consumo de tiempo de lectura). Si se marca, se usará la reducción de cruces agresiva. Si no se marca, no se usará la reducción de cruces agresiva.

*Panel Opciones de Disposición*.

* Espaciado de capas - Ingrese el número por defecto de unidades lógicas entre capas.
* Espaciado de columna - Ingrese el número por defecto de unidades lógicas entre columnas.
* Arriba, abajo, izquierda, derecha - Seleccionar la dirección al cual deberían dirigirse los vínculos.
* Configurar como proyecto predeterminado.

Seleccione esta casilla de verificación para aplicar las configuraciones de disposición del diagrama a todos los diagramas en el proyecto. Si marca esta casilla y presiona el botón Aceptar para un diagrama diferente, las nuevas configuraciones sustituirán las configuraciones guardadas anteriormente.

**Importación o Exportación a XMI**

Un paquete puede ser importado desde un archivo XMI (basado en XML). Esto permite que los elementos de los modelos de Enterprise Architect puedan ser movidos entre distintos modelos, para el desarrollo distribuido, manual control de versiones y otros beneficios.

Los siguientes formatos pueden ser importados:

* UML 1.3 (XMI 1.0)
* UML 1.3 (XMI 1.1)
* UML 1.4 (XMI 1.2)
* UML 2.0 (XMI 2.1)
* UML 2.1 (XMI 2.1)
* MOF 1.3 (XMI 1.1)
* MOF 1.4 (XMI 1.2)

Enterprise Architect también puede importar los archivos \*.emx and \*.uml2 generados por herramientas como por ejemplo Rational Software Architect (RSA) and Rational Software Modeler (RSM).

Para importar un paquete desde XMI se necesita hacer lo siguiente:

1. En el Explorador del proyecto, seleccione el paquete en el cual desea importar el archivo.
2. Puede:

* Hacer clic en el botón derecho y seleccionar la opción Importar/Exportar | Importar paquete desde XMI o
* Seleccione la opción Proyecto | Importar/Exportar | Importar paquete desde XMI.

Se abre la ventana Importar paquete desde XMI.



Ilustración . Ventana de importación de XMI Enterprise Architect

Se puede exportar un Paquete a un archivo XMI (basado en XML). Esto permite que los elementos del Modelo de EA se muevan entre modelos, para desarrollo distribuido, control de versiones y otros beneficios. También permite una exportación limitada de elementos de modelo de EA a Rational Rose y a otras herramientas que implementen los estándares UML 2.1 XMI 2.1 o UML 1.3 XMI 1.2/XMI 1.1 / XMI 1.0.

Para exportar un paquete a XML, proceda con los siguientes pasos:

1. En el Explorador del proyecto, seleccione el paquete que quiere exportar.
2. Puede:

* Hacer con el botón derecho y seleccionar el menú Importar/Exportar | Exportar el paquete a XMI, o
* Seleccionar la opción Proyecto | Importar/Exportar | Exportar el paquete a XMI.

Se abre la ventana Exportar paquete a XMI.



Ilustración . Ventana de exportación de XMI Enterprise Architect

## STARUML

StarUML™ es una plataforma de modelado de software que soporta UML (Unified Modeling Language). Se basa en la versión 1.4 de UML versión y proporciona 11 tipos diferentes de diagrama, y acepta la notación UML 2.0. Apoya activamente el enfoque de MDA (Model Driven Architecture), apoyando el concepto de perfil UML. StarUML destaca en la personalización del entorno del usuario y tiene una alta extensibilidad en su funcionalidad.

### Historia

En un principio StarUML fue formalmente conocido como “Plastic” o “Agora Plastic”. En 1996 se conoce la primera versión de “Plastic” (v0.9), se trataba de una herramienta muy simple para dibujar módulos de software y sus dependencias. En 1997 la versión 1.0 de Plastic fue liberada, se caracterizó por ser una herramienta gratuita con soporte para OMT.

La versión 1.1 de Plastic fue liberada en 1998, con la principal característica que incluía soporte para diagramas de clase UML. En 1999 se funda la empresa Plastic Software, Inc, junto con la liberación de la versión 2.0 de su herramienta Plastic. Esta versión de la herramienta cubría parcialmente el estándar UML, generación de código en Java e ingeniería inversa.

En 2001 se libera la versión 3.0 de Plastic en donde se soporta completamente el estándar UML en su versión 1.3. Sin embargo en 2003 la herramienta Plastic es completamente rediseñada y reescrita, logrando un soporte completo de UML 1.4 mediante una arquitectura abierta.

En 2005 Agora Plastic es liberado, incluyendo características de internacionalización y muchas otras para extender la plataforma. En ese mismo año recibe la certificación de ‘Good Software’ de parte del Ministerio de Información y comunicación de Korea.

A finales de 2005 la herramienta fue renombrada a StarUML 5.0 y fue liberada, se convirtió en un proyecto open source con soporte completo de UML 2.0 y se implementó una notación de tecnología extensible, de tal manera que se pueden desarrollar complementos en varios lenguajes y añadirlos a la aplicación.

### Características Principales

La siguiente tabla muestra las características principales de esta herramienta de modelado. Esta información fue tomada explícitamente del documento guía de usuario [9].

|  |  |
| --- | --- |
| **Característica** | **Descripción** |
| Estándar UML | StarUML se adhiere estrictamente a la especificación estándar UML especificado por el OMG para el modelado de software. Teniendo en cuenta el hecho de que los resultados de la información de diseño pueden llegar a 10 años o más en el futuro, la dependencia específica del proveedor sintaxis UML irregular y la semántica puede ser muy arriesgado. StarUML maximiza la misma para ordenar estándar UML 1.4 y significado, y acepta la notación UML 2.0, sobre la base de meta-modela robusto. |
| Formato de modelo Open Source | A diferencia de muchos productos existentes que gestionan sus propios modelos de formato heredado ineficiente, StarUML gestiona todos los archivos en el formato XML estándar. Códigos escritos en estructuras fáciles de leer y sus formatos se pueden modificar convenientemente usando un analizador XML. Dado que XML es un estándar mundial, este es sin duda una gran ventaja. |
| Soporte real de MDA | StarUML verdaderamente soporta UML Profile. Esto maximiza la extensibilidad de UML, haciendo el modelado de aplicaciones posibles, incluso en áreas como las finanzas, la defensa, el comercio electrónico, los seguros y la aeronáutica. Verdaderamente modelos independientes de plataforma (PIM) pueden ser creados, y modelo específicos de la plataforma (PSM) y códigos ejecutables pueden ser generados de forma automática de cualquier modo. |
| Aplicación de metodologías y plataformas | StarUML manipula el concepto de enfoque, creando de ambientes que se adaptan a cualquier metodología o proceso. No sólo los modelos de Frameworks de aplicación para plataformas como. NET y J2EE, sino también las estructuras básicas de los modelos de software (por ejemplo, 4+1 View Model, etc.) se puede definir fácilmente. |
| Extensibilidad | Todas las funciones de las herramientas de StarUML están automatizadas acorde a Microsoft COM. Cualquier lenguaje que soporte COM (Visual Basic Script, JavaScript, VB, Delphi, C++, C#, VB.NET, Python, etc.) puede ser usado para manejar StarUML y desarrollar elementos integrados Add-In. |
| Función de verificación de modelos de software | Los usuarios pueden cometer muchos errores durante el modelamiento de software. Estos errores pueden ser muy costosos si permanecen hasta la etapa final de codificación. Con el objetivo de prevenir este problema, StarUML automáticamente verifica los modelos de software desarrollados por el usuario, facilitando descubrir de manera temprana errores, permitiendo software más completo y con menos faltas. |
| Add-Ins útiles. | StarUML incluye algunos Add-Ins útiles con varias funcionalidades. Estos generan código fuente en varios lenguajes de programación y convierten código fuente en modelos, importan archivos de Rational Rose, intercambian información de modelos con otras herramientas usando XMI, y soportan diseño de patrones. Estos Add-Ins proporcionan reusabilidad adicional, productividad, flexibilidad e interoperabilidad para la información de modelado. |

