GENERADOR DE APLICACIONES WEB JSP UTILIZANDO NETBEANS, ARQUITECTURA N-TIERS Y El PATRÓN ARQUITECTÓNICO MVC APARTIR DE LOS METADATOS OBTENIDOS DE UNA BASE DE DATOS MYSQL

BELTRÁN RODRÍGUEZ YESICA DANIELA

VARGAS MARTINEZ JEISON ALVEIRO

UNIVERSIDAD DE SAN BUENAVENTURA

FACULTAD DE INGENIERÍA

PROGRAMA INGENIERÍA DE SISTEMAS

BOGOTÁ, D.C.

2012

1. **INTRODUCCIÓN**

En los últimos tiempos el gran crecimiento de Internet ha hecho que las aplicaciones web sean hoy en día uno de los entornos más dinámicos y complejos utilizados en el mundo informático. Sin embargo el principal problema de las aplicaciones web es la vulnerabilidad y mantenibilidad que se brinda al usuario final.

----

De igual forma una aplicación web debidamente estructurado ayudaría a solucionar problemas de seguridad y mantener el ciclo de vida de un de aplicativo web.

--

Con la utilización del patrón arquitectónico MVC se puede mantener estructurado el código de las aplicaciones, separando las diferentes funcionalidades agrupadas en varias capas como lo son: el modelo, la vista y el controlador. Así cada una de estas capas comunicadas entre sí facilita la modificación del código y detección de errores.

Adicionalmente el uso del patrón arquitectónico MVC y la arquitectura N-Tiers ayudan a estructurar una aplicación web y a través de esta estructura es posible realizar modificaciones del código específico en lugar de volver a trabajar con toda la aplicación.

Adicionalmente la arquitectura N-tiers permite generar código más estructurado como también poder modificar al patrón arquitectónico MVC en lugar de volver a trabajar con toda la aplicación.

Velocity pendiente …

La aplicacion generada contiene la conexión a la base de datos

No estamos compitiendo con otros generadores grandes del mercado

No necesitamos de ninguna herramienta para generar código, generamos código apartir de una herramienta como lo es velocity

Como resultado el desarrollador solo tendrá que escoger una base de datos y se generará una aplicación que implemente las operaciones CRUD, la cual puede servir como punto de partida para implementar requerimientos funcionales de la aplicación que el desarrollador desee adicionar

1. **PLANTEAMIENTO DEL PROBLEMA**

En los últimos tiempos el gran crecimiento de Internet ha hecho que las aplicaciones web sean hoy en día uno de los entornos más dinámicos y complejos utilizados en el mundo informático. Sin embargo el principal problema de las aplicaciones web es la vulnerabilidad y mantenibilidad que se brinda al usuario final.

Así, surge la necesidad de implementar una herramienta que permita a los desarrolladores optimizar el tiempo de desarrollo que utilizan para realizar un aplicativo Web con la estructura adecuada, además que combata la vulnerabilidad y proveer una aplicación mantenible.

Puesto que no usa una adecuada estructura para las aplicaciones Web la información es la realmente afectada.

De lo anterior, actualmente existen algunos generadores y Frameworks que intentan combatir dicho problema pero estos no implementan el patrón arquitectónico MVC ni la arquitectura N-Tiers juntas basado en el lenguaje de programación JAVA, tampoco que la aplicación generada sea de fácil modificación y entendimiento generando al usuario final la documentación total del código respectivo, tampoco generan código a partir de los metadatos obtenidos de una base de datos MySQL, algunos de estos generadores no son Open Source, no generan automáticamente un aplicativo web que contenga como mínimo operaciones CRUD, y además que genere las clases usando un motor de plantillas basado en Java como lo es Velocity para un mejor diseño del generador.

En los antecedentes se mostrara algunos de los generadores y frameworks existentes en el mercado con alguna

s características.

* 1. **ANTESCEDENTES**

A continuación se citarán distintos software y frameworks existentes en el mercado que trabajan con el patrón MVC y la arquitectura n-tiers (n-capas).

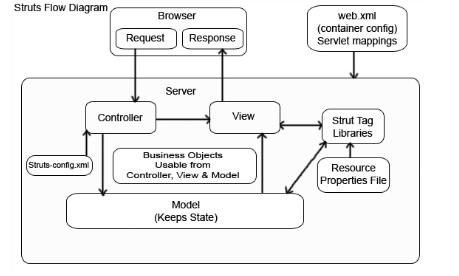
* + 1. **STRUTS (frameworks)**

En el año 2000se desarrolló un proyecto llamado Framework Struts, el cuál se inició en Mayo por Craig R. McClanahan y en el año 2002 pasa a pertenecer a la Apache Software Foundation.

