ISBN: 978-99923-993- 9-2

ESCUELA ESPECIALIZADA EN INGENIERÍA ITCA – FEPADE

DIRECCIÓN DE INVESTIGACIÓN Y PROYECCIÓN SOCIAL

PROGRAMA DE INVESTIGACIÓN APLICADA

INFORME FINAL DE INVESTIGACIÓN

"DESARROLLO DE UNA HERRAMIENTA WEB PARA EL DISEÑO Y CONSTRUCCIÓN DE SOFTWARE ORIENTADO A OBJETOS BAJO UN ENFOQUE PEDAGÓGICO"

SEDES Y ESCUELAS PARTICIPANTES: SEDE CENTRAL
ESCUELA DE INGENIERÍA EN COMPUTACIÓN

DOCENTE INVESTIGADOR RESPONSABLE: ING. CARLOS ENRIQUE LEMUS
ING. JHONY MIKEL ESCOBAR

SANTA TECLA, ENERO 2013

ISBN: 978-99923-993- 9-2

ESCUELA ESPECIALIZADA EN INGENIERÍA ITCA – FEPADE

DIRECCIÓN DE INVESTIGACIÓN Y PROYECCIÓN SOCIAL

PROGRAMA DE INVESTIGACIÓN APLICADA

INFORME FINAL DE INVESTIGACIÓN

"DESARROLLO DE UNA HERRAMIENTA WEB PARA EL DISEÑO Y CONSTRUCCIÓN DE SOFTWARE ORIENTADO A OBJETOS BAJO UN ENFOQUE PEDAGÓGICO"

SEDES Y ESCUELAS PARTICIPANTES: SEDE CENTRAL
ESCUELA DE INGENIERÍA EN COMPUTACIÓN

DOCENTE INVESTIGADOR RESPONSABLE: ING. CARLOS ENRIQUE LEMUS
ING. JHONY MIKEL ESCOBAR

SANTA TECLA, ENERO 2013

Autores
Ing. Carlos Enrique Lemus
Ing. Jhony Mikel Escobar

005.713
L562d Lemus, Carlos Enrique
Desarrollo de una herramienta web para el diseño y construcción de software
sv orientado a objetos bajo un enfoque pedagógico / Carlos Enrique Lemus, Jhony Mikel Escobar. --1ª ed. --San Salvador, El Salvador: ITCA Editores, 2013.

58 p.: il. ; 28 cm.
ISBN: 978-99923-993- 9-2

1. Programación (Computadores electrónicos). 2. Programas para computador.
I. Escobar, Jhony Mikel, coaut. II. Escuela Especializada en Ingeniería ITCA-FEPADE.

Sitio web: www.itca.edu.sv
Correo electrónico: bibliotecologos@itca.edu.sv
Tiraje: 16 ejemplares
PBX: (503) 2132 – 7400
FAX: (503) 2132 – 7423
ISBN: 978-99923-993- 9-2
Año 2013

ÍNDICE

1. INTRODUCCIÓN 4
2. PLANTEAMIENTO DEL PROBLEMA 4
   2.1. DEFINICIÓN DEL PROBLEMA 4
   2.2. ANTECEDENTES 5
   2.3. JUSTIFICACIÓN 6
3. OBJETIVOS 6
   3.1.1. OBJETIVO GENERAL 6
   3.1.2. OBJETIVOS ESPECÍFICOS 6
4. HIPÓTESIS 7
5. MARCO TEÓRICO 7
6. METODOLOGÍA DE LA INVESTIGACIÓN 21
7. RESULTADO Y ALCANCES 22
8. CONCLUSIONES 23
9. RECOMENDACIONES 23
10. GLOSARIO 24
11. REFERENCIAS BIBLIOGRÁFICAS 24

ANEXOS 25
ANEXO NO. 1 26
ANEXO NO. 2 39
ANEXO NO. 3 52
ANEXO NO. 4 56

1. INTRODUCCIÓN

El presente documento recopila toda la información requerida para el diseño y elaboración de una herramienta CASE que permita dar soporte pedagógico a las materias básicas de la carrera de Técnico de Ingeniería en Sistemas Informáticos, impartida en la Escuela de Computación. Para lo cual se estudiaron alternativas de arquitectura en ambiente de escritorio y web; sin embargo se destacó el ambiente web por el eficiente manejo de recursos y fácil acceso.

El objetivo principal es brindar una herramienta a los estudiantes y docentes para realizar ejercicios y evaluaciones relacionados con el diseño de software desde una plataforma virtual en la que se pueda gestionar actividades relacionadas con el diagrama de clases, los cuales pueden contener estructuras UML básicas como interfaces, clases y sus respectivas relaciones tales como: agregación, asociación y composición. La presente herramienta web permite además almacenar y recuperar los diagramas realizados a través de la combinación de arquitecturas web como Ajax, CSS3, HTLM5 y Javascript. Esto permite brindarle al usuario un entorno capaz de trabajar con la misma flexibilidad y dinamismo que una aplicación de escritorio, combinando además ventajas como la fácil centralización y despliegue de una arquitectura web.

2. PLANTEAMIENTO DEL PROBLEMA

2.1. DEFINICIÓN DEL PROBLEMA

Debido a la naturaleza de enseñanza a la que debe someterse un estudiante en una carrera técnica es imprescindible adquirir conocimientos actualizados de forma práctica, por lo cual es necesario apoyarse en herramientas pedagógicas que ayuden desde un inicio a simular los entornos de trabajo. Sin embargo en el área de desarrollo de software resulta muy difícil ya que los paradigmas de desarrollo y lenguajes de programación se mantienen en una constate evolución, encaminados a buscar soluciones cada vez más complejas. Esto demanda que las herramientas pedagógicas tengan que adaptarse mejor a nuevas exigencias e integren todos los elementos necesarios que le permitan comprender al estudiante el dominio de técnicas de desarrollo de software como lo es la programación orientada a objetos.

En la actualidad existen herramientas de software que permiten suplir dicha necesidad de modelado y generación de código; sin embargo resultan herramientas de altos costos en

licencias o inversión en tiempo de capacitación al personal debido al grado de complejidad en su uso; y además dichas herramientas carecen de un enfoque pedagógico.

Por lo cual se pretende desarrollar una herramienta personalizada que atienda los problemas y necesidades de nuestra población estudiantil con toda la potencia y calidad de una herramienta CASE, esto permitirá reducir la curva de aprendizaje en los actuales paradigmas de desarrollo.

2.2. ANTECEDENTES

Desde el año 2003 la Universidad Centro Americana del Salvador (UCA) ha trabajado en proyectos de esta naturaleza, en el año 2011 presentó en CONCAPAN el proyecto Aegoli el cual permite sustituir exitosamente al software DFD implementado en módulos asociados al desarrollo de lógica de programación. Sin embargo se observo que solamente puede utilizarse bajo programación estructurada y no cuenta con soporte para la programación Orientada a Objetos por lo que se iniciaron las gestiones a partir del 2011 en el desarrollo de una herramienta con un enfoque completamente actualizado orientada en la aplicación de áreas competentes a la gestión de calidad y desarrollo de software.

Estado del Arte[1]

Las primeras intervenciones de UML se desarrollaron cerca de 1994 cuando Jim Rumbaugh se une con Grady Booch en Rational Software Corporation con la finalidad de unir sus métodos OMT y Booch respectivamente. Esto dio paso a la versión 0.8 del entonces llamado Método Unificado (1995), Incorporando a Ivar Jacobson, luego en 1997 con el apoyo de IBM, Oracle y Rational Software se presentó la versión 1.0 de UML considerándose como un estándar avalada por OMG.

De esta manera surge el Lenguaje Unificado de Modelado y es aceptado como un estándar para modelar software que a la fecha es utilizado por grandes empresas para la producción de software orientado a objetos. En el desarrollo de software este lenguaje de modelado se acopla a un proceso que se aplica durante el ciclo de vida del software, el cual tiene sus orígenes en 1967 basado en el modelo de Ericsson y cuyo pionero fue Ivar Jacobson quien luego en 1987 funda Objectory AB interpretada como “fábrica de Objetos”. En este entonces el lenguaje de descripción y especificación utilizado era SDL (Specification and Description Languages) considerado como el lenguaje estándar para el

[1]Datos recolectados de Tesis: Herramientas case para la generación de código C++ a partir de diagramas de clase UML

modelado de objetos. Más tarde 1995, Rational Software Corporation luego perteneciente a IBM, compro Objectory AB para incorporar las primeras versiones de UML en Objectory 4.1. Este fue evolucionado gracias a la intervención de muchas empresas para dar paso en Junio de 1998 a Rational RUP (Proceso Unificado de Rational), este soporta todo el ciclo de vida de un sistema de software; lo cual abrió el camino para que se lanzaran al mercado muchas herramientas CASE que permiten, a partir de este estándar optimizar el proceso de desarrollo del Software; siendo estas la base principal sobre la cual se centra la investigación actual.

2.3. JUSTIFICACIÓN

En la programación tradicional existen una serie de técnicas y herramientas de software que guían al estudiante en su aprendizaje; tales como la creación de algoritmos, creación de diagramas de flujo y elaboración de pseudocódigo. Sin embargo estas no se aplican para el aprendizaje de la programación orientada a objetos, por lo que el presente proyecto brindará una alternativa didáctica que asocie el modelado de la lógica de negocio directamente a su equivalente en código fuente.

El mayor aporte que se podrá ofrecer a la comunidad estudiantil, será la posibilidad de distribuir dicha aplicación de forma gratuita ya que las actuales herramientas de pago requieren una gran inversión en licenciamiento y hardware, lo que en muchos casos impide su aplicación en Instituciones educativas a nivel superior. Beneficiando directamente a nuestra institución ya que se podrá implementar en módulos de carreras técnicas como: Análisis y Diseño de Sistemas, Programación Orientada a Objetos, Selección de Técnicas de Ingeniería del Software y en carreras de Ingeniería.

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Desarrollar una interfaz gráfica para generar diagramas orientado a objetos a partir de un modelado UML.

3.2. OBJETIVOS ESPECÍFICOS

- Desarrollar investigación preliminar del funcionamiento de compiladores y lenguajes de bajo nivel
- Construir el código fuente del software en base al diseño elaborado en su fase previa.

- Realizar las pruebas al software con sus respectivas revisiones técnicas formales.
- Documentar el software
- Realizar pruebas al software

4. HIPÓTESIS

A través de la presente investigación se pretende mejorar aspectos de calidad y optimización de procesos, en la técnica aplicada por los estudiantes en el desarrollo de software utilizando el paradigma de la programación orientada a objetos. Además se pretende reducir la curva de aprendizaje actual en la enseñanza de módulos asociados al desarrollo de software.

5. MARCO TEÓRICO

La era de información demanda cada vez mas soluciones agiles y adaptables a problemas cada vez más complejos, representando un reto a los actuales programadores debido a las exorbitantes cantidades de información y requerimientos demandados. La disciplina encargada de brindar el apoyo sustentable en materia de herramientas y métodos que permitan crear software de calidad es la Ingeniería del Software.

La Ingeniería del Software es la ciencia que ayuda a elaborar sistemas con el fin de que sea económico, fiable y funcione eficientemente sobre las maquinas reales [Pressman 93]. A través de la ingeniería del software los analistas pueden desarrollar software bajo métodos estándares que aseguran la competitividad, automatización y calidad del software.

La ingeniería del Software Asistida por Computada (CASE) ha permitido que las grandes empresas solventen las actuales exigencias a través de la optimización de código fuente y ahorro de trabajo, todo esto gracias a la oportunidad de unir algunas de las fases del ciclo de vida del desarrollo del software.

Historia de las Herramientas CASE.

Surgen a principios de los 70's con el procesador de palabras que se usaba en la creación documentos. Los principales usos que se dieron fue en soporte a programas como: traductores, compiladores, ensambladores y procesadores de macros [SEI 99], luego surgieron las herramientas orientadas a diagramas, los primeros fueron: Entidad – Relación, diagramas de Flujo, editores de programas, depuradores, analizadores de código y utilidades de impresión, como los generadores de reportes y documentación.