Tabla . Características principales de StarUML

### Funciones para Diagramas de Clase

StarUML permite muchos elementos para el diseño de diagramas de clase, entre ellos se encuentran: Clases, Interfaces, Enumeraciones, Paquetes, Subsistemas, Signal, Exception, Port, Asociación, Asociación directa, Agregación, Composición, Generalización, Realización, Clase Asociación, Conector, Objeto y Link.

## DRAW.IO

# CONCEPTOS DE DIAGRAMAS DE CLASES

# CLASIFICACIÓN DE LA INFORMACIÓN

## MATRIZ DE CARACTERISTICAS

A continuación se mencionan las características que poseen las herramientas para el diseño de diagramas de clase. Adicionalmente se seleccionan aquellas que serán susceptibles de implementación en el proyecto.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Característica** | **Enterprise Architect** | **StarUML** | **Draw.IO** |  |
|  |  |  |  |  |
|  |  |  |  |  |

## CARACTERISTICAS PROPIAS

Si bien es cierto que nuestra aplicación posee características compartidas y comunes por la mayoría de herramientas de modelado para diagramas de clase, también es cierto que provee algunas características que son propias y fueron consideradas para aportar un valor agregado a la herramienta. A continuación se listan estas características.

* Compartir diagramas entre los usuarios de la aplicación
* Diseñar de manera concurrente un diagrama específico al cual acceden dos o más usuarios diferentes.

Estas dos características representan las funcionalidades propias de la aplicación, las cuales otorgan a la herramienta un grado de singularidad.

## LISTA DE FUNCIONALIDADES

En base al análisis hecho anteriormente con respecto a las características de cada una de las herramientas case estudiadas, se abstrajo la siguiente lista de funcionalidades que serán susceptibles de implementación en el proyecto, estas servirán como insumo principal en la fase de exploración para determinar las historias de usuario del proyecto.

* Registrar usuarios
* Crear diagramas para los usuarios
* Compartir diagramas entre diferentes usuarios
* Diseñar o dibujar de manera concurrente un diagrama
* Generar código fuente a partir de un diagrama
* Generar imágenes planas a partir de un diagrama
* Exportar el diagrama en formato XMI
* Importar un diagrama en formato XMI.

Las características mencionadas anteriormente representan de manera global el alcance del proyecto y los servicios que la aplicación ofrecerá a los usuarios una vez se encuentren en producción.

Adicionalmente existen características un poco más específicas que se deben considerar al momento de dibujar diagramas en el diseñador. Estas características proporcionan una mejor experiencia de usuario y permiten diseñar de manera fácil diagramas de manera conjunta.

* Herramienta Zoom
* Edición común (Copiar, Cortar y Pegar) elementos del diagrama
* Deshacer y Rehacer (Undo-Redo)

# [METODOLOGÍA DE DESARROLLO](#TablaContenido)

La metodología de desarrollo seleccionada para el proyecto fue *eXtreme Programming* (XP), debido a su naturaleza y al tamaño del equipo de trabajo se puede adaptar fácilmente al desarrollo del proyecto, considerando que brinda flexibilidad y la posibilidad de adaptación a los cambios que pueden surgir en el transcurso del proyecto.

## EXPLORACIÓN

### Identificación de Roles del Proyecto

El desarrollo del proyecto será llevado a cabo por un grupo pequeño de personas, sin embargo cada una tiene una labor específica e importante para la correcta ejecución del mismo. A continuación se describen los roles del proyecto y su participación en el proyecto:

* **Desarrollador**: Gabriel Leonardo Díaz Cárdenas, estudiante de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander.
* **Cliente**: En un proceso de desarrollo ágil es de vital importancia para el desarrollo del proyecto, sin embargo en nuestro caso no tenemos un ente o persona específica a la cual se le entregará la aplicación, en su lugar tenemos un grupo general de personas a las cuales irá dirigido el proyecto:

“Estudiantes del Plan de Estudios de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, en especial, estudiantes que cursan materias como Programación Orientada a Objetos y Análisis y Diseño de sistemas. Adicionalmente los Docentes que hacen parte de estas cursos del pensum académico”.

* **Director de Proyecto**: Marco Antonio Adarme Jaimes, Msc. en Computación. Director general del proyecto quien estará controlando su ejecución y orientando el proceso de implementación, además quien estará verificando que se cumplan los objetivos planteados al inicio del proyecto.
* **Asegurador de la Calidad**: Las labores de verificación y pruebas del funcionamiento de la aplicación estarán a cargo del Msc. Margo Antonio Adarme Jaimes, a quien se le entregaran las liberaciones de cada iteración (mencionadas más adelante) y quien dará el visto de aprobación.
* **Consultor**: Este rol es asumido por personas externas, expertos en algunos de los temas que vamos a tratar durante la ejecución del proyecto. A continuación se mencionan las personas que ejercen este rol en el proyecto:
* **Gaudenz Alder**: Cofundador de la empresa JGraph Ltd, creadora de la librería JavaScript mxGraph utilizada para el funcionamiento de este proyecto. Gaudenz inició JGraph como su proyecto de tesis para el instituto tecnológico de Suiza, Zurich en el 2000. Ha liderado el desarrollo del proyecto para la empresa en 13 años continuos de experiencia en el diseño de la librería.
* **PrimeFaces Community**: Este agente externo está compuesto por un grupo de desarrolladores alrededor del mundo que se encargan de resolver problemas relacionados con el uso del FrameWork JSF PrimeFaces, y permiten a usuarios novatos aprender fácilmente el uso de la herramienta.