Su principal objetivo fue*“[…] facilitar las tareas propias del ciclo de vida del software, incluyendo la construcción, desarrollo y mantenimiento de la solución. Es fácilmente extensible, ya que todos los elementos están basados en interfaces y clases bases.*”[[1]](#footnote-1)

Struts un software libre en el mercado (open source), se basa en el patrón de arquitectura MVC, donde la capa del controlador automáticamente se encuentra implementada en este software, a pesar de esto el usuario tiene la facilidad de modificarla a su gusto .

Así mismo una de las ventajas de utilizar este tipo de software es que la generación de la interfaz busca evitar el uso de código Java como <% %> es decir malas practicas, para generar una mayor mantenibilidad del mismo como lo describe la siguiente imagen.



**Figura 1. Funcionamiento de Struts[[2]](#footnote-2)**

* + - 1. **Ventajas de Struts**
* Facilita el mantenimiento de la aplicación.
* Es Open Source
* Utiliza MVC y separa en capas los componentes
* El controlador ya se encuentra implementado
* Soporta múltiples interfaces de usuario (Html, sHtml, Wml, Desktop applications, etc.).
  + - 1. **Inconvenientes de Struts**
* Es necesario tener conocimiento profundo de Servlets y JSp´s
* Está dirigida a ser implementada solo en HTML, JPS´s y Servlets
* Con Struts es muy difícil usar algo distinto a MVC.
* Requiere un contenedor de Servlets que soporte JSP 1.1 y Servlet 2.2
  + 1. **SPRING (frameworks)**

SpringSource desarrolló un framework llamado Spring , el cuál se inició en octubre de 2002 por Rod Jonhson, fue lanzado en junio de 2003 bajo Apache 2.0 hasta 13 de diciembre 2011, su principal objetivo fue “*utilizar cualquier objeto como comando o como formulario sin que sea necesario implementar un interface específico del frameworks o una clase base. El enlace de datos en Spring es muy flexible: por ejemplo, trata errores de tipos como errores de validación que pueden ser evaluados por la aplicación, no como errores de sistema. Esto significa que no es necesario duplicar las propiedades de los objetos de negocio, tan simple como cadenas sin tipo en los objetos de formulario son capaces de manejar entradas inválidas o convertir las cadenas adecuadamente*.” [[3]](#footnote-3)

A continuación se presentan algunas de las características y una pequeña reseña de SPRING. Contiene distintas características que se encuentran organizadas en módulos así:

* Core: Encargado de dar la funcionalidad del contenedor de Beans (Se usan para encapsular varios objetos en un único objeto)
* JEE: Integración con aplicaciones J2EE
* DAO: Módulo que elimina el código redundante
* ORM: Módulo que proporciona mapeo objeto-relacional
* Web: Módulo que proporciona características orientadas a web como sevlets, carga de archivos.
  + 1. **CODESMITH (software)**

CEO Eric J. Smith, creó el generador de código CODESMITH donde el objetivo principal de esta herramienta es: “[…] *construir las mejores herramientas para desarrolladores disponibles para ayudar a los desarrolladores de todo el mundo realizar su trabajo más rápido y ayudar a evitar el trabajo repetitivo*.”[[4]](#footnote-4).

Esta herramienta permite el uso de plantillas, así como también permite personalizarlas.

El precio aproximado de este frameworks va desde 300 dólares hasta los 5000 dólares.

* + 1. **GENEXUS (software)**

La compañía uruguaya ARTech creó en el año 1998 “Genexus” – “*una herramienta para el desarrollo de aplicaciones, cuyo objetico es el de ayudar a los analistas de sistemas a implementar aplicaciones en el menor tiempo y con la mejor calidad posible, también genera código para múltiples lenguajes como: Cobol, RPG, Visual Basic, entre otros*”[[5]](#footnote-5)

Quitar

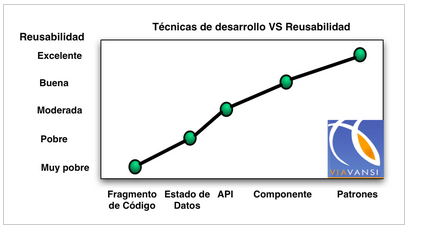
Cabe resaltar que no es necesario volver a inventar la rueda sino aplicar lo que ya existe de manera adecuada, así mismo ocurre con el patrón arquitectónico MVC y la arquitectura N-Tiers, donde ya hemos visto que distintos software en el mercado lo aplican y garantizan una mayor mantenibilidad y sostenibilidad en cualquier aplicativo.