Entre los años 1980 y 1990 surgieron herramientas CASE que pretendían resolver de forma separada las necesidades de software, de la siguiente manera:

- Herramientas CASE para Desarrollo Orientado a Objetos: Ayudaban a crear código reutilizable en diferentes lenguajes y plataformas.
- Herramienta de Desarrollo Visual: Estas herramientas permiten al desarrollador construir rápidamente interfaces de usuario, reportes y otras características de los sistemas, lo que hace posible que puedan ver los resultados de su trabajo en un instante, El siguiente diagrama muestra de forma resumida la historia de las herramientas CASE a lo largo de los años:

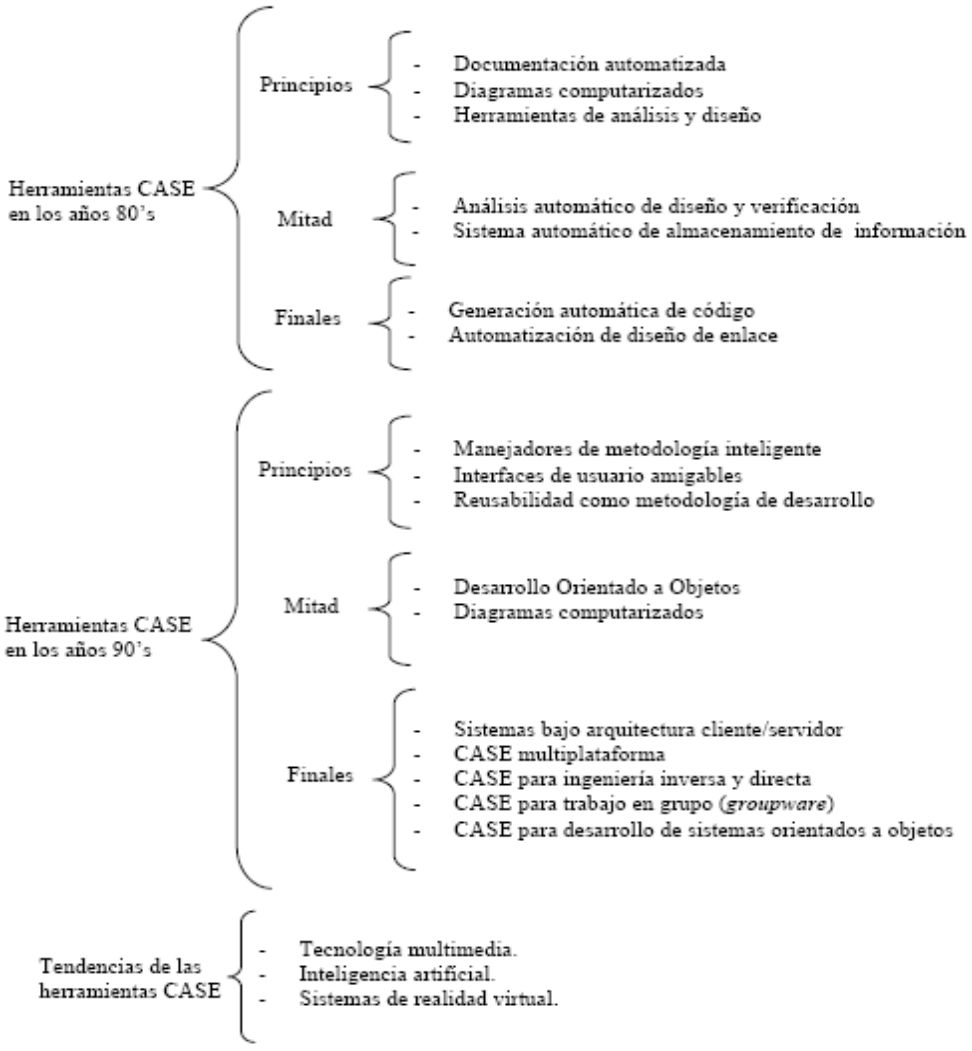


Imagen 1. Componentes de diversas Herramientas CASE.

Aplicación de herramientas CASE.

Las herramientas CASE tienen como objetivo el soporte a las fases del ciclo de vida del desarrollo de software; algunas de ellas se centran en las primeras fases proporcionando ayuda para el dibujo de diagramas y creación de pantallas. Otras se enfocan en las fases de implementación, incluyendo codificación automatizada y generadores de prueba.

Las CASE también permiten el apoyo computacional para el diseño del software lo que permite detección de errores en el modelado y generación de código.

Ventajas de las herramientas CASE.

Las herramientas CASE tienen varias ventajas que les permiten mejorar aspectos puntuales asociados al ciclo de vida del Desarrollo de Software, los cuales se detallan en la siguiente tabla:

Aspectos	Descripción
Facilidad para la revisión de aplicaciones	El desarrollo de diagramas que modelan un software permite la actualización de nuevos procesos de forma rápida y fiable puesto que no se altera directamente el código fuente.
Ambiente interactivo en el proceso de desarrollo	Las herramientas CASE permiten un interacción directa con la codificación de aplicaciones esto permite obtener una idea mucho más clara del resultado esperado en los diagramas.
Generación de código	La generación de código automático permite acelerar el proceso de desarrollo al optimizar el tedioso proceso de diagramar y luego codificar, además asegura una estructura consistente lo que disminuye ocurrencia de varios tipos de errores y ayuda en la fase de mantenimiento.
Reducción de Costos	Las empresas se ven en la necesidad de contratar personal adicional para generar vistas de pantallas y reportes con el fin de mostrar la interfaz final del usuario y por lo general en las etapas de mantenimientos es común realizar cambios que al final se transforman en costos de tiempo y dinero
Incremento en la productividad	En la utilización de herramientas CASE se observa un claro incremento en la productividad ya que permiten administrar los recursos tecnológicos y humanos de forma eficiente.

Tabla 1. Aspectos relevantes de una herramienta CASE.

En la actualidad las herramientas CASE son parte fundamental en grandes empresas que necesitan desarrollar software para suplir de forma rápida y eficiente procesos que generarían altos costos de mantenimiento y actualización del software operativo. Pese a su alto precio aun resultan ser una atractiva alternativa para el desarrollo de software ya que empresas que desarrollan sus operaciones a niveles internacionales no pueden permitir

errores en sus procesos operativos puesto que podría causar pérdidas millonarias en algunos casos.

Desventajas de las herramientas CASE.

Las herramientas CASE también presentan algunos puntos débiles que deberán conocerse y tomarse en cuenta, a continuación se muestran los más relevantes:

Aspectos	Descripción
El costo	Uno de los principales aspectos que no permite la adquisición de este tipo de herramientas es el costo ya que algunas herramientas se encuentran entre $500 hasta más de $4000.00 aproximadamente, esto sin tomar en cuenta costos de mantenimiento y entrenamiento. Por lo que las instituciones deberán tener mucho cuidado y considerar si los beneficios son mayores al momento de implementarlas.
Preparación del personal	Para poder obtener el provecho posible es necesario entrenar al personal de forma correcta tanto en el uso de las herramientas como en el lenguaje de modelado y metodología asociada, esto implica costos adicionales tanto en recurso humano como en tiempo.
Métodos estructurados	Las herramientas CASE más utilizadas se aplican a una metodología estructurada lo que limita al desarrollador a incurrir solamente en esta metodología, desaprovechando los beneficios que conllevan las metodologías actualizadas.
Función limitada	Aunque la idea de una herramienta CASE es apoyar muchas fases del Ciclo de Vida del Desarrollo de Sistemas por lo general estas tienden a centrarse en una fase en específico lo que implica en ocasiones implementar más de una CASE.

Tabla 2. Desventajas de una Herramienta CASE.

El uso de herramientas CASE en muchos casos no resultaron ser una alternativa fiable y efectiva debía a que en muchas empresas los costos de mantenimiento y adquisición representaban un gasto mucho más alto de los beneficios que estas pueden aportar.

Componentes

Las herramientas CASE se encuentran conformadas por una serie de elementos que las hace funcionar, estos se presentan a continuación:

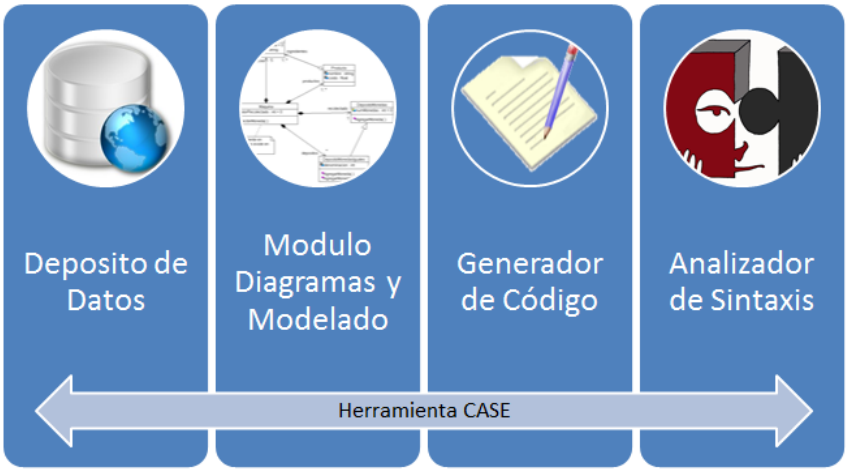


Figura 1. Representación gráfica de los componentes de una Herramienta CASE.

Depósito de Datos.

Este depósito permite almacenar la información creada y generada a través de las etapas que cubre la herramienta de determinado proceso de desarrollo, entre estos datos están: componentes de análisis y diseño, diagramas de entrada – salida, diagramas de clase, etc. Es también conocido como Diccionario de Recursos de Información.

El depósito de datos coordina, integra y estandariza los diferentes componentes de información, entre sus principales características están: [INEI 99].

- *Tipo de metodología que soporta.*
- *La forma de actualización de los datos.*
- *La reutilización de módulos para otros diseños.*
- *Compatibilidad del depósito de datos***.**

Módulo de Diagramas y Modelado.

Consiste en dar soporte para la creación de los diagramas más utilizados según el paradigma de programación, en el caso de la programación estructurada se auxilia de los diagramas de entidad-relación, flujo de datos, etc. Para el caso de los diagramas aplicables en UML se pueden mencionar los diagramas de clase, estado, casos de uso, etc.

Las características que hacen más fácil o más poderosa a una herramienta en cuanto al soporte de diagramas son: [INEI 99]

- *Número máximo de niveles permitidos en un diagrama***.**
- *Número máximo de objetos que permite en un diagrama.*
- *Posibilidad de deshacer los cambios hechos a los diagramas.*

Analizador de sintaxis.

Permite verificar la validez y el correcto uso de la información introducida a la herramienta de acuerdo a la metodología o lenguaje soportado por la herramienta. Este componente es vital ya que ayuda al desarrollador a prever errores de sintaxis al momento de crear los diagramas y generar un código completo y preciso.

Lenguaje Unificado de modelado (UML)

En la actualidad la Ingeniería del Software a inclinado su tendencia al desarrollo de Sistemas Orientados a Objetos por la factibilidad y fiabilidad que estos presentan; es por esta razón, que la Ingeniería del Software ha desarrollado diversas metodologías para proporcionar un soporte estándar para analizar y diseñar con más precisión estos sistemas informáticos.

- *¿Qué es el UML?*

  Es un Lenguaje de modelado grafico y visual utilizado para especificar, visualizar, construir y documentar los componentes de un sistema de software. El UML permite captar la información estática del sistema proporcionada sobre los objetos que intervienen en un determinado proceso y la relación entre ellos, también nos permite captar la información dinámica del sistema que define el comportamiento de los objetos a lo largo de todo el tiempo en el que estos interactúan hasta llegar a sus objetivos.

  La ventaja principal del UML [Coleman 97] sobre otras notaciones orientadas a objetos es que elimina las diferencias entre semánticas y notaciones.

- *Las vistas de UML*: Se refieren a la forma en la que se modela una parte del sistema a desarrollar. En la siguiente figura se muestra una clasificación de los diagramas que proporcionan las vistas de UML (vale la pena aclarar que esta es una propuesta y que cada desarrollador puede crear su propia clasificación) [Schmuller 00].
- Diagrama UML que proporcionan una visión estática del sistema.

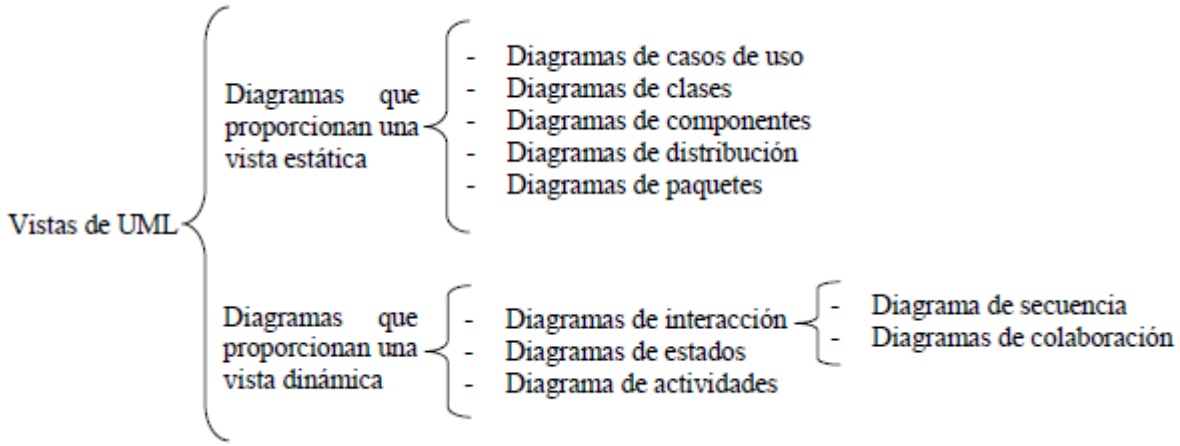


Figura 2. Diagramas que componen el lenguaje UML.

Diagrama de Casos de Uso: Nos permite modelar la forma en cómo los actores interactuan con el sistema. El propósito de esta vista es realizar una descripción lógica de una parte funcional del sistema y no del sistema en su totalidad. Este diagrama consta de elementos estructurales y relacionales.