### Identificación de Actores del Sistema

En UML se define un *actor* como aquella persona, proceso o cosa que interactúa con un sistema, subsistema o clase[[15]](#footnote-15). Para nuestro proceso de desarrollo es de vital importancia determinar cuáles son los actores de nuestro sistema, de modo que se pueda tener una idea clara de las necesidades conjuntas y particulares de cada uno de ellos. A continuación se mencionan cada uno de los actores y su descripción:

|  |  |  |
| --- | --- | --- |
| Visitante.bmp | **VISITANTE** | |
| Representa cualquier persona que llega a la aplicación en búsqueda de información, ya sea de manera accidental o voluntaria. Este actor no tiene una representación real dentro de la aplicación y solamente es manejado de manera conceptual, en efecto cualquier persona inicia siendo un visitante de la aplicación. | |
| **Clase** | Ninguna |
| **Súper Clase** |  |
| **Interacción con la aplicación** | | |
| * Búsqueda de Información o curiosidad. * Registrarse en el sistema. | | |

Tabla . Especificación del actor Visitante

|  |  |  |
| --- | --- | --- |
| Visitante.bmp | **Usuario** | |
| Representa una persona que hace login dentro de la aplicación y que le es permitido usar el diseñador de diagramas. Este es un actor abstracto y necesita ser redefinido en una representación concreta. Dentro de la aplicación Usuario es una interfaz implementada por dos clases. | |
| **Clase** | *IUser* |
| **Súper Clase** |  |
| **Interacción con la aplicación** | | |
| * Inicio de Sesión * Cerrar Sesión * Diseñar diagrama * Generar Código * Generar Imagen * Exportar XMI | | |

Tabla . Especificación del actor Usuario

|  |  |  |
| --- | --- | --- |
| Visitante.bmp | **Invitado** | |
| Un invitado en la aplicación es la representación de un usuario que utiliza la opción de *Ver Demostración* e inicia sesión dentro de la aplicación sin registro previo. Este actor posee una representación real dentro de la aplicación, sin embargo sus privilegios son limitados y no tiene acceso a algunas funcionalidades. | |
| **Clase** | *Guest* |
| **Súper Clase** | *IUser* |
| **Interacción con la aplicación** | | |
| * Todas las interacciones heredadas * Ver demostración: Esta opción puede ser usada por cualquier visitante, sin embargo en el momento de ingresar a la aplicación dicho visitante se convierte en un usuario invitado. | | |

Tabla . Especificación del actor Invitado

|  |  |  |
| --- | --- | --- |
| Visitante.bmp | **Diagramador** | |
| Representa una persona registrada en la aplicación mediante el formulario designado para tal labor. A esta persona se le asigna una cuenta usuario mediante su dirección de correo electrónico, y por medio de la cual ingresa a la aplicación de manera controlada para hacer uso de todos los servicios que están a su disposición. | |
| **Clase** | *User* |
| **Súper Clase** | *IUser* |
| **Interacción con la aplicación** | | |
| * Todas las interacciones heredadas * Ver panel de control (Dashboard) * Gestionar diagramas (Crear, editar, borrar, copiar) * Compartir diagramas * Importar XMI * Gestionar Perfil * Cambiar Contraseña * Desactivar cuenta * Reactivar cuenta * Recuperar contraseña * Edición concurrente de diagramas | | |

Tabla . Especificación del actor Diagramador

Dentro de la aplicación no se dispone de un usuario administrador, ya que no existen tareas o configuraciones especiales que se tengan que hacer de manera controlada. Cada usuario es capaz de interactuar con la aplicación de la misma manera y todos ellos disponen de los mismos privilegios. Esto evita la necesidad de tener a una persona realizando configuraciones en la aplicación y la libera de este tipo de dependencia. Ambos *Invitado* y *Diagramador* son *Usuarios* de la aplicación debido a que pueden iniciar sesión dentro de ella e ingresar al diseñador. Los actores pueden ser representados mediante el siguiente diagrama:



Ilustración 18. Diagrama de Actores

### Historias de Usuario

A continuación se describen las historias de usuario recolectadas para la implementación del proyecto. El siguiente formato [7] dispone de la información correspondiente a cada historia de usuario, incluyendo la prioridad, la estimación de tiempo y la iteración a la cual fue asignada, estos conceptos son explicados en la sección 7.2.

|  |  |  |
| --- | --- | --- |
| **HISTORIA DE USUARIO** | | |
| **ID:** H1 | **Nombre de historia:** Crear cuenta de diagramador | |
| **Usuario:** Visitante | | **Iteración Asignada:**  Iteración 1 |
| **Prioridad:** Media | | **Estimación:** 1 Semana |
| **Descripción:**  Un visitante puede crear una cuenta de diagramador para iniciar sesión en el sistema. Para ello, dispone de un formulario de registro donde se captura información básica del visitante, como los nombres, apellidos, correo electrónico, género y contraseña. | | |
| **Reglas u Observaciones:**   * La dirección de correo electrónico de cada diagramador debe ser única en el sistema, de modo que la aplicación debe controlar y verificar que se cumpla esta condición. El correo electrónico es usado como identificador de la cuenta. * La contraseña ingresada debe tener mínimo 5 caracteres de longitud y máximo 20. Puede incluir cualquier combinación de letras y números. * El correo electrónico es único para todo el sistema, por lo tanto un diagramador no puede registrarse con un correo electrónico ya usado por otra persona. * El género se usa para asignar una imagen por defecto al diagramador una vez crea su cuenta, esta imagen puede ser cambiada a gusto de la persona en cualquier momento. * Una vez creada la cuenta de diagramador, esta permanece INACTIVA hasta que se realice la activación respectiva mediante el correo de confirmación. Este correo se envía a la dirección ingresada al momento de crear la cuenta. * Al momento de crear la cuenta de diagramador el sistema debe generar un código de verificación encriptado, el cual se envía junto con el correo de activación y sirve para validar los datos al momento de ejecutar el proceso. * El código de activación debe ser único por cada solicitud. * El diagramador no puede iniciar sesión en el sistema hasta que active su cuenta. | | |

Tabla . Formato de historia de usuario H1.

|  |  |  |
| --- | --- | --- |
| **HISTORIA DE USUARIO** | | |
| **ID:** H2 | **Nombre de historia:** Activar cuenta de diagramador | |
| **Usuario:** Diagramador | | **Iteración Asignada:** Iteración 1 |
| **Prioridad:** Media | | **Estimación:** 1 Semana |
| **Descripción:**  Este es un requerimiento para poder usar la aplicación una vez se creada la cuenta de diagramador. La activación de la cuenta se hace mediante un link que se envía a la dirección de correo que el visitante ingresó al momento de su registro, este link contiene la información necesaria para validar los datos y un código único encriptado para brindar seguridad. | | |
| **Reglas u Observaciones:**   * El código de activación tiene un periodo de vigencia de 2 días, por lo tanto si la persona no realiza la activación dentro de ese tiempo el código debe caducar y se debe generar uno nuevo. * El código de activación es válido solamente una vez. Por lo tanto una vez utilizado ya se será válido nunca más. * Las cuentas de diagramador pueden tener 3 estados diferentes: INACTIVA el cual se obtiene cuando recién se crea la cuenta de diagramador, ACTIVADA que es el estado normal de una cuenta vigente y DESACTIVADA el cual indica que la cuenta esta temporalmente invalida y no puede ser usada, esto último hecho por voluntad del diagramador. * La cuenta podrá ser activada únicamente si esta existe y el código de verificación concuerda con el guardado en base de datos y además si el código de verificación no ha caducado. * Cuando el código esta vencido la aplicación debe generar uno nuevo y enviarlo automáticamente a la dirección de correo del diagramador, adicionalmente debe informarle que se ha generado un nuevo código. * Una vez realizada la activación de la cuenta, el usuario puede ingresar normalmente a la aplicación por medio de la opción iniciar sesión. | | |