…..

1. **DESCRIPCIÓN Y FORMULACIÓN DEL PROBLEMA**

Este proyecto presenta una propuesta de solución tecnológica que busca satisfacer una problemática real, la cual se presenta al momento de crear aplicativos web donde los desarrolladores realizan distintos procesos en los cuales intervienen una serie de factores, estos son: crear las clases necesarias para el aplicativo, generar la interfaz gráfica, tener conexión a las bases de datos, tener conexión al servidor correspondiente, entre otros, es por esto que es necesario contar con una herramienta que implemente arquitecturas N-tiersy patrón arquitectónico de software MVC,que ayuden a reducir el tiempo y eltrabajo del desarrollador, pero que así mismo permita mayor mantenibilidad y sostenibilidad del aplicativo utilizando código estructurado para una fácil de modificar.

Según Viavansi (Empresa especialista en soluciones de las TIC en el mundo)“No es necesario inventar un patrón de diseño, es mejor implementar estas soluciones que ya han funcionado en el pasado, en la arquitectura del software existen tres pilares que garantizan la calidad del software como lo son: la abstracción, la reutilización, y el uso de patrones como se ve en la siguiente gráfica:”[[6]](#footnote-6)



**Figura 2. Arquitectura de Software**

El anterior planteamiento conduce a formular la siguiente pregunta de investigación

¿Cómo hacer posible el desarrollo de un software que genere aplicaciones web quitargenerador de aplicaciones ..JSP utilizando Netbeans, la arquitectura N-TIERS y el patrón arquitectónico MVC a partir de los metadatos obtenidos de una base de datos MySQL?

1. **JUSTIFICACIÓN**

Actualmente los desarrolladores de software al momento de crear aplicativos web toman bastante tiempo para el desarrollo de aplicaciones, a pesar de que las soluciones ya están en el mercado estas no se aplican, es por esto que el propósito de este proyecto es reducir tiempo y trabajo en el desarrollo, aplicando patrones de software para así obtener resultados de buena calidad con una estructura adecuada utilizando arquitecturas como N-TIERS y patrón arquitectónico MVC

Es importante resaltar que no existen en el mercado aplicativos que utilicen la arquitectura N-TIERS y el patrón arquitectónico MVC simultáneamente. Es así que este proyecto se realizará mediante la herramienta de NETBEANS, ya que este permite el uso de las dos arquitecturas mencionadas.

Con el fin de generar un aplicativo web que permita crear automáticamente JSP´s utilizando el patrón arquitectónico MVC y la arquitectura N-TIERS se podrá brindar al desarrollador facilidad de uso, y de la misma manera ofrecer un mejor rendimiento en cuanto a tiempo de ejecución y optimización de recursos.

1. **OBJETIVOS DE LA INVESTIGACIÓN**
   1. **Objetivo General**

Desarrollar un generador de aplicaciones web JSP utilizando NETBEANS, la arquitectura N-TIERS y el patrón arquitectónico MVC a partir de los metadatos obtenidos de una base de datos MYSQL.

* 1. **Objetivos Específicos**
* Diseñar el modelo estático de clases para representar el diseño lógico especificado a través de los metadatos almacenados en una base de datos MySQL.
* Obtener la Metadata de una base de datos MySQL.
* Diseñar e implementar el generador para las aplicaciones web, basados en la información presente en el modelo estático de clases, las cuales deben ser estructuradas de acuerdo al patrón arquitectónico MVC y la arquitectura n-Tiers y deben soportar las operaciones CRUD sobre las entidades especificadas en los metadatos de la base de datos -> Se carga modelo falta clases generadas
* Generar automáticamente la aplicación web que cumpla con el patrón arquitectónico MVC y la arquitectura n-Tiers, la conexión a la base de datos y los procedimientos CRUD.

1. **ALCANCES Y LIMITACIONES DEL PROYECTO**
   1. **Alcances**

Con este proyecto se pretende llegar hasta el punto de generar una aplicación, la cual cumpla con las características descritas, contando con un aplicativo que genere código Java, bajo el patrón arquitectónico MVC y arquitectura N\_TIERS, en la plataforma Netbeans, la cual funcionará con la estructura extraída los metadatos obtenidos de una base de datos MySQL.