Los elementos estructurales de los casos de uso representan las partes físicas o conceptuales de un modelo, en la siguiente tabla se muestran los elementos estructurales de un caso de uso [Rumbaugh 00] y [Schmuller 00].

 Actor	**Actor:** Un actor es un rol que un usuario juega cuando interacciona con el sistema, es un comportamiento específico frente a tal sistema. Un actor no necesariamente representa una persona en particular, sino más bien la labor que realiza ante el sistema. Puede ser un humano, otro sistema de software o algún otro proceso.
 Caso de uso	**Casos de Uso:** Un caso de uso es una acción o tarea especifica que el sistema lleva a cabo tras una petición de un actor o de otro caso de uso. Un caso de uso conduce a un estado observable de interés para un actor.

Tabla 3. Componentes de un diagrama de caso de uso.

Las relaciones conectan a los elementos estructurales para darle sentido al diagrama de casos de uso, estas relaciones se muestran en la tabla siguiente.

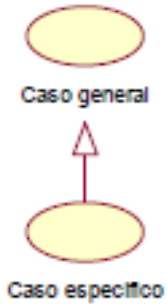
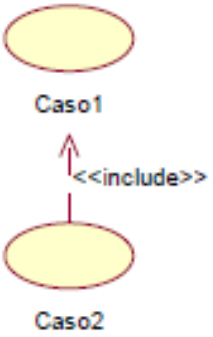
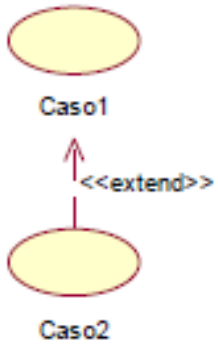
	**Asociación:** La asociación es la relación más simple del UML, es la relación que se da entre el actor y el caso de uso, o entre dos casos de uso, se denota como una línea dirigida entre ellos.
	**Generalización:** Este tipo de relación se da entre un caso de uso general y un caso de uso más específico. Donde este ultimo hereda propiedades del primero y agrega sus propias acciones, se denota como una línea continua con una punta de flecha en forma de triangulo sin rellenar.
	**Inclusión:** Este tipo de relación se da entre casos de uso cuando se tiene una parte de comportamiento que es similar en más de un caso de uso, y no se desea copiar la descripción de tal conducta para cada uno de ellos o bien cuando un caso de uso necesita utilizar otro caso de uso. Se denota con una línea discontinua con una punta de flecha y con la palabra “include”
	**Extensión:** Esta relación se da también solo en casos de uso cuando se tiene un caso de uso que es similar a otro, pero que hace un poco más, se recomienda utilizarlo cuando se describa una variación de una conducta normal. Se denota con una línea discontinua con una punta de flecha y con la palabra “extend”

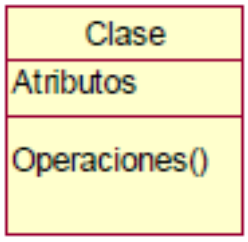
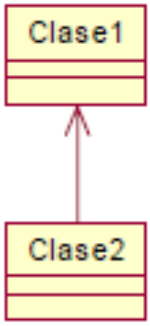
Tabla 4. Relaciones asociadas a los casos de uso.

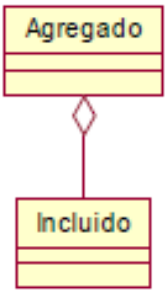
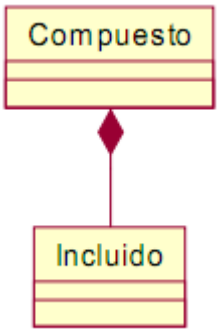
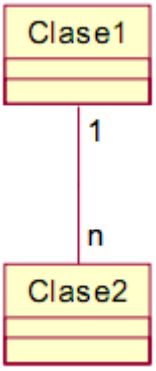
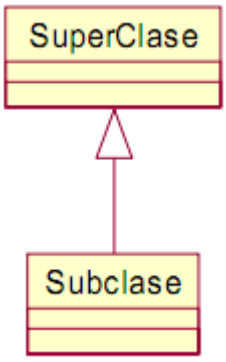
Diagrama de Clases: La vista de los diagramas de clase, visualiza las relaciones entre las clases que se involucran en el sistema. Un diagrama de clases muestra un conjunto de clases,

interfaces, colaboraciones y las relaciones entre estas, mostrando así la estructura estática de un sistema. Los diagramas de clases pueden mostrar.

- Clases.
- Atributos.
- Operaciones.
- Asociaciones.
- Generalizaciones.
- Agregaciones.
- Composiciones.

En la siguiente tabla se describen los elementos que confirman el diagrama de clases.

<table>
<tr>
<td rowspan="3"> 
</td>
<td>Clase:
Es la unidad básica que encapsula toda la información que comparten los objetos del mismo tipo. A través de una clase se puede modelar el entorno del sistema. En UML una clase se representa como un rectángulo que posee tres divisiones.
Superior: Contiene el nombre de la clase.
Intermedio: Contienen los atributos.
Inferior: Contiene las operaciones.</td>
</tr>
<tr>
<td>Atributos:
Se utilizan para almacenar información, estos atributos tienen asignados un tipo de visibilidad que define el grado de comunicación con el entorno de acuerdo con la programación orientada a objetos tenemos tres tipos pública (public), protegida (protected), privadas (prívate).</td>
</tr>
<tr>
<td>Operaciones:
Son la forma en cómo esta interactúa con su entorno, estas también pueden tener uno de los tipos de visibilidad mencionadas en los atributos.</td>
</tr>
<tr>
<td> 
</td>
<td>Asociación:
Es la relación entre las instancias de las clases, permite relacionar objetos que colaboran entre sí. La asociación puede ser unidireccional o bidireccional. Cuando es unidireccional significa que solo existe comunicación de la clase de la que parte la flecha hacia la que apunta. En caso que sea bidireccional significa que existe comunicación entre ambas clases.</td>
</tr>
</table>

	**Agregación:** Es un tipo especial de asociación, con la cual se pueden representar entidades formadas por varios componentes. La agregación es una relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye, en la programación orientada a objetos se dice que ésta es una relación por referencia. Se representa con una flecha con punta de rombo en blanco en el extremo del compuesto o base.
	**Composición:** Es similar a la agregación, ya que también representa entidades formadas por componentes, solo que esta relación es estática, en donde el tiempo de vida del objeto incluido está condicionado por el tiempo de vida del que lo incluye ya que el objeto base se construye a partir del objeto incluido, en la programación orientada a objetos se dice que es una relación por valor. Se representa con una flecha con punta de rombo relleno en el extremo del compuesto o base.
	**Multiplicidad:** Cada asociación tiene dos roles o papeles que se encuentran en cada extremo de la línea, cada uno de estos papeles tiene asignada una multiplicidad, la cual indica el número de objetos que pueden participar en la relación, es decir los limites inferior y superior de los objetos que participan, la multiplicidad puede ser. 1..* (1..n) Uno o más * (n) Cero o más M (m es un entero) Número fijo Estos números se colocan sobre la línea de asociación junto a la clase correspondiente.
	**Generalización o Herencia:** Indica que una subclase hereda las operaciones y atributos especificados por la superclase, por ende, la subclase además de poseer sus propios métodos y atributos, poseerá las operaciones y métodos de la superclase siempre y cuando estos tengan una visibilidad pública o protegida. Se denota como una línea continua con una punta de flecha en forma de triangulo sin rellenar que apunta a la superclase.

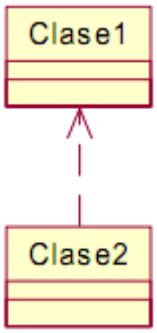
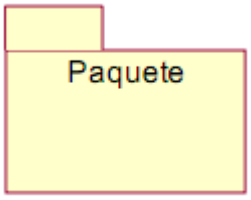
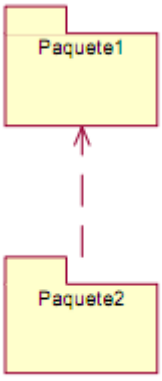
	**Dependencia:** Es un tipo de relación entre dos elementos, donde el uso más peculiar de este tipo de relación es para denotar una dependencia de uso que tiene una clase con otra, en otras palabras un cambio en una de ellas causaría un cambio en la otra. En la dependencia el objeto utilizado no se almacena dentro del objeto que la crea. Se denota como una línea discontinua con punta de flecha formada por dos líneas.

Tabla 5. Componentes de un Diagrama de Clases

Diagrama de Paquetes: Cuando se tiene un sistema muy grande, es conveniente separarlo en piezas pequeñas para poder trabajar más fácil con él, son los diagramas de paquetes los que muestran las relaciones y las dependencias entre los diferentes paquetes en que un sistema está dividido. Este diagrama muestra los siguientes componentes según [Figueroa 97], [Raumbbaugh 00] y [Schmuller 00].

	**Paquete:** Es un mecanismo para organizar un conjunto de elementos. Estos pueden contener otros elementos del modelo como diagramas de clase, diagramas de casos de uso. Comúnmente este paquete representa a un subsistema.
	**Interfaz:** Es un elemento del modelado que define un conjunto de comportamientos a través de operaciones ofrecidas por un elemento, ya sea una clase, un subsistema u otro componente.
	**Dependencia:** La dependencia entre paquetes se da si existiera una dependencia entre dos clases en diferentes paquetes.

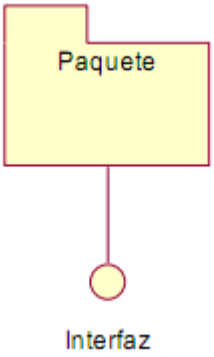
	**Realización:** Es una relación que se da entre un paquete y una interfaz, lo que indica que el paquete define su comportamiento a través de una o más interfaces.

Tabla 6. Componentes de un diagrama de paquetes.

Metodología para el desarrollo de Software.

Como ya se ha mencionado, actualmente la tendencia de los sistemas es construir software lo más rápido posible. Por tal motivo los desarrolladores necesitan de metodologías que permitan dar a los usuarios productos de calidad. Para desarrollar software orientado a objetos la metodología más utilizada es "El Proceso Unificado de Desarrollo de Software", que es una guía de trabajo que se puede adaptar a diferentes tipos de sistemas de software, diferentes campos de acción y diferentes tamaños de proyectos.

El proceso unificado de desarrollo tiene como objetivo proveer una disciplina para la asignación de tareas y responsabilidades dentro de una organización de desarrollo.

Las características que definen el Proceso Unificado de Desarrollo son:

- Esta dirigido por casos de uso: Cada proceso se centra en lo que debe hacer el sistema, comenzando por especificar los requisitos funcionales del sistema y el conjunto de todos los casos de uso forman lo que se conoce como modelo de casos de uso, el cual contiene la descripción de todo el sistema.
- Esta centrado en la arquitectura: Esta metodología describe al sistema en varios puntos de vista. La arquitectura es la forma que tendrá el sistema y sirve como base para comprender como quedará una vez terminado según el análisis hecho de los casos de uso.
- Es interactivo e incremental: Esta metodología nos permite también dividir el proyecto en mini-proyectos, donde a cada uno de estos se le conoce como interacciones, las cuales una vez terminadas significan un incremento en el proyecto.

La Vida del Proceso Unificado del Desarrollo de Software.

Esta metodología está basada en 4 fases que se repiten a lo largo del tiempo de vida del desarrollo del software, cada una de estas fases a su vez, se divide en un conjunto de

interacciones que se repiten a lo largo de una serie de ciclos, donde al término de cada ciclo se obtiene una versión del software listo para ser entregado al cliente. Una interacción puede pasar por 5 flujos de trabajo. En la siguiente figura se muestra los 5 flujos de trabajo que se llevan a cabo durante las 4 fases.

Los 5 flujos de trabajo de la metodología de desarrollo son:

1. Requisitos: Es fundamental para definir una estrategia informática que encaje dentro de las metas del proyecto, en este flujo se identifican todos las necesidades que los usuarios esperan satisfacer con un sistema de software.
2. Análisis: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas.
3. Diseño: En este flujo se define con detalle y precisión lo que se debe programar.
4. Implantación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
5. Pruebas: Busca los defectos a los largo del ciclo de vida.

Las 4 fases de la metodología de desarrollo son:

1. Inicio: En esta fase se obtiene la descripción del problema a resolver mediante los casos de uso.
2. Elaboración: En esta fase se definen y especifican en detalle los casos de uso encontrados en la fase anterior y con ello se diseña la arquitectura del sistema, también en esta fase debemos planificar las actividades y recursos necesarios para realizar el proyecto.
3. Construcción: En esta fase se lleva a cabo la construcción del producto partiendo de la arquitectura creada en la fase anterior y se lleva el trabajo hasta tener el sistema completo (es muy probable que en esta fase el sistema tenga errores).
4. Transición: En esta fase se prueba el sistema y un número de usuarios verifican e informan de posibles errores que este pueda tener; se corrigen los errores para poder entregar un aversión del producto.