Tabla . Formato de historia de usuario H2

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H3 | **Usuario:** Diagramador |
| **Nombre de historia:** Desactivar cuenta de diagramador | |
| **Descripción:**  Por medio de esta opción el usuario podrá desactivar su cuenta temporalmente en caso de así quererlo. Cuando se desactiva una cuenta de usuario, se mantiene toda la información que el usuario ya tenía, pero no le es permitido iniciar sesión en la aplicación, tampoco le podrán ser compartidos diagramas de otros usuarios. | |
| **Reglas u Observaciones:**   * Para desactivar la cuenta el diagramador debe iniciar sesión y utilizar la opción disponible para tal labor. * El sistema debe solicitar la contraseña antes de desactivar la cuenta. * Al momento de desactivar la cuenta se le envía un correo al diagramador con la información pertinente, aclarando que no recibirá más notificaciones de la aplicación. | |

Tabla . Formato de historia de usuario H3.

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H4 | **Usuario:** Diagramador |
| **Nombre de historia:** Reactivar cuenta de diagramador | |
| **Descripción:**  Esta es la operación inversa a la descrita en H3, el usuario reactiva su cuenta mediante un correo de confirmación en donde se le envía un código encriptado y la información necesaria auto contenida en un link, lo único que debe hacer el usuario es clicar el link y lo llevará automáticamente a la aplicación donde la cuenta debe estar ACTIVADA nuevamente. | |
| **Reglas u Observaciones:**   * Este proceso se ejecuta cuando el diagramador intenta iniciar sesión (H6) en la aplicación pero esta detecta que ha sido desactivada, en este caso la aplicación le informa al usuario que esta ha sido desactivada y le pregunta que si desea volver a activarla. * En caso de que el diagramador desee volver a activar la cuenta, entonces se le envía un correo electrónico con un código de verificación encriptado, mediante el cual se valida la información necesaria. * La activación se realiza tal cual como se menciona en H2. | |

Tabla . Formato de historia de usuario H4

|  |  |  |
| --- | --- | --- |
| **HISTORIA DE USUARIO** | | |
| **ID:** H5 | **Nombre de historia:** Recuperar contraseña | |
| **Usuario:** Diagramador | | **Iteración Asignada:** Iteración 1 |
| **Prioridad:** Baja | | **Estimación:** 1 Semana |
| **Descripción:**  Un diagramador puede reasignar una nueva contraseña a su cuenta en caso de haber olvidado la anterior. El proceso consiste en ingresar la dirección de correo y de manera controlada se le envía un email de confirmación con un link para hacer la reasignación. El link contiene la información necesaria para validar la cuenta y lleva al diagramador a un formulario donde puede asignar una nueva contraseña para su cuenta. | | |
| **Reglas u Observaciones:**   * Las reglas de validación del código de verificación deben ser las mismas que las ya mencionadas anteriormente. * Una vez validado el código de verificación se le debe mostrar al diagramador un formulario donde debe ingresar la nueva contraseña y la confirmación, las reglas para la contraseña son las mismas descritas en H1. * La nueva contraseña y la confirmación deben coincidir. * La contraseña anterior no puede ser recuperada y no es necesario validar que la nueva contraseña sea diferente a la anterior. * Una vez los datos sean validados se procede a cambiar la contraseña actual por la nueva. | | |

Tabla . Formato de historia de usuario H5

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H6 | **Usuario:** Diagramador o Invitado |
| **Nombre de historia:** Iniciar sesión | |
| **Descripción:**  Funcionalidad mediante la cual un usuario ingresa a la aplicación. Esta operación puede ser llevada a cabo de dos maneras diferentes, un usuario registrado (diagramador) que ingresa a la aplicación con su dirección de correo o un usuario invitado que ingresa a la aplicación sin registro previo.  Para los usuarios registrados se dispone de un formulario de ingreso, donde se solicita la dirección de correo electrónico y la contraseña. | |
| **Reglas u Observaciones:**   * Para poder iniciar sesión con una cuenta creada es necesario que esta cuenta tenga el estado ACTIVADA, en caso de no cumplir esta restricción el usuario no podrá ingresar a su cuenta. * Los usuarios tipo invitado iniciar sesión para entrar en modo demostración a la aplicación, el cual se realiza sin necesidad de registro previo. | |

Tabla . Formato de historia de usuario H6

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H7 | **Usuario:** Diagramador |
| **Nombre de historia:** Modificar perfil de usuario | |
| **Descripción:**  Esta opción permite cambiar la información de registro del usuario, tal como nombres, apellidos, fecha de nacimiento, sexo y avatar (imagen de identificación). No se puede cambiar la dirección de correo electrónico del usuario. | |
| **Reglas u Observaciones:**   * La dirección de correo electrónico no puede ser cambiada. | |

Tabla . Formato de historia de usuario H7

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H8 | **Usuario:** Diagramador |
| **Nombre de historia:** Cambiar contraseña | |
| **Descripción:**  Permite cambiar la contraseña del usuario, es necesario siempre ingresar la contraseña anterior. El formulario solicita una nueva contraseña y una confirmación para garantizar que el usuario puede recordarla fácilmente. Solo se podrá cambiar la contraseña si la información requerida es correcta, es decir, si la anterior contraseña coincide con la guardada en base de datos y la nueva contraseña y su confirmación son exactamente iguales. | |
| **Reglas u Observaciones:**   * Se debe solicitar la contraseña anterior para proceder a cambiarla. * El cambio se hace mediante un formulario que tiene 3 campos, la contraseña anterior, la nueva contraseña y una confirmación de la nueva contraseña. * Las reglas para la nueva contraseña deben ser las mismas ya mencionadas. | |

Tabla . Formato de historia de usuario H8

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H9 | **Usuario:** Diagramador |
| **Nombre de historia:** Crear diagrama | |
| **Descripción:**  Esta opción le permite a un usuario crear un diagrama para sí mismo. Los datos básicos de entrada son solamente el nombre y una descripción opcional del diagrama. No se realiza ninguna validación especial aparte de que exista un nombre. El usuario quien crea el diagrama es asignado como propietario del mismo y puede compartirlo con otros. Inicialmente la representación del diagrama en XMI es vacía, solamente contiene un package (paquete) principal que tiene el mismo nombre del diagrama y no tiene elementos hijos. | |
| **Reglas u Observaciones:**   * Se debe solicitar el nombre del diagrama y una descripción opcional. | |