Los elementos conceptuales que va a contener el código generado será:

* N-TIERS
* MVC
* JSP`s
* Conexión a la Base de Datos
* Estilos HTML y CSS
* Jquery

Se pueden incluir más alcances

* 1. **Limitaciones**

Las limitaciones que tiene este proyecto de grado es que se utilizará únicamente una base de datos MYSQL.

El generador estará desarrollado bajo la plataforma NETBEANS y como mínimo se generarán solamente procedimientos básicos del CRUD.

**Se puede incluir limitaciones de base de datos ….????**

1. **MARCO DE REFERENCIA**
   1. **MARCO TEÓRICO** 
      1. **Arquitectura De Software**

Se utiliza para organizar las diferentes partes de una aplicación, la definición de una arquitectura de software es:

Ayudar a decidir cómo dividir la aplicación web en capas.

Proporciona una pauta para definir la forma en que todos los componentes trabajen juntos para llevar a cabo la funcionalidad que se pretende conseguir con la aplicación.

* + 1. **Arquitecturas para la construcción de Aplicaciones Web**
       1. ***Arquitectura centrada en páginas:***

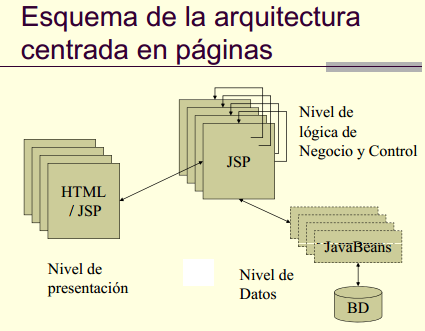
Utiliza páginas con extensión .jsp que gestionan peticiones de manera directa.

Esta basada en páginas JSP y HTML para la interacción con el usuario así es también llamada lógica de presentación**.**

Contiene páginas JSP las cuales son las responsables de controlar el flujo de la aplicación por tal motivo reciben peticiones de las capas relacionadas y direcciona a las siguientes páginas es por esto que a este proceso se le denomina lógica de negocio y control.

Esta arquitectura accede a los datos desde la misma página JSP.

A continuación se presenta un esquema de la arquitectura centrada en páginas.

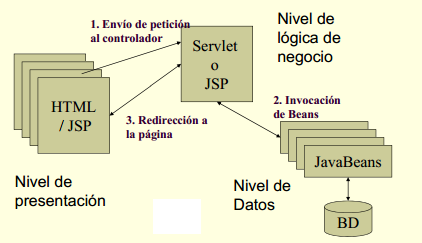


**Figura 3. Esquema de la arquitectura centrada en páginas[[7]](#footnote-7)**

* + - 1. ***Arquitectura de “dispatcher” o MVC***

En el patrón arquitectónico dispatcher o MVC (modelo vista controlador), utiliza un servlet o un JSP el cual actúa de mediador o controlador delegando la resolución de peticiones en páginas JSP.

Los JSP se utilizan para generar el nivel de presentación y los servlets para realizar las tareas que requieren procesamiento y control



**Figura 4. Esquema de la arquitectura MVC “**[[8]](#footnote-8)

Es por esto que para la creación de aplicativos es necesario la implementación de arquitecturas de software, para poder organizar y estructurar código apoyando el trabajo del desarrollador.

También se puede concluir que la arquitectura de software es un grupo de unión de distintos patrones, donde su función principal esguiar al desarrollador al momento de fabricarsoftware, permitiendoa los desarrolladores y analistascubrir los objetivos y metas propuestas para un proyecto determinado.

* + - 1. **MVC (Model View Controller)**

Según Tony Marston “*el Modelo-Vista-Controlador es un patrón arquitectónico* orientado a la arquitectura de software publicado por Buschmann en 1996,estipulaque *divide una aplicación interactiva en tres componentes. Fue desarrollado originalmente para definir los roles tradicionales de entrada, procesamiento y salida hacia el reino GUI (Graphic User Interface), las cuales son aplicaciones que poseen interface grafica Windows.*