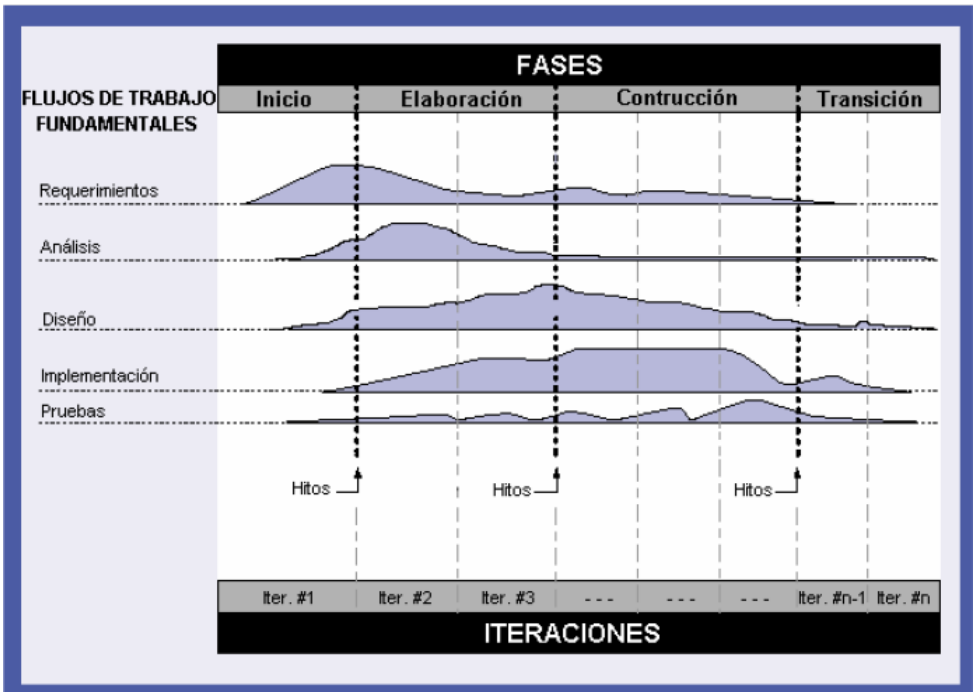


Figura 2. Fases del ciclo de vida del desarrollo de un sistema.

Modelo Vista Controlador

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de negocios de un software en tres componentes distintos.

Descripción del Patrón:

Modelo: Esta es la representación específica de la información con la cual el sistema opera.

Vista: Este presenta el modelo en un formato adecuado para interactuar con e usuario. Usualmente se llama Interfaz de Usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelado y a la vista.

Flujo del Modelo Vista Controlador:

1. El usuario interactúa con la interfaz de usuario de alguna forma.
2. El controlador recibe por parte de los objetos de la interfaz de usuario la notificación de la acción solicitada por el usuario, el controlador gestiona el evento que llega frecuentemente a través de un evento.
3. El controlador accede al modelo y ejecuta la lógica del negocio delegando a los objetos tareas para las que fueron creadas usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador obtiene resultados y los envía a los objetos de la vista asignándoles la tarea de desplegar en la interfaz de usuario dichos resultados.
   En la siguiente grafica se puede apreciar este modelo.

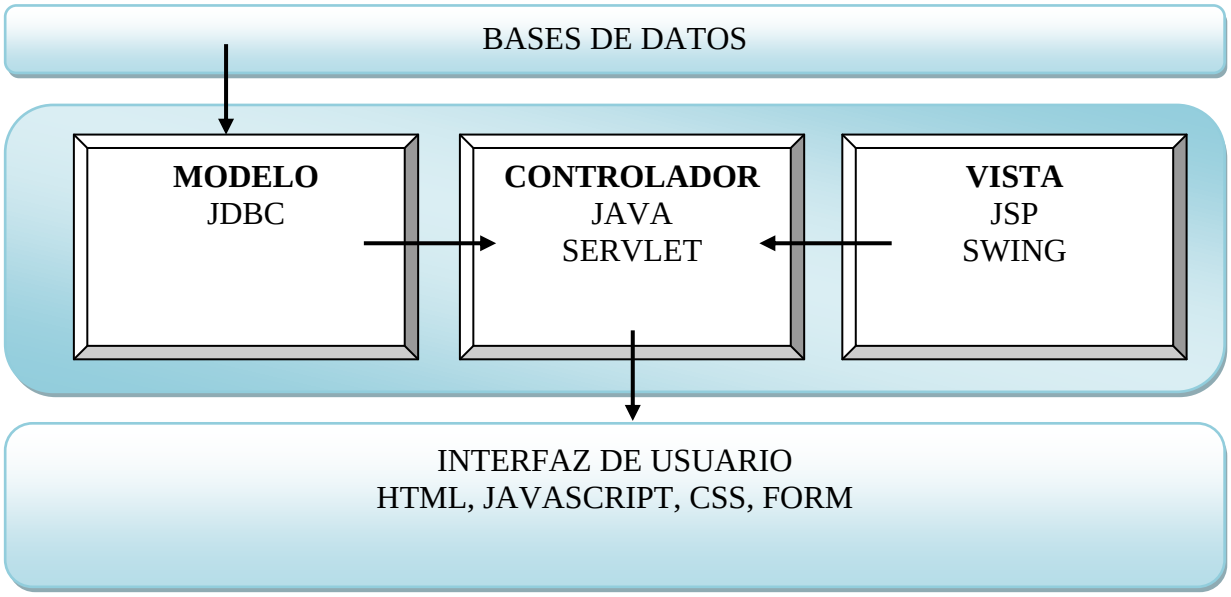


Figura 3. Componentes de una arquitectura Java.

6. METODOLOGÍA DE LA INVESTIGACIÓN

La presente investigación presenta grandes retos en materia de tecnología ya que es necesario comprender el funcionamiento de los compiladores, analizadores de sintaxis y funcionamiento directo de los diferentes diagramas que conllevan el desarrollo de software aplicando una metodología de trabajo.

Por lo cual se pretende distribuir en diferentes fases que producirán los insumos de forma iterativa para realizar la aplicación de forma exitosa, estas se detallan a continuación:

Fase 1: Investigación Tecnológica

En esta fase se deberá recolectar la información necesaria para la comprensión, diseño y desarrollo de compiladores y herramientas case, analizando además alternativas de de lenguajes desarrollo.

Fase 2: Recolección de Datos.

En la recolección de datos se elaborará una serie de instrumentos necesarios para enfatizar y descartar herramientas y controles para el diseño del Sistema tanto para el mercado actual como para suplir necesidades de nuestra comunidad estudiantil, además se tomarán

parámetros actuales de calidad, seguridad y eficiencia de la técnica actual de los estudiantes en materia de programación para posteriormente destacar controles y procesos que permitan mejorarla.

Fase 3: Análisis y Diseño del Sistema.

Con toda la información necesaria acerca de las necesidades tanto del mercado como de la comunidad estudiantil actual de nuestra institución, se procederá a desarrollar el análisis y diseño del sistema. Desarrollando los diagramas necesarios que permitan dar paso a una codificación planeada.

Fase 4: Desarrollo y Prueba del Sistema.

La última fase permitirá obtener el software con sus respectivos manuales de usuario, mantenimiento e Implantación, además se someterá a las pruebas necesarias para asegurar su funcionamiento.

Para la realización de las fases antes descritas se procederá a seleccionar a dos equipos de alumnos que colaborarán con el proyecto, estableciendo previamente reuniones acorde a los horarios de estudio y carga académica.

7. RESULTADO Y ALCANCES

El desarrollo y producción del software se enmarcó en la base principal del desarrollo de software que son los procesos asociados a modelar la lógica de la aplicación, por lo que el software se centra en brindar una alternativa para obtener diagramas enfocados al modelado de clases y sus respectivas relaciones. Por lo que es preciso aclarar que el software no permite cumplir con el ciclo de vida completo de un sistema, puesto que esté está adaptado únicamente para implementar diagramas a partir de UML estático que de soporte a una plataforma pedagógica para suplir necesidades de índole pedagógicas.

Se considera pertinente aclarar además que la herramienta obtenida no pretende manipular diagramas relacionados a base datos ya que se puntualizó en las necesidades detectadas (Anexo 1. Análisis de requerimientos de sistema ITCASE) en la población estudiantil de la Escuela de Computación de ITCA-FEPADE.

Como parte de los entregables del proyecto se detallan a continuación los resultados obtenidos:

- Análisis del requerimiento del software a través de instrumentos de recolección de datos. (Anexo 1)
- Diseño y desarrollo de los componentes del software aplicando los diagramas UML y normas de calidad de software. (Anexo 2)
- Instrumentos para la recolección de datos dirigido a las empresas (Anexo 3)
- Instrumentos para la recolección de datos dirigido a los estudiantes (Anexo 4)

8. CONCLUSIONES

El desarrollo del presente proyecto ha permitido incurrir en un nuevo campo de investigación en materia de desarrollo y gestión de calidad de software, aplicando conceptos de Herramientas CASE de forma innovadora en una arquitectura web, a través de nuevas tecnologías como HTML5, Ajax y Jquery. Permitiendo realizar una herramienta web liviana que puede incorporarse con facilidad en cualquier plataforma virtual o aplicación pedagógica distribuida. Por lo que el proyecto ha permitido sentar las bases para futuras líneas de investigación como lo son los procesos de compilación e interpretación de lenguajes de programación con el objetivo de crear herramientas que reduzcan y optimicen las rutinas de código en la creación de nuevas aplicaciones en el ámbito empresarial y a la ves brinden soporte en el área pedagógica.

9. RECOMENDACIONES

El presente proyecto permite obtener una herramienta actualizada, amigable y fácil de implementar o desplegar en línea, esto permite que el estudiante cuente con los recursos pedagógicos necesarios para desarrollar sus fortalezas en el área de programación, particularmente en este caso enfocada en el análisis de procesos que sirve como base para desarrollar un sistema informático exitoso. Muchos de estos resultados son posibles gracias a tecnología Javascript y HTML 5, que ya está siendo explotada para integrarse en aplicaciones móviles por lo que es recomendable tomar el presente trabajo y sus tecnologías adyacentes para ejecutar nuevos proyectos que a través de diversos frameworks faciliten la ardua tarea de programación.

10. GLOSARIO

Actividad	:	La ejecución de una acción.
Arquitectura	:	Conjunto de decisiones significativas acerca de la organización de un sistema de software, la selección de los elementos estructurales a partir de los cuales se compone el sistema y las interfaces entre ellos, junto con sus comportamientos. La arquitectura del software también se interesa por las restricciones y compromisos de uso, funcionalidad, funcionamiento, flexibilidad al cambio, reutilización, comprensión, economía, tecnología y aspectos estéticos.
CASE	:	Computer Aided Software Engeniering – Ingeniería del Software Asistida por Computadora.
Caso de uso	:	Es una descripción de un tipo de secuencias de acciones, incluyendo variaciones, que un sistema lleva a cabo y que conduce a un resultado observable de interés para un actor determinado.
Ciclo	:	Ciclo de vida del software que cubre las cuatro fases del Proceso Unificado de Desarrollo del Software.
Diagrama	:	Representación gráfica de un conjunto de elementos, usualmente representado por un grafo conectado por vértices y arcos.
Fase	:	Periodo de tiempo entre dos hitos principales de un proceso de desarrollo.
Flujo de trabajo	:	Realización de un caso de uso o parte de él.
Instancia	:	Una manifestación concreta de una abstracción; una entidad sobre la que pueden aplicarse un conjunto de operaciones y que tiene un estado que almacena los efectos de las operaciones; un sinónimo de objeto.
Método	:	La implementación de una operación.
Modelo	:	Una abstracción de un sistema cerrado semánticamente.
Modelo de negocios	:	Técnica que provee una metodología para modelar procesos de negocio basada en la utilización del UML.
Requerimiento	:	Condición o capacidad que debe cumplir un sistema.
Rol	:	Comportamiento especifico de una entidad que participa en un contexto particular.
UML	:	Unified Modeling Languaje – Lenguaje Unificado de Modelado.
Vista	:	Proyección de un modelo, que es visto desde una perspectiva dada o un lugar estratégico, y que omite las entidades que no son relevantes para esta perspectiva.