Tabla . Formato de historia de usuario H9

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H10 | **Usuario:** Diagramador |
| **Nombre de historia:** Compartir diagrama | |
| **Descripción:**  Permite al usuario registrado compartir diagramas con otros usuarios del sistema. Esta opción es permitida únicamente para los diagramas de los cuales el usuario tenga privilegios.  En general existen 4 diferentes tipos de privilegios ordenados de menor a mayor rango: READ\_ONLY (solo lectura), EDIT (edición), SHARE (volver a compartir) y OWNER (Propietario), este último se otorga por defecto al usuario quien crea el diagrama. Básicamente un usuario solamente podrá compartir diagramas de su propia autoría o cuando se le ha sido asignado el privilegio de SHARE, en cuyo caso el usuario es un segundo propietario del diagrama. Como se mencionó anteriormente, un usuario con su cuenta INACTIVADA o DESACTIVADA no podrá recibir ningún diagrama compartido. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H10

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H11 | **Usuario:** Diagramador |
| **Nombre de historia:** Gestionar privilegios del diagrama | |
| **Descripción:**  Esta opción le permite al propietario del diagrama modificar o quitar los privilegios otorgados a otros usuarios (ya sea por el mismo o por otros), mediante el uso de la funcionalidad descrita en H10. Básicamente el usuario puede remover los privilegios otorgados sobre un diagrama, lo que significa que el usuario afectado no podrá seguir haciendo uso de este, o también puede modificar los privilegios con el objetivo de otorgar al otro usuario mayor o menor control sobre el diagrama. Esta operación solo puede ser ejecutada por el propietario del diagrama, los demás usuario solo pueden ver quienes acceden a dicho diagrama pero no pueden hacer ningún cambio. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H11

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H12 | **Usuario:** Diagramador |
| **Nombre de historia:** Editar diagrama | |
| **Descripción:**  Permite cambiar los datos de creación del diagrama (nombre y descripción). Para poder ejecutar esta opción es necesario que el usuario tenga los privilegios necesarios para ello.  El privilegio necesario para ejecutar esta operación debe ser mayor que READ\_ONLY, por lo tanto un usuario a quien solo se le ha permitido ver el diagrama no podrá hacer uso de esta opción. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H12

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H13 | **Usuario:** Diagramador |
| **Nombre de historia:** Eliminar diagrama | |
| **Descripción:**  Permite eliminar un diagrama de manera controlada. Esta opción tiene varias restricciones:   * Un diagrama solo puede ser borrado por el propietario del mismo * Un usuario quien tenga compartido un diagrama, no puede borrar dicho diagrama, solamente puede borrar su acceso a este. * Cuando se borra un diagrama que esta compartido por parte de su propietario, se informa a los usuarios que este fue eliminado por medio de correo electrónico. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H13

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H14 | **Usuario:** Diagramador |
| **Nombre de historia:** Copiar diagrama | |
| **Descripción:**  Esta opción permite copiar un diagrama al cual se tiene acceso (compartido o propio). Los datos son copiados exactamente pero el propietario de la copia será quien realiza la opción. Los diagramas compartidos pueden ser copiados pero ninguno de los usuarios podrá ver el nuevo diagrama a menos que se comparta explícitamente.  Un usuario solo puede copiar un diagrama si tiene el privilegio para editarlo, por lo tanto un usuario a quien solamente se le ha otorgado privilegios de lectura no podrá hacer uso de esta opción. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H14

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H15 | **Usuario:** Diagramador |
| **Nombre de historia:** Diseñar diagrama | |
| **Descripción:**  Permite abrir el Diseñador y editar los elementos UML del diagrama de clase. Dentro de este diseñador se encuentran todas las funciones básicas para manejar los diagramas, agregar elementos, editarlos, copiarlos, agregar comentarios.  Esta historia de usuario engloba todas las opciones disponibles en el diseñador relacionadas con el pintado de diagramas, a continuación se mencionan algunas de ellas:   * Añadir, editar, borrar elementos del diagrama. Clases, Interfaces, Enumeraciones, Relaciones, Comentarios, etc. * Agregar variables y métodos a los elementos respectivos. * Usar las opciones de edición: Copiar, Pegar, Cortar, Deshacer, Rehacer. * Zoom. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H15

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H16 | **Usuario:** Diagramador |
| **Nombre de historia:** Diseñar diagrama concurrentemente | |
| **Descripción:**  Esta opción permite a varios usuarios hacer uso de un diagrama de manera concurrente, reflejando los cambios hechos por uno o por otro en tiempo real. Para el usuario el uso de esta funcionalidad es transparente ya que no necesita realizar ninguna operación explicita adicional a la descrita en la historia H15. Sin embargo en el diseñador se les informa a los usuarios sobre quienes están haciendo uso del diagrama al mismo tiempo. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H16

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H17 | **Usuario:** Invitado |
| **Nombre de historia:** Ver demostración del diseñador | |
| **Descripción:**  Un invitado de la aplicación podrá hacer uso de ella sin registro previo, con el objetivo de obtener información o verificar los beneficios que esta puede prestarle. En la página de bienvenida se encuentra un link de ingreso, este link redirige al usuario al *Diseñador* de diagramas y crea un diagrama vacío, a partir de allí el usuario podrá hacer modificaciones a su antojo. Sin embargo, algunas funcionalidades están restringidas, como el compartir diagramas, guardarlos, general código fuente o imágenes. Únicamente se podrán hacer exportaciones en formato XMI del diagrama construido. Esta opción está estrechamente relacionada con H15. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H17

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H18 | **Usuario:** Diagramador |
| **Nombre de historia:** Generar código fuente | |
| **Descripción:**  Esta opción le permite al usuario generar el código fuente del diagrama que se encuentra editando mediante el Diagramador. El código fuente generado es únicamente en lenguaje Java. Los archivos fuentes pueden ser descargados uno a uno, o descargar un empaquetado que contenga todos los archivos generados. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H18

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H19 | **Usuario:** Diagramador |
| **Nombre de historia:** Generar imagen del diagrama | |
| **Descripción:**  Permite obtener una representación plana del diagrama en una imagen con diferentes tipos de formatos (.gif, .png, .jpg). Esta opción es muy útil para realizar documentaciones o impresiones del diagrama completo. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H19

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H20 | **Usuario:** Diagramador |
| **Nombre de historia:** Importar XMI | |
| **Descripción:**  Permite importar un diagrama desde un archivo en disco en formato estándar XMI. La importación del archivo XMI reemplaza la representación actual del diagrama. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario H20

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **ID:** H21 | **Usuario:** Diagramador |
| **Nombre de historia:** Exportar XMI | |
| **Descripción:**  Permite exportar un diagrama en formato del estándar XMI. La exportación se realiza del diagrama completo, incluyendo todos los elementos creados en él. La salida de este será un archivo XML con el formato específico del estándar. | |
| **Reglas u Observaciones:** | |

Tabla . Formato de historia de usuario 21

## PLANIFICACIÓN

### Priorización de Historias de Usuario

Una vez determinadas las historias de usuario es necesario proceder a realizar una priorización y establecer cuáles deben ser implementadas en iteraciones tempranas y cuales se pueden postergar hasta el final del proyecto. Así como determinar la interdependencia que pueda existir entre estas y poder definir el orden de implementación.