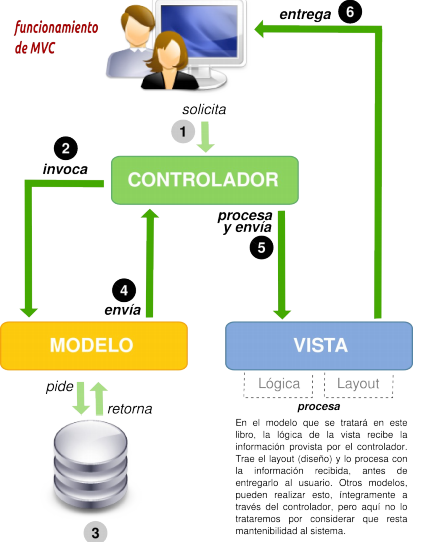
* *El* ***modelo*** *contiene la funcionalidad básica, este componente es independiente de cualquier representación de salida y/o comportamiento de entrada.*
* *La* ***vista*** *se encarga de mostrar información al usuario donde pueden existir múltiples vistas del modelo y donde cada vista tiene asociado un componente controlador.*
* *El* ***controlador*** *se encarga de la entrada del usuario, estas entradas se entienden como pulsaciones de botones, teclado entre otras, estos eventos son traducidos a solicitudes de servicio o también llamados “service requests”.*[[9]](#footnote-9)

De lo anterior se puede deducir que se creó este patrón para definir roles y ayudar a estructurar la aplicación en diferentes capas modelo, vista y controlador, dándole un trabajo específico a cada capa del patrón arquitectónico.

Según Eugenia Bahit (Analista Programadora de la Ciudad de Buenos Aires) expresa que:“*El patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario, a la vez que permite no mezclar lenguajes de programación en el mismo código*

Así mismo define el funcionamiento del patrón MVC así:

* El usuario realiza una petición
* El controlador captura el evento
* Hace la llamada al modelo
* El modelo será el encargado de interactuar con la base de datos
* El controlador recibe la información y la envía a la vista
* La vista, procesa esta información ”[[10]](#footnote-10)

****

**Figura 5. Funcionamiento MVC**

Para que el patrón MVC se puede concluir las siguientes ventajas y desventajas mostradas así:

* + - * 1. ***Ventajas***
* Reutilización del código
* Separación de las capas
* Mantenimiento de los sistemas
* Se puede crear distintas representaciones de los datos obtenidos
  + - * 1. ***Desventajas***
* Tener que señirse por una estructura predefinida
* Su implementación es costosa porque es un diseño orientado a objetos QUITAR
  + - 1. **N-TIERS o N-Capas**

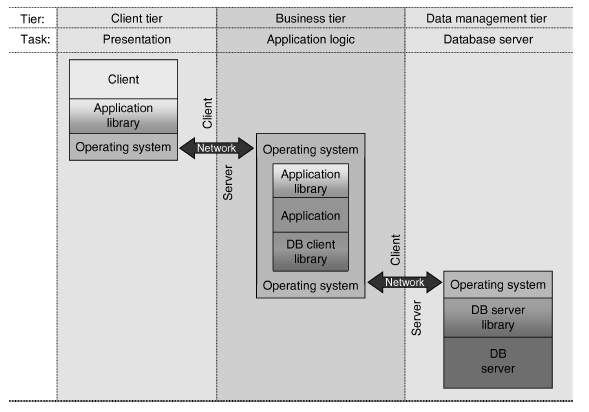
Esta arquitectura es muy usada en distintos proyectos de software, se aplica desde 3 capas hasta n-capa.

Según **Microsoft** la arquitectura de tres capas es:

“*Es el marco fundamental para el modelo de diseño lógico, los segmentos de componentes de una aplicación en tres niveles de servicios. Estos niveles no se corresponden necesariamente con ubicaciones físicas en diferentes ordenadores, sino mas bien a las capas lógicas de la aplicación*”[[11]](#footnote-11)

Según **Heiko Schuldt (***Departamento de ciencias de la computación de la universidad de Basilea, suiza***),** expresa que:

“*Una arquitectura de varios niveles es una arquitectura de software en el que diferentes componentes de software, organizados en niveles (capas), proporcionan una funcionalidad específica. La ocurrencia más común de una arquitectura de varios niveles es una arquitectura de 3 niveles que consta de un nivel de gestión de datos (sobre todo que abarca uno o varios servidores de bases de datos), un nivel de aplicación (lógica de negocios) y un nivel de cliente (funcionalidad de la interfaz)*”[[12]](#footnote-12)

****

**Figura 6. Estructura de una arquitectura 3 capas**

La anterior arquitectura contiene varias capas que se describen en la siguiente tabla:

|  |  |
| --- | --- |
| **CAPA** | **DESCRIPCIÓN** |
| **Datos** | La cual define todos los objetos de la base de datos como tablas, objetos, vistas entre otros, donde los desarrolladores pueden administrar dicha base de datos |
| **Negocio** | Controla el acceso a los servicios desde otras capas |
| **Interfaz** | Muestra los datos obtenidos desde la capa de datos y el procesamiento que se realiza en la capa de negocio para posterior a eso mostrarle al usuario una interfaz con la información obtenida |