11. REFERENCIAS BIBLIOGRÁFICAS

[Pressman 93]

Nombre de la obra	:	Ingeniería del Software: un enfoque práctico.
Autor	:	Roger S. Pressman.
Casa editorial	:	McGraw Hill.
País:	:	España.
Año edición	:	1993

[Schmuller 00]

Nombre de la obra : Aprendiendo UML en 24 horas.
Autor : Joseph Schmuller.
Casa editorial : Pearson.
País: : México.
Año edición : 2000

[Rumbaugh 00]
Nombre de la obra : Lenguaje Unificado de Modelado Manual de Referencia.
Autor : Rumbaugh, I. Jacobson, G. Booch.
Casa editorial : Pearson.
País: : Madrid.
Año edición : 2000

Sitiografía:

[SEI 99]
Nombre del sitio : Computer Aided Software Engineering (CASE) Environments
Enlace : http://www.sei.cmu.edu/legacy/case/case_whatis.html
Fecha visita : 9 de febrero de 2012

[INEI 99]
Nombre del sitio : Herramientas case
Enlace : http://www.uap.edu.pe/samples/demo/Libro-5103.pdf
Fecha visita : 9 de febrero de 2012

[Figueroa 97]
Nombre del sitio : Elementos Notacionales de UML
Enlace : http://www.cs.ualberta.ca/~pfiguero/soo/uml/
Fecha visita : 9 de febrero de 2012

12. ANEXOS

Proyecto de Investigación

Desarrollo de una herramienta web para el diseño y construcción de software orientado a objetos bajo un enfoque pedagógico

(ITCASE)

ANEXO No. 1

Informe:

Análisis de requerimientos del Sistema ITCASE

Departamento de Ingeniería en Computación

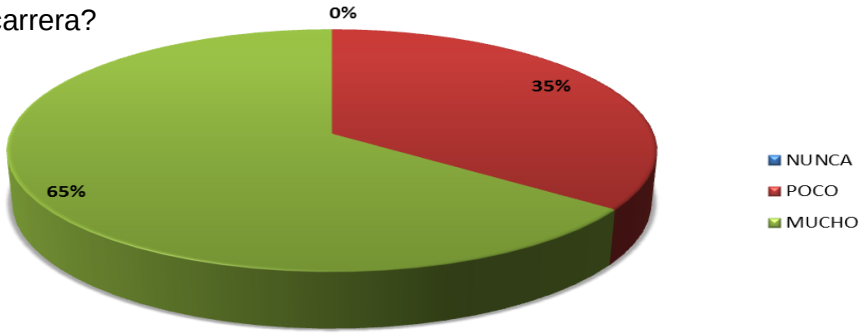
Objetivo: Analizar las alternativas óptimas que permitan generar una herramienta CASE capaz de modelar las necesidades de software a través de diagramas UML bajo un enfoque pedagógico.

Identificación de Requerimientos.

Para poder validar todas las necesidades con las que debe cumplir la presente herramienta, se desarrolló una entrevista a un grupo de 20 profesionales que desempeñan cargos de analistas programadores en diversas entidades gubernamentales y privadas, lo cual permitió recolectar información relacionada a los requerimientos de la herramienta desarrollada.

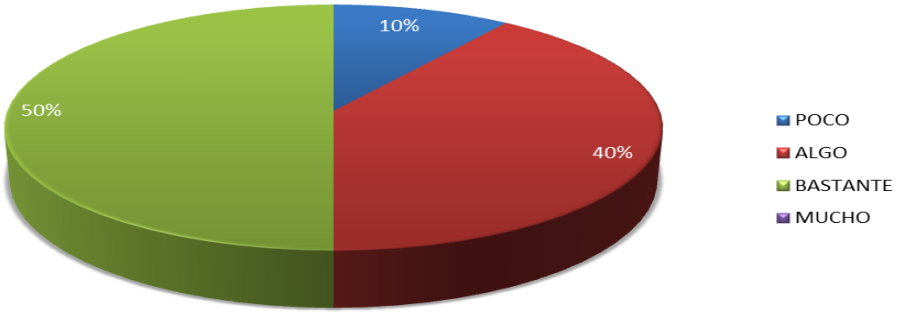
A continuación se presenta la consolidación de datos obtenida a través de las encuetas (ver anexo..):

Pregunta 1: ¿Con que frecuencia utiliza software para el diseño o codificación de sistemas en los módulos asociados a su carrera?



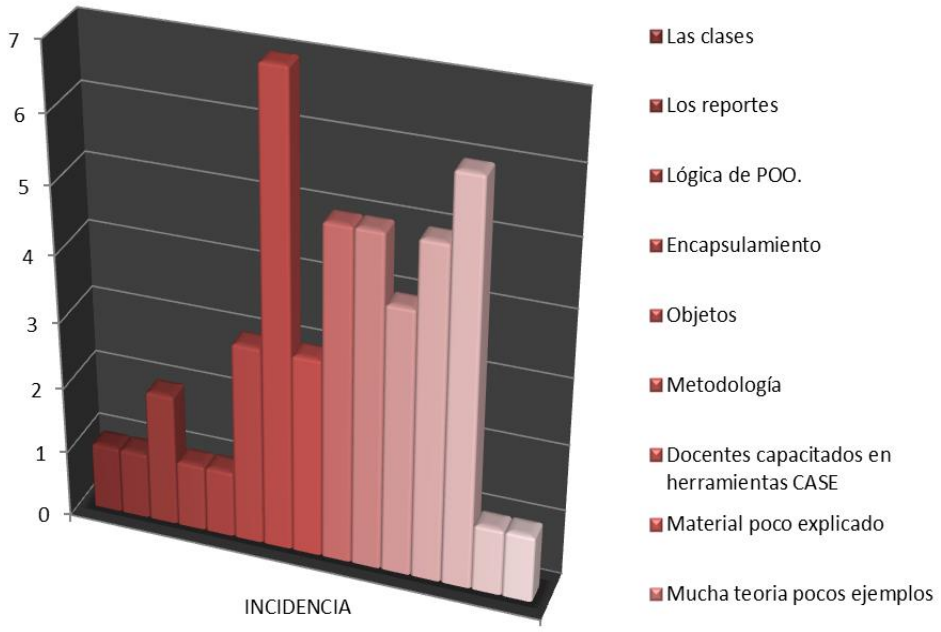
Análisis: Se obtuvo un total de 13 aciertos en la opción que denomina mayor cantidad de veces (mucho) mientras en la opción que denomina la mínima cantidad (poco) se obtuvo una cantidad de 7 puntos. Esto indica que las herramientas de desarrollo de software aún están presentes en las empresas de mayor relevancia en la zona de Santa Tecla.

Pregunta 2: Señale el nivel de dificultad que representa la curva de aprendizaje de los lenguajes de Programación Orientada a Objetos (POO).



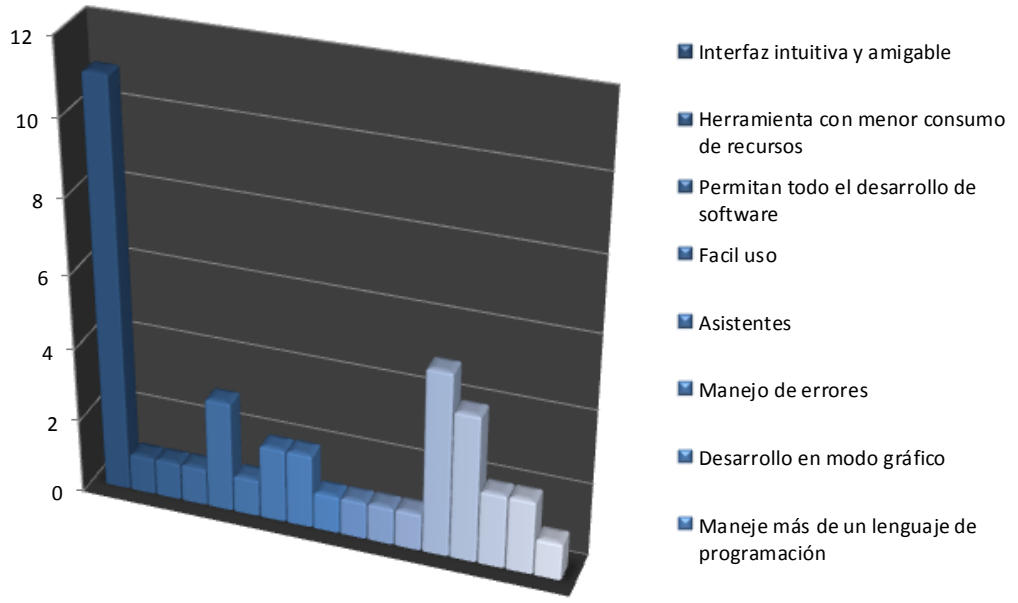
Análisis: El nivel de dificultad en el aprendizaje de lenguajes de Programación Orientada a Objetos (POO) **s**e presentó una puntuación de 10, lo cual representa el 50% de la muestra total; mientras que en las categorías restantes se obtuvieron 8 y 2 puntos respectivamente. Lo que indica claramente que existen dificultades para adaptarse a los lenguajes de programación que implican el dominio de POO.

Pregunta 3: Mencione 3 problemas con mayor relevancia que dificulten el aprendizaje de la POO.



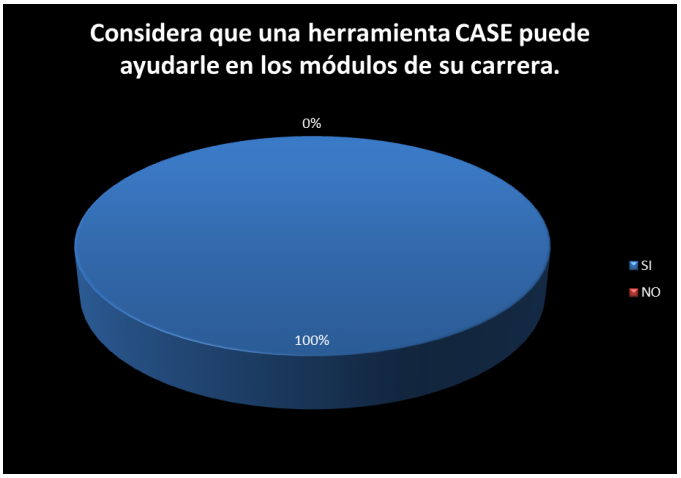
Análisis: Entre los aspectos más destacados en los resultados de esta pregunta son: Docentes Capacitados en herramientas CASE (7 puntos), Más tiempo de clases (6 puntos), Mucha teoría pocos ejemplos (5 puntos) y Pasar de UML a código (5 puntos). Podemos observar que los tres primeros problemas están relacionados directamente en el entorno de aprendizaje dentro del aula. Por lo que con la presente herramienta se busca optimizar el tiempo de clase y enfatizar el desarrollo de ejercicios prácticos.

Pregunta 4: ¿Según su experiencia con software de diseño o codificación, que cualidad debería tener una herramienta CASE para poder apoyar el aprendizaje de la POO?



Análisis: Según la tabulación realizada a esta interrogante se destacaron los siguientes aspectos; Interfaz intuitiva y amigable (11 puntos), Manipular Base de Datos (5 puntos), Incorporar código fuente con autocomplementación (4 puntos) y Uso de Asistentes (3 puntos). Esto permitió desarrollar una herramienta orientada a la comodidad y fácil utilización para el usuario.

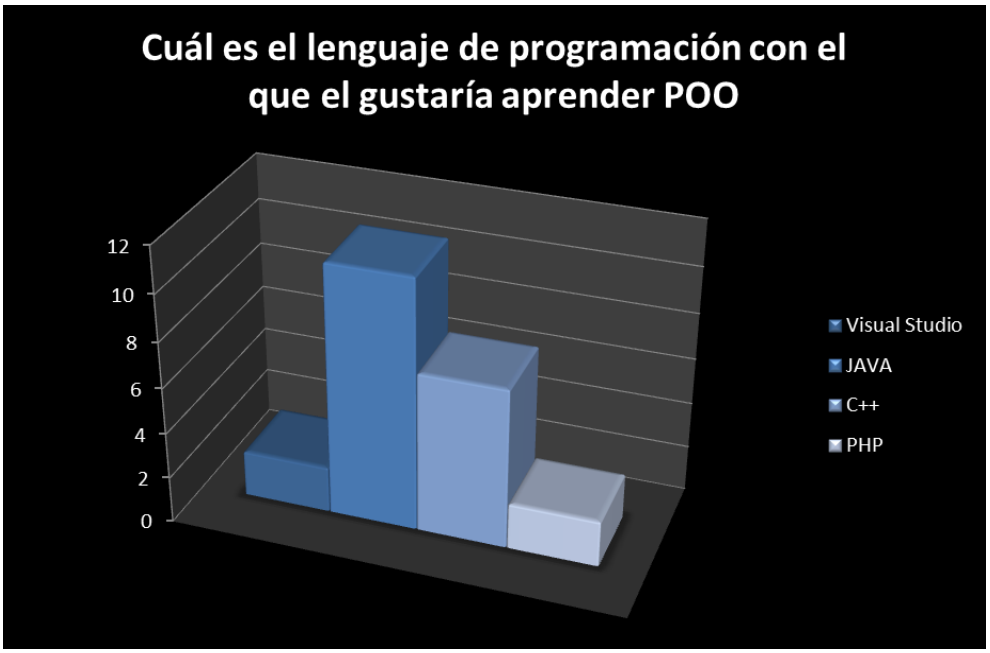
Pregunta 5: ¿Considera que una herramienta CASE puede ayudarle en los módulos de su carrera?



.

Análisis: El 100% de una muestra de 20 profesionales, reiteraron la importancia de contar con herramientas CASE que pueda aplicarse en los contenidos desarrollados en los diversos módulos técnicos.

Pregunta 6: ¿Cuál es el lenguaje de programación con el que le gustaría aprender POO?



Análisis: Los resultados obtenidos permitieron conocer los lenguajes destacados y apropiados para favorecer el aprendizaje de POO, los cuales son: Java con 11 puntos de 20 y C++ con 7 puntos de 20. Lo que permitió tomar en cuenta algunos elementos soportados por java como lo son las Interfaces y crear además estas estructuras bajo la misma nomenclatura.