La prioridad se asigna en conjunto con el cliente y se determina en base al impacto que la historia de usuario tiene sobre los objetivos del proyecto. De esta manera se establecen la siguiente clasificación de prioridades:

* **ALTA**: Significa que la historia de usuario tiene un impacto significativo en los objetivos del proyecto y debe implementarse para poder alcanzar los mismos.
* **MEDIA**: Significa que la historia de usuario tiene un impacto moderado sobre los objetivos del proyecto.
* **BAJA**: Significa que la historia de usuario es trivial y no impacta directamente los objetivos del proyecto, se puede describir como un *Nice to Have* (bueno tener) el cual puede aumentar la satisfacción del cliente.

Adicional a la prioridad asignada a cada historia de usuario, se ha considerado la interdependencia que existe entre ellas, el cual es un concepto importante al determinar el orden de implementación y la precedencia.

|  |  |  |
| --- | --- | --- |
| **HISTORIA DE USUARIO** | **PRIORIDAD** | **DEPENDENCIA** |
| H1. Crear cuenta de usuario | MEDIA |  |
| H2. Activar cuenta de usuario | MEDIA | H1 |
| H3. Desactivar cuenta de usuario | BAJA | H6 |
| H4. Recuperar cuenta de usuario | BAJA | H3 |
| H5. Reasignar contraseña | BAJA | H1, H2 |
| H6. Iniciar sesión | MEDIA | H1, H2 |
| H7. Modificar perfil de usuario | BAJA | H6 |
| H8. Cambiar contraseña | BAJA | H6 |
| H9. Crear diagrama | MEDIA | H6 |
| H10. Compartir diagrama | ALTA | H9 |
| H11. Gestionar privilegios de diagrama | ALTA | H9 |
| H12. Editar diagrama | MEDIA | H9 |
| H13. Eliminar diagrama | MEDIA | H9 |
| H14. Copiar diagrama | BAJA | H9 |
| H15. Diseñar diagrama | ALTA | H9 |
| H16. Diseñar diagrama concurrentemente | ALTA | H15 |
| H17. Ver demostración | BAJA | H15 |
| H18. Generar código fuente del diagrama | ALTA | H9 |
| H19. Generar imagen del diagrama | ALTA | H9 |
| H20. Importar XMI | BAJA | H9 |
| H22. Exportar XMI | BAJA | H9 |

Tabla . Priorización de Historias de Usuario

### Estimación de Historias de Usuario

La estimación del esfuerzo en base al tiempo que emplea el llevar a cabo cada historia de usuario. Esta tarea es realizada por el equipo de desarrollo considerando la complejidad que implica la implementación.

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | **ESTIMACIÓN** |
| H1. Crear cuenta de usuario | 1 Semana |
| H2. Activar cuenta de usuario | 1 Semana |
| H3. Desactivar cuenta de usuario | 1 Semana |
| H4. Recuperar cuenta de usuario | 1 Semana |
| H5. Reasignar contraseña | 1 Semana |
| H6. Iniciar sesión | 2 Semanas |
| H7. Modificar perfil de usuario | 1 Semana |
| H8. Cambiar contraseña | 1 Semana |
| H9. Crear diagrama | 1 Semana |
| H10. Compartir diagrama | 2 Semanas |
| H11. Gestionar privilegios de diagrama | 2 Semanas |
| H12. Editar diagrama | 1 Semana |
| H13. Eliminar diagrama | 1 Semana |
| H14. Copiar diagrama | 1 Semana |
| H15. Diseñar diagrama | 3 Semanas |
| H16. Diseñar diagrama concurrentemente | 2 Semanas |
| H17. Ver demostración | 1 Semana |
| H18. Generar código fuente del diagrama | 2 Semanas |
| H19. Generar imagen del diagrama | 2 Semanas |
| H20. Importar XMI | 2 Semanas |
| H22. Exportar XMI | 2 Semanas |

Tabla . Estimación de tiempo para Historias de Usuario

### Definición de Iteraciones

Basado en la priorización y estimación de las historias de usuarios, se definen las iteraciones del proyecto, determinando la cantidad de iteraciones (entregables del proyecto) necesarias para la completa ejecución del mismo y el cumplimiento de los objetivos, definiendo las historias de usuario cubiertas por cada una de ellas.

Basado el tamaño del proyecto se determinó que las iteraciones deben ser máximo de 3 semanas de duración, por lo tanto se trataran establecen la iteraciones basado en la estimación de tiempo realizada por cada historia de usuario. Otro factor adicional para la definición de las iteraciones es la dependencia que existe entre las historias de usuario, considerando que para poder implementar algunas de ellas es necesario que otras funcionalidades ya estén disponibles en la aplicación.

La siguiente tabla muestra el plan de iteraciones y la distribución de las historias de usuario en cada una de ellas.

|  |  |
| --- | --- |
| **ITERACIÓN** | **HISTORIAS DE USUARIO** |
| Iteración 1 | H1, H2, H5 |
| Iteración 2 | H6, H9 |
| Iteración 3 | H15 |
| Iteración 4 | H10, H12 |
| Iteración 5 | H16, H17 |
| Iteración 6 | H18, H19 |
| Iteración 7 | H11, H13, H14 |
| Iteración 8 | H7, H8 |
| Iteración 9 | H3, H4 |
| Iteración 10 | H20, H21 |

Tabla . Plan de iteraciones

### Metáfora del Sistema

Aplicación web basada en ambiente Cloud que asemeja una herramienta CASE desktop para la creación, edición y manejo de Diagramas de Clase UML. Permite a los usuarios diseñar de manera fácil diagramas de clase UML a través de una interfaz web, así como compartir sus diagramas con otros usuarios y editar concurrentemente los diagramas compartidos. Se permite agregar componentes desde una paleta al lienzo principal del diseñador, y se dispone de los elementos básicos para el diseño de diagramas de clase. Al final de la edición la aplicación permite obtener el código fuente que representa el diagrama dibujado.

## DESCRIPCIÓN DE LAS ITERACIONES

A continuación se menciona cada iteración del proyecto describiendo las historias de usuario que estas abarcan y detallando las actividades a realizar en cada una de ellas.

### Iteración 1

Las historias de usuario abarcadas por esta iteración son H1, H2 y H5, por lo tanto para cada una de ellas se necesitan realizar tareas específicas y tareas conjuntas, a continuación se listan las actividades a realizar a lo largo de esta iteración.

**Actividades:**

* Investigar sobre patrones de diseño J2EE para el desarrollo de aplicaciones en Java.
* Investigar las APIs de Java para envío de correos electrónicos.
* Investigar las APIs de Java para encriptación de textos.
* Modelar la estructura de clases necesarias para implementar la historia de usuario, considerando lógica del negocio y relaciones entre los componentes.
* Modelar los datos que deben ser persistentes y diseñar la estructura de base de datos.
* Diseñar la interfaz gráfica principal de la aplicación.
* Diseñar el formulario de creación de usuarios.
* Implementar un mecanismo para enviar correos electrónicos desde la aplicación.
* Implementar un mecanismo para encriptar textos desde la aplicación.
* Implementar el mecanismo de guardado para nuevos usuarios.
* Verificar los resultados mediante los casos de prueba.