**Tabla 1. Capas N-tiers**

Así mismo se tienen algunas ventajas y desventajas de la implementación de esta arquitectura:

* + - * 1. ***Ventajas***
* **Escalabilidad:** Habilidad de adicionar recursos para soportar mayores números de usuarios sin modificar las aplicaciones
* **Extensibilidad:** Habilidad de incrementar la funcionalidad de una aplicación sin alterar lo ya existente
* **Seguridad:** Capacidad de extender y propagar la autenticación en el “front-end” hacia las capas intermedias, salvaguardando la información de los niveles de datos
  + - * 1. ***Desventajas***
* **Complejidad:** En la medida en que existan mayores elementos en hardware y software en momentos de falla la complejidad de detectarla es elevada
* **Comunicaciones:** En la mayoría de los casos, cada capa se encuentra distribuida en la red. Lo cual genera mayor consumo de ancho de banda
* **Costos de Mantenimiento:** En la medida en que cada capa crece, tanto los costos de instalación, actualización y mantenimiento de hardware y software se incrementan.”[[13]](#footnote-13)
  + - 1. **METADATOS**

“Los metadatos, en sí, no suponen algo completamente nuevo dentro del mundo bibliotecario. Según Hoew (1993), el término fue acuñado por Jack Myers en la década de los 60 para describir conjuntos de datos, la definición que se dio fue: Dato sobre el dato, ya que proporcionaba información mínima necesaria para identificar un recurso”[[14]](#footnote-14)

Otra definición es: “algo que va mas allá de los datos, datos sobre datos, en general un objeto que describe o dice algo sobre otro objeto de información, también pueden definirse como: es un dato que se encarga de mantener un registro sobre el significado, contexto o propósito de un objeto informativo, de tal forma para poder describir, entender, extraer y administrar dicho objeto”[[15]](#footnote-15)

Lo anterior puede se resume en la siguiente imagen



**Figura 7. Metadatos**

* + - 1. **NETBEANS**

“NetBeans es un entorno de desarrollo, creado en Junio de 2000 por Sun MicroSystems[[16]](#footnote-16), hecho principalmente para el lenguaje de programación Java, es un producto libre y gratuito sin restricciones de uso.

Es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento.

QUITAR

Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.”

**…**

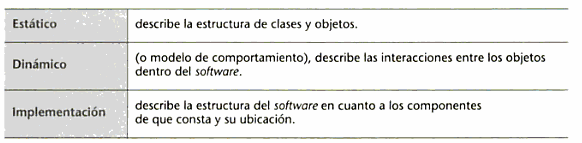
* + - 1. **CRUD**

Por sus siglas en inglés (*create, read, update, delete*), es la implementación de funciones de procesamiento básicas como crear, leer, actualizar y eliminar datos de una base de datos.

En bases de datos crear se realiza con la palabra INSERT, leer con SELECT, actualizar con UPDATE y eliminar con la palabra DELETE

* + - 1. **MODELO ESTÁTICO DE CLASES**

El UML comprende un cierto número de diagramas interrelacionados mediante conceptos comunes así:



**Figura 8. UML[[17]](#footnote-17)**

El modelo estático del UML, es aquél en el que se describen las clases y los objetos, y se denomina estático porque muestra las relaciones posibles a lo largo del tiempo.

Es por esto que es de vital importancia tener el modelo estático de clases, porque con esta información encontrada, es más fácil conocer la base de datos con la que estamos trabajando y entender las relaciones entre tablas.

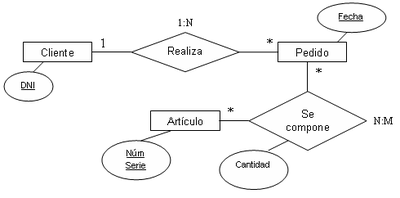
* + - 1. **MODELO ENTIDAD – RELACIÓN**

Se caracteriza por utilizar una serie de símbolos y reglas para representar los datos y sus relaciones, con este modelo se logra representar de manera estructurada y lógica una base de datos, los principales elementos del método entidad-relación son las entidades y sus atributos.[[18]](#footnote-18)

La entidad es un objeto que obtiene información de la base de datos, se representan gráficamente por medio de un rectángulo

La relación es la asociación de dos o más entidades, existen relaciones 1: 1, 1:N y N:M