A través del estudio realizado, se obtuvieron los parámetros requeridos para dar continuidad al proyecto bajo una línea específica en la Programación Orientada a Objetos. Además se puede observar que muchos de los profesionales avalan el desarrollo e implementación de la presente herramienta en el entorno académico.

Identificación de Procesos.

El presente proyecto deberá sustentar una serie de necesidades que le permitan al usuario realizar diagramas con mayor rapidez y facilidad, incorporando además un componente pedagógico que le ayude a entender el desarrollo de un diagrama a través de asistentes visuales. Además debido a la puntual recomendación de los expertos de aplicar la herramienta en estudio, a un ambiente pedagógico; se tomo la decisión de realizar toda una plataforma web en la que los estudiantes puedan realizar sus diagramas de forma fácil y efectiva. Esto conlleva a detallar en la siguiente tabla todos los procesos con los cuales debe contar dicha plataforma.

Actividad	Descripción
Ingresar al Sistema	Tanto el docente como el estudiante deben ingresar al sistema para registrar sus datos y hacer uso del proceso de autorización.
Registro de Alumnos	El registro de alumnos permite que se pueda validar el uso solamente para alumnos de la institución y la materia que actualmente se está cursando y además llevar un registro detallado de sus diagramas.
Realizar Diagrama	El sistema permitirá que el usuario arrastre los componentes necesarios para realizar un diagrama de clases incluyendo sus respectivas relaciones.
Almacenar Diagrama	Se permitirá guardar un diagrama o proyecto, el cual puede ser retomado en caso de no finalizarse y poder además almacenar sus respectivos campos.
Crear diagramas a partir de otros diagramas	El usuario tendrá la posibilidad de crear un diagrama a partir de otro en caso de que un diagrama pueda ser utilizado para modelar un nuevo proceso.
Desplegar los diagramas a un docente	El sistema podrá dar la oportunidad de desplegar los diferentes diagramas según el grupo al que pertenece el alumno
Imprimir diagrama	El sistema además permitirá imprimir los diversos diagramas desde un clic en la opción correspondiente.

Tabla 7: Actividades de un STI.

Análisis del Entorno de trabajo.

Actualmente la carrera de Tec. En Ingeniería de Sistemas Informáticos cuenta con promedio de 200 estudiantes en su primer año y cursan sus materias de socialización en modalidad virtual, lo que permite que pueda brindárseles mayor contenido y flexibilidad de aprendizaje investigativo. Esto implica que toda herramienta que pueda adaptarse a una plataforma web permitirá y garantizará su uso para toda la actual demanda de estudiantes, por lo que el diseño de ITCASE será bajo una arquitectura distribuida aplicando componentes y tecnologías de última generación para garantizar una interacción con todas las características de un software de escritorio pero con la facilidad de ser cargado en equipos de pocos recurso.

Análisis de la Solución propuesta

Para poder plantear una solución que factible y que represente el mínimo de inconvenientes durante su proceso de elaboración, es necesario contar con elementos de peso que permitan aislar cada propuesta según su tecnología, arquitectura e incluso de la plataforma de trabajo a lo largo del proyecto.

Por lo que si observamos y re-inspeccionamos el trabajo a la fecha, observaremos que se cuenta hasta el momento con un estudio que valida una clara herramienta enfocada a un nivel pedagógico y que además debe ser distribuida para su uso en la web; sin embargo existen muchas alternativas validas que pueden implementarse para dar como resultado la herramienta en estudio, es por ello que debemos delimitar 3 soluciones en función de de los requisitos establecidos y documentación existente.

La siguiente tabla mostrará las diferentes alternativas tecnológicas con las cuales se investigo que puede desarrollarse el presente proyecto, mostrará también las ventajas que aportan, las desventajas o los inconvenientes por lo cual puede descartarse cada una de las posibilidades, los software o aplicaciones adicionales que debe utilizarse y su forma de gestionar los datos.

Características	Alternativa 1	Alternativa 2	Alternativa 3
Solución Informática	ITCASE aplicando HTML5, PHP, Jquery, Ajax y Msql	ITCASE aplicando Swing de java y Mysql	ITCASE aplicando JSP, Servlet y applets
Ventajas	- Mayor flexibilidad para integración con CMS - Mucha documentación disponible. -Lenguaje de desarrollo muy utilizado. - Cuenta con todos los elementos requeridos por el proyecto.	- Lenguaje de programación robusto y muy estable. - Sistema de compilación integrado.	- Lenguaje de programación muy robusto y estable. - Funciona en la web.
Desventajas	- Dificultad en el desarrollo de un intérprete para diagramas.	- No funciona en la web - Alto nivel de complejidad - Requiere complementos java para su instalación del lado del cliente	- Dificultad para integrar con CMS - Alto nivel de complejidad - Requiere complementos java para su instalación del lado del cliente

Herramientas / Aplicaciones y Software	- WinXP/ 7 - Ajax, Json y Jquery - Mysql	- WinXP / 7 - Netbeans - JDK - Mysql	- WinXP / 7 - Netbeans - JDK - Mysql
Procesos de datos	Arquitectura Distribuida	Arquitectura de Escritorio	Arquitectura Distribuida

Tabla 8: Alternativas de Solución.

En función de los aspectos planteados en la tabla anterior es necesario ponderar cada alternativa con un peso representativo, sopesando sus ventajas y desventajas sobre la factibilidad técnica, económica y operativa. Además debemos medir si cumple con el tiempo según su nivel de dificultad de desarrollo.

Características	Peso	Alternativa 1	Alternativa 2	Alternativa 3
Factibilidad Técnica	40	Se dice que es factible debido a que actualmente HTML5 y las nuevas tecnologías JavaScript, cuenta con toda una gama de opciones que mejoran la gestión de componentes requeridos para modelar diagramas con mucha facilidad	La alternativa es viable y puede producirse; sin embargo requeriría mucho más tiempo y recursos. Además se cuenta con poca información concreta.	La alternativa es viable y puede desarrollarse pero requiere que el usuario cuente con ciertos requisitos o complementos en su navegador además de agregar que es poca la información de proyectos relacionados.
		Puntuación= 38	Puntuación=18	Puntuación=25
Factibilidad Económica	20	Debido a que toda la tecnología y arquitectura de desarrollo es libre no es necesario cancelar ninguna licencia o licenciamiento de aplicaciones adicionales. Además por la misma arquitectura web se puede cargar con pocos recursos de hardware.	Cuenta con tecnología de libre distribución pero requiere de contar con recursos o computadores con mayor capacidad que la alternativa 1	Cuenta con tecnología de libre distribución pero requiere de contar con recursos o computadores con mayor capacidad que la alternativa 1
		Puntuación=20	Puntuación=18	Puntuación=18
Factibilidad Operativa	20	Por la capacidad de aplicarse en plataformas web como CMS resulta ideal para operar en la actual plataforma	Se considera poco factible debido al inconveniente de operar bajo un entorno	Permite interactuar y desplegarse en red pero sin embargo es difícil de adaptar a la plataforma web del

		del departamento ideal y permite además brindar un entorno agradable y fácil de usar al usuario	poco distribuido.	departamento de virtual.
		Puntuación=20	Puntuación=5	Puntuación=15
Factibilidad Calendario	20	Desarrollo de 3 fases en un promedio de un año	Desarrollo de 3 fases extendidas por 2 años	Desarrollo de 3 fases extendidas por 2 años
		Puntuación=20	Puntuación=15	Puntuación=15
Total	100	98%	56%	73%

Tabla 9: Matriz de posibles soluciones.

Arquitectura de Desarrollo.

Según el análisis obtenido la alternativa de desarrollo más aceptable es la implementación de tecnología basada en JavaScript y HTML 5, por sus actuales opciones avanzadas en el trabajo de gráficos y vectores. Por lo que se implementará una gama de tecnologías basadas en JavaScript para el arrastre de clases y enlaces de relación en un diagrama de forma dinámica; permitiendo así elaborar diagramas que puedan ser ajustados o escalables para el usuario desde un ambiente amigable y muy interactivo.

Además se podrá contar con tecnologías como Ajax y Jquery que nos permiten gestionar la información de registro pertinente los datos generales de los usuarios, brindando formularios de captura de información y una interfaz de usuario atractiva y amigable.

A continuación se presenta una representación de la arquitectura general del proceso de desarrollo:

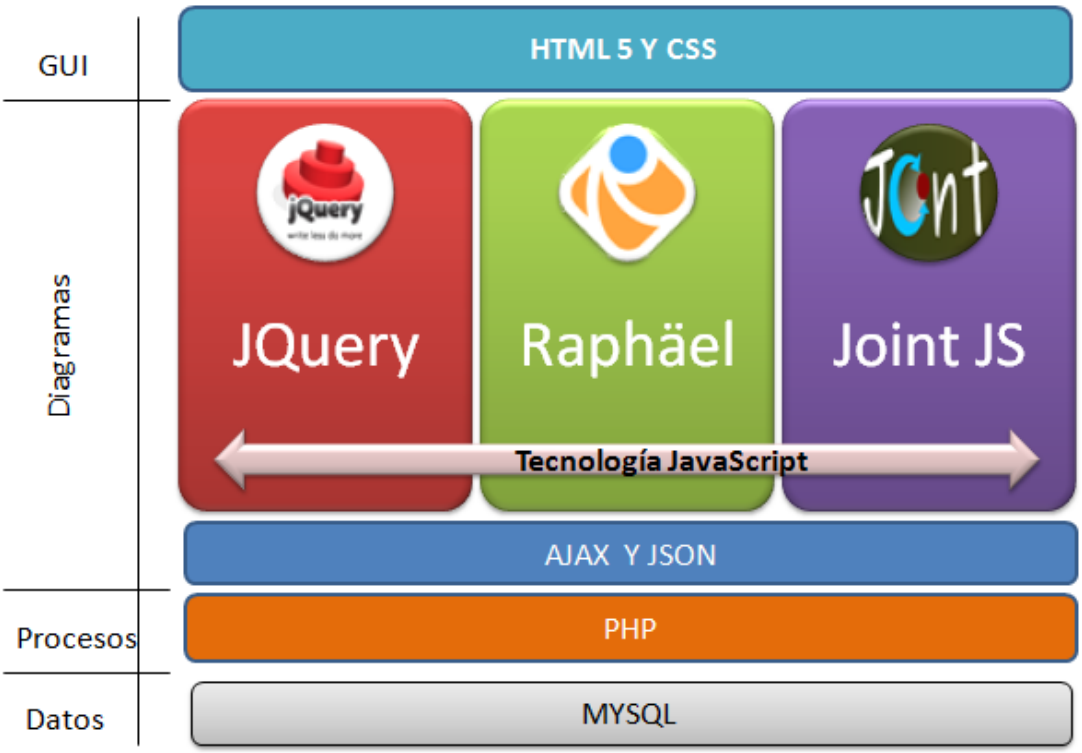


Figura 4: Representación de tecnologías utilizadas.

Proyecto de Investigación

Desarrollo de una herramienta web para el diseño y construcción de software orientado a objetos bajo un enfoque pedagógico

(ITCASE)

ANEXO No. 2

Informe:

Diseño y Desarrollo del Sistema ITCASE

Departamento de Ingeniería en Computación

Objetivo: Desarrollar el modelado requerido de las tareas que beberá ejecutar el Sistema ITCAS, a través de representaciones graficas de UML.

Diagrama de Caso de Uso.

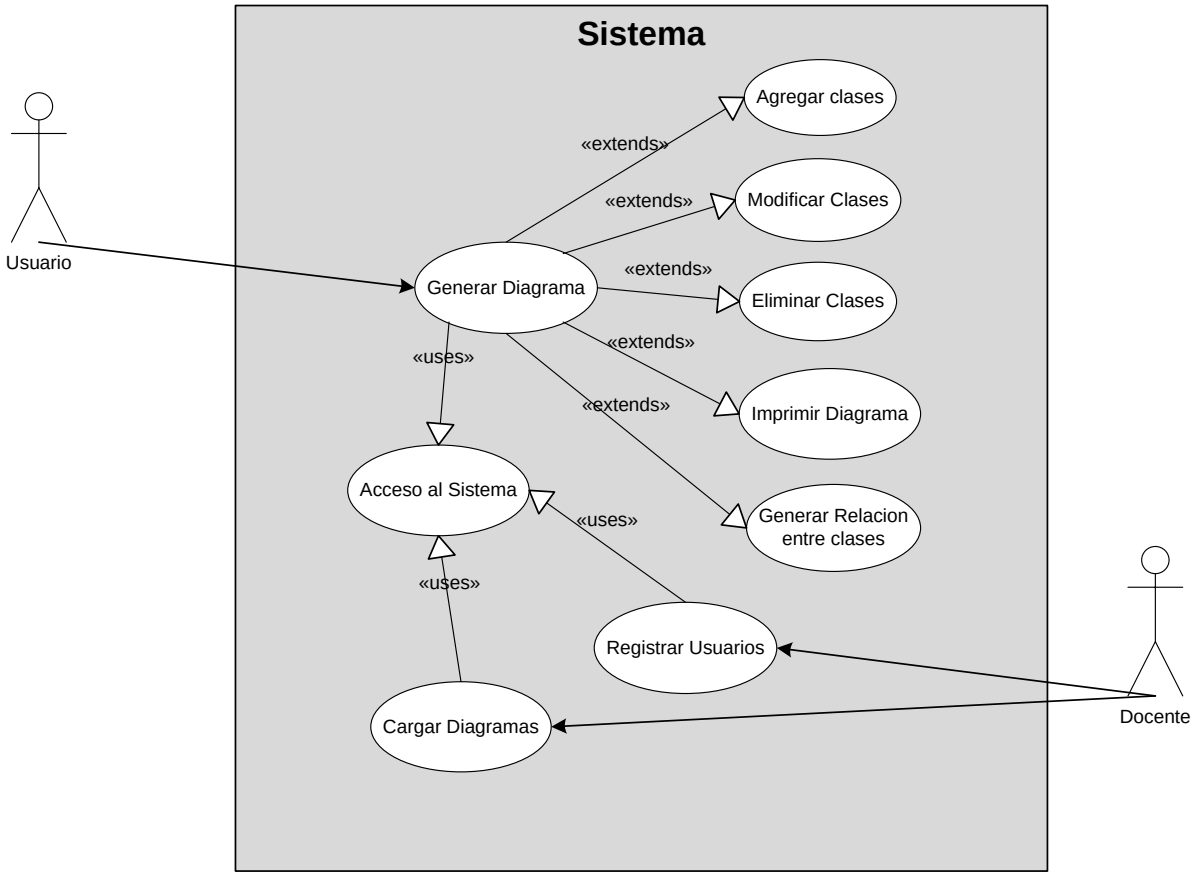


Figura 5: Diagrama de casos de uso de sistema web ITCASE.