### Iteración 2

### Iteración 3

### Iteración 4

### Iteración 5

### Iteración 6

### Iteración 7

### Iteración 8

### Iteración 9

## PLANIFICACIÓN DE LAS PRUEBAS

### Casos de Prueba

### Pruebas Unitarias

# [IMPLEMENTACIÓN](#TablaContenido)

## DEFINICIÓN DE LA ARQUITECTURA

La arquitectura de la aplicación es una arquitectura basada en capas, cada capa cumple una función específica y dispone de componentes especializados para su correcto funcionamiento. La arquitectura establecida para la aplicación se asemeja a la arquitectura de la plataforma J2EE y de hecho se basa en dicha arquitectura, ya que la mayoría de herramientas usadas para el desarrollo son hechas en lenguaje Java.

La siguiente imagen describe a grandes rasgos la arquitectura usada para la aplicación.

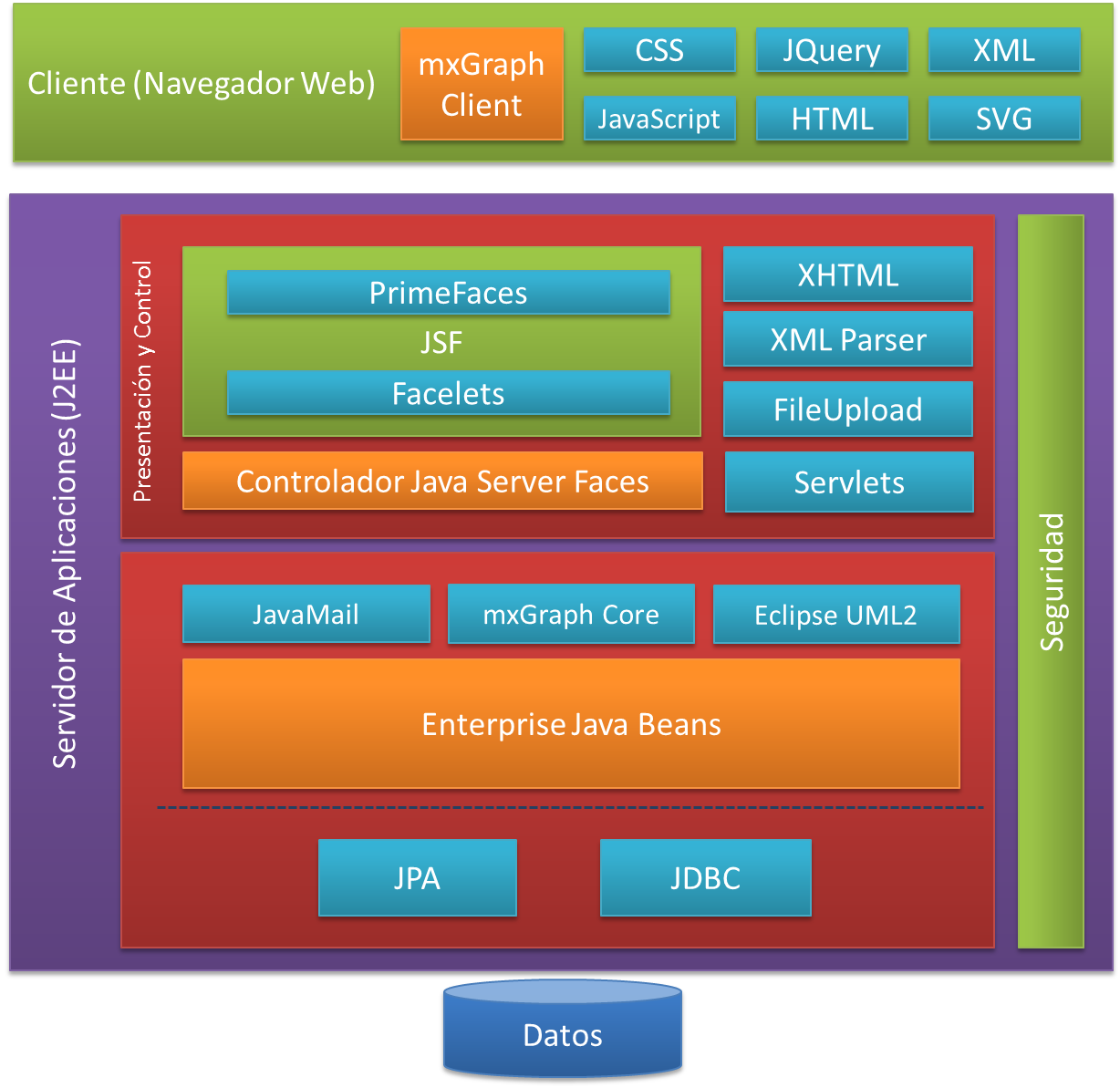


Ilustración . Arquitectura de la aplicación

La imagen anterior se divide en 3 grandes grupos: *Cliente*, *Servidor* y *Datos*; estos representan los niveles de la aplicación *CLASS Modeler*, cada nivel interactúa con los otros mediante interfaces definidas y cada uno tiene responsabilidades propias.

El nivel *Cliente* representa la interfaz mediante la cual un usuario interactúa con la aplicación. Básicamente estamos hablando de una Interfaz Gráfica de Usuario (GUI) a la cual un usuario de la aplicación accede y mediante la cual se comunica con los otros componentes. Este componente puede ser personificado por cualquier Navegador Web disponible en el mercado, siempre y cuando posea las características necesarias para ejecutar la aplicación.

El nivel *Servidor* representa el Backend de la aplicación, desde y hacia la cual fluyen los datos manejados por la misma. Las tecnologías usadas para el desarrollo de la aplicación requieren que se disponga de un servidor especializado conocido como Servidor de Aplicaciones o Middleware, este tipo de software tiene características especiales más que el solo recibir y procesar peticiones HTTP.

Por último, el nivel *Datos* representa el almacenamiento de los datos de la aplicación de manera persistente, este es personificado por un motor de bases de datos del mercado, más adelante en la sección 8.4 se describe con mayor detalle.

A su vez, cada componente se divide en diferentes capas, tal como se mencionó anteriormente, la arquitectura de la aplicación está basada en capas. A continuación se describen cada una de las capas.

* **Capa de Vista**: Esta capa se encuentra en el Cliente, básicamente se encarga de presentar los datos de la aplicación de una manera agradable al usuario, adicionalmente se encarga de tomar las acciones realizadas por el usuario y enviarlas al Servidor, en donde serán procesadas y se generará una respuesta en base a los datos.
* **Capa de Control y Presentación**: Esta capa se encuentra en el Servidor, se encarga de tomar los datos enviados desde la vista, procesarlos y validarlos, de tal manera que se envíen a la capa de negocio donde reposa la lógica necesaria para determinar cuál debe ser la respuesta esperada para determinada acción. Adicionalmente se encarga de tomar la respuesta de la capa de negocio, generar
* **Capa de Lógica de Negocio**: Contiene la lógica de operación de la aplicación, define los procesos y procedimientos realizados por la aplicación según las acciones del usuario.
* **Capa de Integración**: También conocida como la capa de acceso a datos, esta capa se encarga de la comunicación con el repositorio de datos de la aplicación.