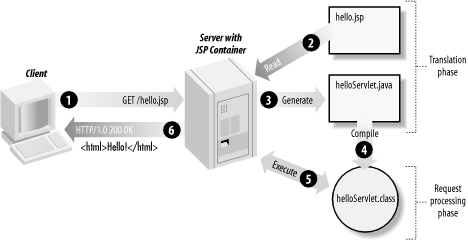
En la siguiente imagen se explicará gráficamente las entidades y relaciones de un problema específico:



**Figura 9. Modelo entidad – relación**

* + - 1. **JSP**

Por sus siglas en inglés (***Java Server Pages***), “*permite a los desarrolladores web y diseñadores para desarrollar rápidamente y mantener con facilidad, ricas en información, las páginas Web dinámicas que aprovechan los sistemas empresariales existentes. Como parte de la familia de tecnologías Java, JSP tecnología permite el desarrollo rápido de aplicaciones basadas en Web que son independientes de la plataforma. La tecnología JSP separa la interfaz de usuario de generación de contenidos, permitiendo a los diseñadores cambiar el diseño general de la página, sin alterar el contenido dinámico subyacente*.”[[19]](#footnote-19).



**Figura 10. JSP[[20]](#footnote-20)**

Es una tecnología Java que permite al desarrollador mostrar gráficamente distintas temáticas, esta interfaz gráfica es la que finalmente hará la interacción con el usuario final y en ella se mostrará todo lo que procese el Servlet como se ve en la anterior imagen.

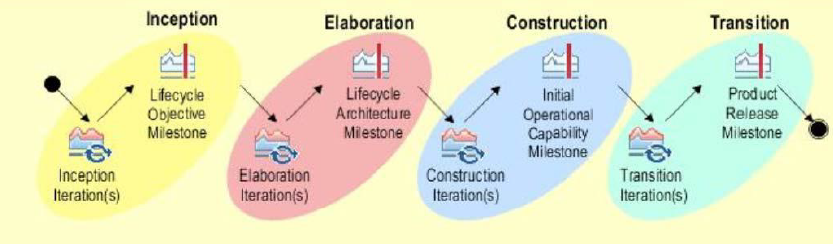
**METODOLOGÍA CICLO DE VIDA DEL PROYECTO - RUP**

1. **CICLO DE VIDA DEL PROYECTO - RUP**

**COLOCAR MAS INFORMACION DEL METODOLOGIA RUP**

El ciclo de vida del proyecto prevé a las partes interesadas la supervisión, transparencia y mecanismos de dirección para controlar la financiación de proyectos, el alcance, la exposición al riesgo, valor, y otros aspectos del proceso.

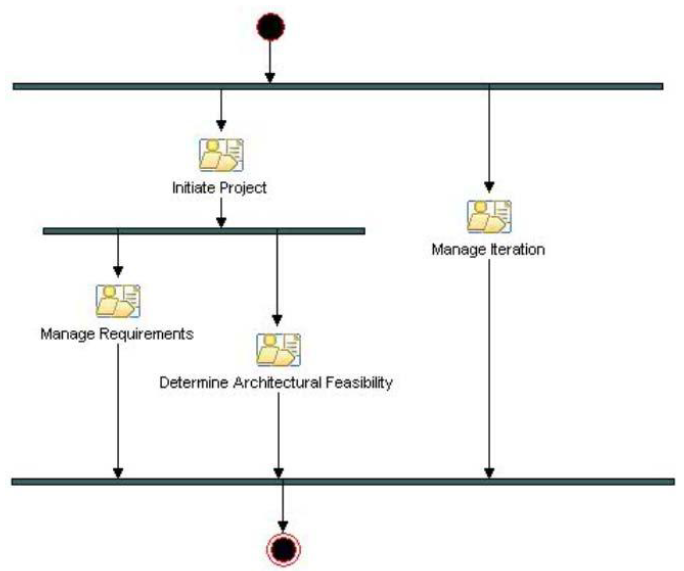
Cada iteración ofrece un producto de incremento, lo que proporciona una oportunidad para que las partes interesadas comprendan el valor de lo que ha sido entregado y realicen un seguimiento continuo del proyecto, también ofrece al equipo de desarrollo la oportunidad de hacer cambios al proyecto para optimizar los resultados.



**Figura 11. Fases del Ciclo de Vida de un Proyecto[[21]](#footnote-21)**

* 1. **FASES** 
     1. **Creación**
* Identificar y entender el problema.
* Identificar la funcionalidad del sistema.
* Determinar al menos una posible solución.
* Elaborar las plantillas iníciales donde se realiza la recolección de información de requerimientos y el diseño de la posible solución.
* Entender el costo, tiempo y riesgos asociados con el proyecto.
* Documentar sobre posibles riesgos relacionados con el proyecto.
* Definir los objetivos del proyecto.