Descripción de los Casos de Usos

CU: "Acceso al Sistema".

Precondición.

El caso de uso se inicia cuando el sistema carga la pantalla de acceso cargada a través del link de referencia de la plataforma virtual.

Flujo de Eventos.

Camino Básico.

1. El usuario o docente ingresa el usuario y contraseña
2. El usuario selecciona su tipo de sesión.
3. El sistema busca los datos del usuario y los valida con una base de datos
4. El acceso notifica el acceso exitoso.
5. El Sistema carga el menú principal.

Caminos Alternativos.

1. El usuario ingresa incorrectamente el usuario o contraseña
2. El sistema permite 3 intentos y bloquea el usuario en caso de exceder los mismos
3. El sistema informa el bloqueo y brinda instrucciones de comunicarse con el administrador

Post-Condición.

El caso de uso termina cuando el Sistema carga exitosamente la el menú principal del sistema.

CU: "Generar Diagramas"

Precondición.

El caso de uso se inicia cuando el usuario desee realizar un nuevo diagrama.

Flujo de Eventos.

Camino Básico.

1. El usuario ingresa con una nueva sesión.
2. El usuario agrega una nueva clase o interfaz.
3. El usuario realiza las relaciones pertinentes.
4. El usuario guarda el nuevo diagrama.
5. El usuario abandona la sesión actual.

Camino alternativo.

1. El usuario trabaja sobre un nuevo proyecto.
2. El usuario trabaja sobre un diagrama existente.
3. El usuario genera un nuevo proyecto a partir de uno existente.

Post-Condición.

El caso de uso termina cuando el usuario ha guardado correctamente los cambios realizados a un diagrama.

CU: "Agregar Clases".

Precondición.

El caso de uso se inicia cuando el usuario necesita agregar nuevas clases o interfaces a un diagrama.

Flujo de Eventos.

Camino Básico.

1. El usuario accede a la opción "Clase" y digita el nombre de la nueva clase
2. El usuario establece el numero de atributos que contendrá la clase
3. El usuario digita los nombres, visibilidad y tipo de atributos
4. El usuario selecciona cuantos métodos tendrá la clase.
5. El usuario digita los nombres, visibilidad y tipo de datos que devolverá o no el método.
6. El usuario presiona el botón crear clase.
7. El sistema crea la representación de la clase.
8. El usuario arrastra la clase y la posiciona en el lugar de su preferencia.

Camino Alternativo.

1. El usuario digita un nombre de clase ya existente
2. El sistema informa del error y reenvía al usuario para modificar el nombre.
3. El usuario realiza los cambios pertinentes y procede a crear la clase
4. El sistema dibuja la clase según los requerimientos del usuario.

Post-condición.

El caso de uso termina cuando el usuario ha agregado y ubicado las clases necesarias para realizar su diagrama.

CU: "Modificar Clases".

Precondición.

El caso de uso se inicia cuando existe necesidad de realizar cambios a una clase ya creada.

Flujo de Eventos.

Camino Básico.

1. El usuario selecciona la opción modificar del menú principal.
2. El usuario selecciona la clase a modificar.
3. El usuario agrega los cambios pertinentes.
4. El usuario presiona el botón guardar.
5. El sistema guarda los cambios y lo envía al área de trabajo.

Caminos Alternativos.

1. El usuario decide no realizar los cambios pertinentes y presiona el botón cancelar
2. El sistema redirige al usuario al area de trabajo sin hacer efectivos los cambios.

Post-Condición.

El caso de uso termina cuando el usuario es re-direccionado al área de trabajo con los cambios solicitados.

CU: "Eliminar Clases"

Precondición.

El caso de uso inicia cuando el usuario desea eliminar una clase que ya no pretende utilizar.

Flujo de eventos.

Camino Básico.

1. El usuario selecciona la opción eliminar del menú principal.
2. El usuario selecciona la clase a eliminar.
3. El usuario presiona el botón eliminar
4. El sistema elimina la clase y las relaciones asociadas a ella.
5. El sistema dirige al usuario al área de trabajo.

Camino Alternativo.

1. El usuario decide no eliminar la clase y presiona el botón cancelar
2. El sistema dirige al usuario al área de trabajo sin eliminar la clase.

Post-condición.

El caso de uso termina cuando el usuario observa que la clase a eliminar ya no se encuentra en el área de trabajo.

CU: “Generar Relaciones entre clases”

Precondición.

El caso de uso inicia cuando el usuario decide establecer un tipo de relación a su diagrama de clases.

Flujo de Eventos.

Camino Básico.

1. El usuario ingresa en la opción de relaciones entre clases del el menú principal.
2. El usuario selecciona la clase origen y la clase destino.
3. El usuario selecciona el tipo de relación entre clases e interfaces.
4. El sistema genera la relación especificada y remite al usuario al área de trabajo.

Caminos Alternativos.

1. El usuario decide abortar la relación a establecer entre clases, presionando el botón cancelar.
2. El sistema remite al usuario al área de trabajo sin efectuar cambios.

Post-Condiciones.

El caso de uso termina cuando se observa en el área de trabajo las clases debidamente relacionadas según las especificaciones del usuario.

CU: “Imprimir Diagrama”

Precondición.

El caso de uso se inicia cuando el usuario desea imprimir un diagrama realizado previamente.

Flujo de Eventos.

Camino Básico.

1. El usuario abre o realiza un diagrama.
2. El usuario selecciona la opción de imprimir en el menú inicio.
3. El sistema genera el diagrama en una ventana nueva.
4. El usuario configura los parámetros de impresión e imprime el diagrama.

Caminos Alternativos.

1. El usuario desea cancelar la acción de imprimir y cierra la pestaña nueva.
2. El sistema posiciona al usuario en el área de trabajo.

Post-Condición.

El caso de uso finaliza cuando el usuario imprime exitosamente el diagrama seleccionado.

CU: "Registrar Usuarios"

Precondición.

El caso de uso inicia cuando el docente decide ingresar los datos de un estudiante que tendrá acceso al sistema.

Flujo de Eventos.

Camino Básico.

1. El docente inicia sesión e ingresa al sistema.
2. El docente ingresa los datos del usuario.
3. El sistema gestiona el registro y retorna el mensaje de proceso realizado con éxito.
4. El sistema genera una contraseña que deberá proporcionarse al usuario
5. El sistema direcciona al docente al formulario de registro por si es necesario realizar nuevamente el proceso.
6. El docente proporciona usuario y contraseña del estudiante.
7. El estudiante accede a la plataforma con el usuario y contraseña obtenida, edita la información y guarda los cambios.

Camino Alternativo.

1. El sistema retorna un error en caso de encontrar un problema e inconsistencia con los datos
2. El usuario ingresa los datos correctos en sus respectivos formatos.
3. El sistema registra la información y retorna un mensaje de confirmación.

Post-Condición.

El caso de uso finaliza cuando el usuario obtiene su respectivo ID y contraseña.

CU: "Cargar Diagrama"

Precondición.

El caso de uso inicia cuando el docente necesita revisar los diagramas generados por los alumnos a su tutela.

Flujo de Eventos.

Camino Básico.

1. El docente ingresa al sistema
2. El docente selecciona el grupo de alumnos al cual les revisará los diagramas
3. El sistema Carga todos los proyectos específicos de los alumnos pertenecientes al docente.
4. El docente califica los diagramas y adjudica la nota en la plataforma virtual.

Camino Alternativo.

1. El docente elige cargar todos los proyectos de sus grupos pertinentes.
2. El sistema carga todos los proyectos asociados al docente sin importar el grupo al que pertenezcan

Post-Condición.

El caso de uso finaliza una vez que el docente tiene acceso a todos los proyectos pertinentes a sus respectivos alumnos.

.

Diagrama de Clases.

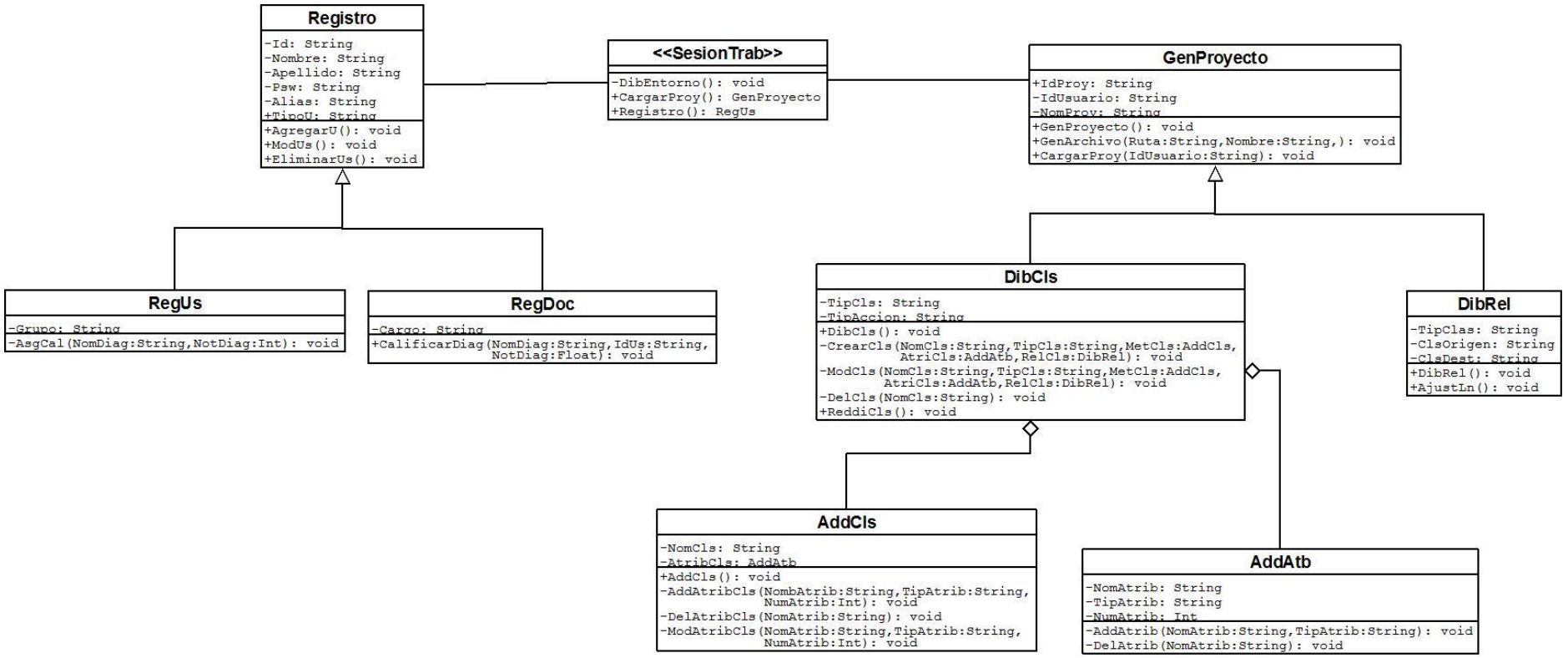


Figura 6: Diagrama de clases de Sistema web ITCASE.