En la imagen vista anteriormente, adicional a la arquitectura de la aplicación, se mencionan algunas tecnologías empleadas en cada capa, en la siguiente sección se explica con más detalle cada una de estas tecnologías y la función que cumplen dentro de la aplicación.

## DEFINICIÓN DE LA TECNOLOGIA

La tecnología seleccionada para la aplicación fue el desarrollo basado en el lenguaje Java, específicamente el desarrollo Web en Java, utilizando algunas características propias del lenguaje enfocadas directamente en este tipo de desarrollo. Adicionalmente se incorporaron algunos Frameworks externos que facilitan el desarrollo de los componentes necesarios para la aplicación. A continuación se mencionaran cada una de las herramientas y tecnologías usadas.

**JSF**

PrimeFaces

XHTML

CSS

JavaScript

Facelets

EJB

MxGraph

Glassfish

## PATRONES DE DESARROLLO

Durante el diseño de la aplicación se tuvieron en cuenta varios patrones que solucionan problemas comunes en los proyectos de software.

Modelo Vista Controlador

Singleton

Data Access Object

Decorating Filter

Front Controller

Facade

## BASES DE DATOS

Los datos de la aplicación se almacenan en una base de datos relacional,

## CONFIGURACIÓN

## ESPECIFICACIÓN DE FUNCIONALIDADES

## DESPLIEGUE

# [ASEGURAMIENTO DE LA CALIDAD](#TablaContenido)

## CASOS DE PRUEBA

## PRUEBAS DE UNIDAD

## ANALISIS ESTATICO DE CODIGO

# [CONCLUSIONES](#TablaContenido)

# REFERENCIAS BIBLIOGRÁFICAS

1. **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.

2. **Sommerville, Ian.** *Ingenieria de Software.* Madrid : Pearson Education S.A., 2005. ISBN 8478290741.

3. **Rumbaugh, James, Booch, Grady y Jacobson, Ivar.** *El lenguaje unificado de modelado, Manual de Referencia.* Madrid : Pearson Education S.A., 2000. ISBN 8478290370.

4. **Dominguez, José Alberto.** UML: Lenguaje Unificado de Modelado. [En línea] 14 de Marzo de 2009. [Citado el: 8 de Octubre de 2011.] http://www.que-informatica.com/index.php/tag/uml.

5. **Pressman, Roger.** *Ingenieria de Software un enfoque práctico.* Mexico : McGraw Hill, 2005. ISBN 9701054733.

6. **Creative Commons.** Atribución No Comercial Compartir igual a 2.5 (Colombia). [En línea] http://creativecommons.org/licenses/by-nc-sa/2.5/co/legalcode.

7. **Sun Microsystems.** Licencia de la tecnologia Java SE. [En línea] http://www.bhweb.es/cdlibre/pdf/jvm.pdf.

8. **Free Software Fundation Inc.** GNU General Public Licence. [En línea] 23 de Junio de 2007. http://www.gnu.org/licenses/gpl.txt.

9. **Universidad Francisco de Paula Santander, Programa de Ingenieria.** Pensum académico vigente. [En línea] 2011. http://ingsistemas.ufps.edu.co/images/Descargas/pensum115.pdf.

10. **StarUML.** StarUML 5.0 Guía de Usuario (Versión en Ingles). [En línea] http://staruml.sourceforge.net/docs/user-guide(en)/toc.html.

11. **Canós, Jose H., Letelier, Patricio y Penadés, Carmen Maria.** *Metodologías Ágiles para el desarrollo de software.* Valencia : Universidad Politécnica de Valencia.

1. CASE (Computer Aided Software Engineering, Ingeniería de software asistida por computadora), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo los costos del mismo en términos de tiempo, esfuerzo y dinero. [↑](#footnote-ref-1)
2. Página web oficial de Enterprise Architect, <http://www.sparxsystems.com/> [↑](#footnote-ref-2)
3. Tomado del portal web del programa Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, en la sección Presentación del Programa – Objeto de Estudio.

   Disponible en : <http://ingsistemas.ufps.edu.co/presentacion/objeto-estudio.html> [↑](#footnote-ref-3)
4. Cloud Computing, definición, características, tipos y ventajas, información disponible en la sección 2.2.2 del documento. [↑](#footnote-ref-4)
5. Google Drive es parte de la nube de servicios proporcionada por la empresa Google para sus usuarios, a través de esta nube los usuarios pueden disponer de un kit de herramientas entre los que se incluye un procesador de texto, hojas de cálculo, diseñador de presentaciones, entre otras. [↑](#footnote-ref-5)
6. Backend se le llama a la parte de la aplicación que se ejecuta en el servidor, la cual está encargada de procesar los datos enviados desde el cliente (Frontend) y generar una respuesta. [↑](#footnote-ref-6)
7. API (“Application Programming Interface”, en español “Interfaz de programación de aplicaciones”), se refiere a un conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizada por otro software. [↑](#footnote-ref-7)
8. GWT es un framework para la construcción de aplicaciones web en lenguaje Java, el cual permite trabajan con la tecnología AJAX (*Asynchronous Javascript and XML*) de forma trasparente para el desarrollador, evitándole tener que lidiar con los aspectos complejos y tediosos de la tecnología; además de ser compatible con la mayoría de navegadores web, lo cual es verdaderamente significativo ya que el código escrito funciona igual para todos. [↑](#footnote-ref-8)
9. ORM Object-Relational Mapping, se refiere la técnica de programación para convertir datos entre el sistema de tipos utilizado en lenguajes de programación orientados a objetos y el utilizado en una base de datos relacional, mediante un motor de persistencia. [↑](#footnote-ref-9)
10. JOYANES AGUILAR, Luis. FERNANDEZ AZUELA, Matilde. Java 2, Manual de programación. Mc Graw Hill. ISBN 8448131932. 2001, p3. [↑](#footnote-ref-10)
11. Más información en: <http://www.oracle.com/technetwork/java/javaee/documentation/index.html> [↑](#footnote-ref-11)
12. Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. [↑](#footnote-ref-12)
13. Universidad Francisco de Paula Santander. Estatuto estudiantil. Acuerdo 065, agosto 26, 1996.

    Disponible en : <http://www.ufps.edu.co/ufpsnuevo/archivos/reglamentacion/acuerdo065.pdf> [↑](#footnote-ref-13)
14. El Object Management Group u OMG (o por su nombre en español, Grupo de Gestión de Objetos) es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización sin ánimo de lucro que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para las mismas. El grupo está formado por diversas compañías y organizaciones con distintos privilegios dentro de la misma. [↑](#footnote-ref-14)
15. *The Unified Modeling Language Reference Manual.* RUMBAUGH James, JACOBSON Ivar, BOOCH Grady. La guía completa del proceso unificado escrita por sus creadores. *Chapter 5*, *Use Case View.* Pág. 63. [↑](#footnote-ref-15)