Diagrama de Base de Datos

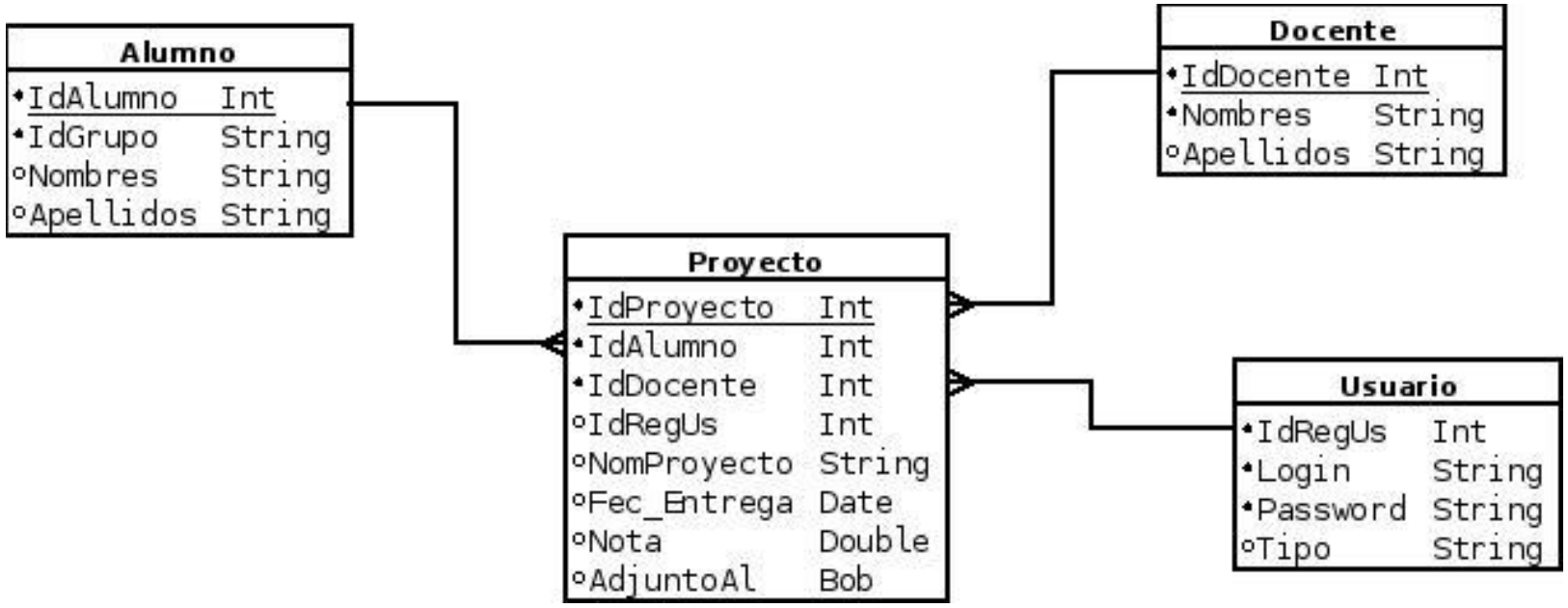


Figura 7: Diagrama de Base de Datos Sistema Web ITCASE.

Codificación.

Para realizar la codificación se utilizó un selecto grupo de tecnologías que permitió realizar una herramienta web interactiva y amigable. Por lo cual es necesario en este apartado describir el papel de cada tecnología y los resultados obtenidos en el software.

Interfaz y estilo de aplicación web.

A través de HTML y CSS se logró desarrollar la maquetación y estilos del sistema obteniendo la apariencia del sitio, brindando los colores y encabezados de la aplicación. Es preciso mencionar que para la captura de algunos datos se utilizó HTML en su versión 5.0


```
#ref{
    float:right;
    color:gray;
}

#dibu{
    background-image:url('../images/line.png');
    background-repeat:no-repeat;
    width:20px;
    height:30px;
}

#borra{
    background-image:url('../images/eraser.png');
    background-repeat:no-repeat;
    width:20px;
    height:30px;
```

Imagen 1. Código CSS y en maquetado HTML

Acceso a componentes e iconos de la aplicación web

JavaScript se utiliza como núcleo base del sistema web ya que permite integrar diversos elementos generados a través del DOM, que en este caso en particular permiten representar gráficos y líneas necesarias para la representación de diagramas.
A continuación se detallan los Frameworks y su aplicación en el desarrollo de la herramienta web en estudio:

JQuery

Implementa tecnología Ajax para crear efectos atractivos al momento de desplegar la interfaz de usuario y el menú de opciones que contiene ITCASE. Ajax también fue utilizado para el recolectar información de los diversos formularios de forma asíncrona optimizando el proceso de validación y carga de datos en MySql.

Imagen 2. Código Jquery para generar menú de opciones

Raphäel

Esta librería permite trabajar con mayor comodidad en vectores gráficos en la web, utiliza SVG W3C como base para crear objetos DOM lo que permite mayor flexibilidad para poder adaptar los objetos gráficos según la medida y forma que el usuario necesite. Esto permitió realizar un espacio de trabajo en el que el usuario pueda insertar, mover o redimensionar los elementos que representan una clase.

Para el dibujo de las clases y sus relaciones se utilizó además Joint JS, el cual es una librería basada en Raphäel.

Imagen 3. Objetos con Raphäel y Joint Js

JSON

Permite completar la tarea de AJAX del intercambio de la información con una base de datos ya que brinda una alternativa notable como sustitución de XLM, permitiendo optimizar el código fuente para los procesos de estructuración y envió de datos a una base de datos, a través de PHP.

```
var atri = JSON.encode(metodos);

var met = JSON.encode(atributos);

qwea =0;
zxca =0;

while(qwea < config.length)
{
    if(config[qwea][0] != "")
    {
        gon[gon.length]=new Array(config[qwea][0],confi
    }
    qwea++;
}
//#Region "Jacobo"
    var configuracion = JSON.encode(gon);
//#EndRegion XD
var stile = JSON.encode(estilo);

var fond = JSON.encode(fondo);
```

Imagen 8. “Código JSON para establecer comunicación amigable con PHP”

Proyecto de Investigación

Desarrollo de una herramienta web para el diseño y construcción de software orientado a objetos bajo un enfoque pedagógico

(ITCASE)

ANEXO No. 3

Informe:

Instrumento de recolección de datos para empresas.

Departamento de Ingeniería en Computación

DEPARTAMENTO DE INVESTIGACION Y DESARROLLO

ESCUELA DE COMPUTACIÓN

INSTRUMENTO DE RECOLECCIÓN DE DATOS

25 de Abril del 2012

Objetivo: Medir el impacto en la calidad, eficiencia y productividad del software desarrollado a través de CASE

1. **¿Ha implementado Herramientas Case para el desarrollo de Software?**

   Si _____ **No** _____

   Si su respuesta es no, favor pasar al literal 11

2. **¿Con que frecuencia utiliza herramientas Case para el desarrollo de Software?**

   Poca **Mucha** **Siempre**

3. **¿Mencione las Herramientas Case más utilizadas en su empresa?**

   a)_____

   b)_____

   c)_____

   d)_____

4. **¿Considera que el tiempo de producción de Software es menor utilizando Herramientas Case?**

   Si_____ **No** _____

5. **¿Considera funcional un Software producido con Herramientas Case?**

   Si_____ **No** _____

6. **¿Mencione las ventajas más relevantes al aplicar Herramientas Case?**

   a)_____

   b)_____

   c)_____

   d)_____

7. **¿Las Herramientas Case implementadas actualmente en su empresa, ¿Son software de pago?**

Si_____ **No** _____

8. **En su entorno de trabajo, ¿Para qué fines utiliza las Herramientas Case?**
   ( ) Diseño de la Base de Datos
   ( ) Modelo Vista Controlador
   ( ) Interfaces graficas
   ( ) Clases
9. **Califique la curva de aprendizaje de las Herramientas Case utilizadas en su empresa**

   1 2 3 4 5 6 7 8 9 10
10. **¿Las Herramientas Case que usted utiliza permiten la integración de Framework de desarrollo?**

    Si_____ **No** _____

Objetivo: Medir parámetros de diseño y desarrollo de software

11. **Mencione los procesos principales que aplica en el desarrollo de Software.**
    a. _____
    b. _____
    c. _____
    d. _____
    e. _____
    f. _____
    g. _____
12. **Mencione 3 problemas más frecuentes presentados en el desarrollo de software de forma colaborativa.**
    a. _____
    b. _____
    c. _____
13. **¿Utiliza estándares a la hora de definir clases, métodos, variables y otros. En el desarrollo de Software?**

    Si_____ **No** _____

    ¿Por qué?

    _____

_____

14. ¿Aplica programación Orientada a Objetos para el desarrollo de software?

Si_____ **No** _____

15. ¿Aplicaría en la producción de software, una herramienta que le permita generar código y diagramas de diseño automáticamente?

Si_____ **No** _____

16. Si su respuesta es afirmativa, que elementos de interfaces consideraría necesarios integrar para realizar de manera eficiente las funciones más relevantes en la producción de software.

a. _____
b. _____
c. _____
d. _____
e. _____
f. _____
g. _____

17. ¿Cuál es el lenguaje de programación que utiliza con mayor frecuencia?

Visual Studio **Java** **PHP** **otros.**

Especifique: _____

18. ¿Qué metodología aplica para el desarrollo de software?

_____

19. ¿Aplica UML para el diseño de Software?

Si_____ **No** _____

20. ¿Utiliza algún IDE para el desarrollo de software?

Si_____ **No** _____

Especifique: _____

Proyecto de Investigación

Desarrollo de una herramienta web para el diseño y construcción de software orientado a objetos bajo un enfoque pedagógico

(ITCASE)

ANEXO No. 4

Informe:

Instrumento de recolección de datos para estudiantes.

Departamento de Ingeniería en Computación

DEPARTAMENTO DE INVESTIGACION Y DESARROLLO

ESCUELA DE COMPUTACIÓN

INSTRUMENTO DE RECOLECCIÓN DE DATOS

10 JUNIO DEL 2012

Objetivo: Medir aceptación y confianza en Herramientas Case

1. **¿Labora actualmente como docente en ITCA-FADE?**

   Si _____ **No** _____

2. **¿Considera que las herramientas de software como: PSeInt, DIA o Visio, ayudan a disminuir la curva de aprendizaje en los módulos de programación?**

   Si _____ **No** _____

   Porque: _____

3. **¿Ha utilizado alguna herramienta de las mencionadas en el literal 2 para la enseñanza o aprendizaje de algún modulo relacionado con la programación de software?**

   Si _____ **No** _____

   Si su respuesta es no, favor pasar al numeral 5

4. **Según el numeral 3 califique su experiencia con todas las herramientas que ha utilizado. (donde 1 el mínimo y 10 es el máximo)**

   1 2 3 4 5 6 7 8 9 10

5. **¿Ha recibido o impartido módulos en los cuales es necesario aplicar programación orientada a objetos?**

   Si _____ **No** _____

6. **¿Utilizaría en los módulos que ha recibido o impartido una herramienta de software que le permita relacionar diagramas UML con su equivalente en código fuente?**

   Si _____ **No** _____

   Porque: _____

Objetivo: Identificar que aportes deben incluirse en una herramientas case para la comunidad estudiantil

7. **¿Con que frecuencia utiliza software para el diseño o codificación de sistemas en los módulos asociados a su carrera?**

   ( )Nunca

   ( )Poco

   ( )Mucho

8. **Señale el nivel de dificultad que representa la curva de aprendizaje de los lenguajes de Programación Orientado a Objetos. (donde 1 el mínimo y 10 es el máximo)**

   1 2 3 4 5 6 7 8 9 10

9. **Mencione los 3 problemas más relevantes que dificultan el aprendizaje de la Programación Orientada a Objetos.**

   a. _____
   b. _____
   c. _____

10. **¿Según su experiencia con software de diseño o codificación, que cualidades debería tener una herramienta case para poder apoyar el aprendizaje de la Programación Orientada a Objetos?**

    a. _____
    b. _____
    c. _____

11. **¿Considera que una Herramienta Case puede ayudarle en el diversos módulos de su carrera?**

    Si _____ **No** _____

12. **¿Cuál es el lenguaje de programación con el que le gustaría aprender Programación Orientada a Objetos?**

    Visual Studio Java C++ otros.

Especifique: _____

UN FUTURO LLENO DE OPORTUNIDADES

Escuela Especializada en Ingeniería

ITCA FEPADE

SANTA TECLA · ZACATECOLUCA · SAN MIGUEL · SANTA ANA · LA UNIÓN

www.itca.edu.sv

Sede Central Santa Tecla
Km. 11 Carretera a Santa Tecla.
Tel. (503) 2132-7400
Fax. (503) 2132-7599

MEGATEC La Unión
C. Santa María, Col. Belén, atrás del
Instituto Nacional de La Unión.
Tel. (503) 2668-4700

MEGATEC Zacatecoluca
Km. 64 1/2, desvío Hacienda El Nilo,
sobre autopista a Zacatecoluca y
Usulután. Tel. (503) 2334-0763, (503)
2334-0768 Fax. (503) 2334-0462

Centro Regional San Miguel
Km. 140, Carretera a Santa Rosa de Lima.
Tel. (503) 2669-2292, (503) 2669-2299
Fax. (503) 2669-0961

Centro Regional Santa Ana
Final 10a. Av. Sur, Finca Procavia
Tel. (503) 2440-4348, (503) 2440-2007
Tel. Fax. (503) 2440-3183