En cuanto a las herramientas y ambiente se debe tener definido el equipo para la primera iteración, un entorno de desarrollo incluyendo herramientas e infraestructura y procesos de desarrollo.

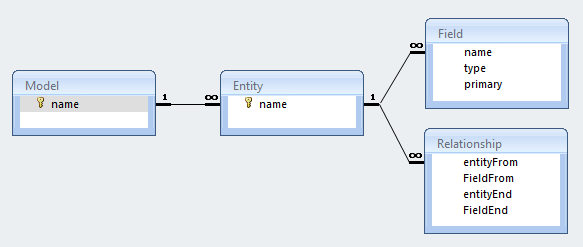


**Figura 12 Fase de Inicialización[[22]](#footnote-22)**

Para la fase de inicialización o creación, a continuación se explican unos procesos que se aplicaron para el desarrollo y cumplimiento del primero objetivo mencionado en este documento, en primer lugar se realiza el diseño del Modelo estático de clases, punto clave del generador, en el cual se almacena la información extraída de la base de datos por la clase Builder, su función es obtener la metadata de la base de datos dada, este modelo lo que hace es mostrar como las clases trabajan entre sí para lograr tener una base consistente para la cual no sea necesaria la conexión de nuevo a la base de datos sino trabajar con objetos obtenidos por el lenguaje de programación.

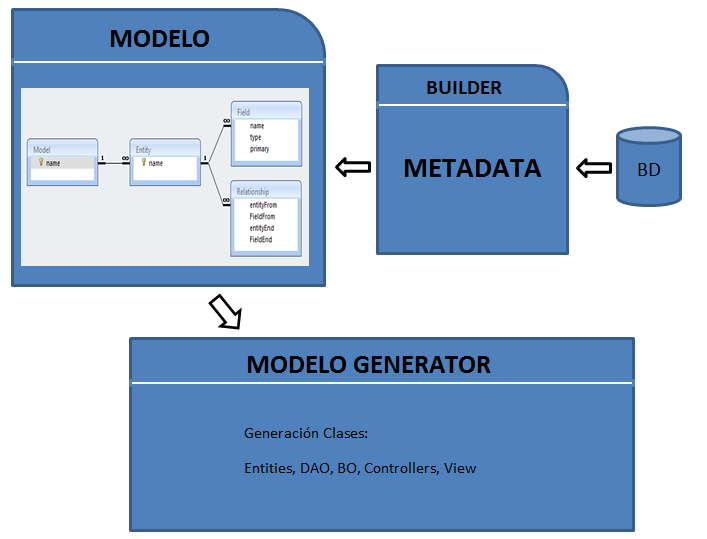
* + - 1. **MODELO ESTÁTICO DE CLASES - PROYECTO**

Para la generación de código y la extracción de los datos almacenados en una base de datos, es necesaria la creación de un modelo adecuado que permita la obtención de los mismos, con el fin de tener una adecuada estructura separada en objetos, como se muestra en la siguiente figura.

****

**Figura 13 Modelo estático de clases - Proyecto**

Cuando se tenga la información de la base de datos en el modelo, el generador dejara de interactuar indirectamente con la base de datos real, con el fin de generar correctamente y estructuradamente las clases a partir del mismo, es decir, el modelo estático de clases es el puente entre el generador y la base de datos.

****

**Figura 14 Relación Modelo – Base de Datos**

Con lo anterior se especifica cómo se conecta la clase modelo con la información extraída de la base de datos, con el fin de no tener que interactuar con la base de datos sino trabajar con la información obtenida en el modelo, y posterior a esto generar las clases necesarias para el aplicativo. Se muestra como trabajan estas clases juntas con el fin de brindar mayor mantenibilidad y funcionalidad para el generador

* + - 1. **OBTENER LA METADATA DE UNA BASE DE DATOS MYSQL.**

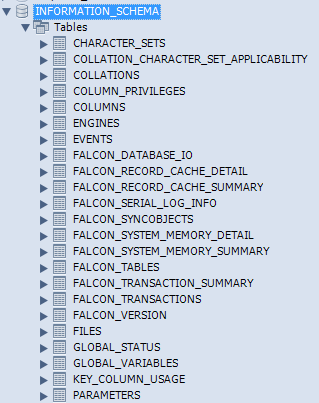
Para la obtención de los metadatos de la base de datos MySQL se realizaron dos clases especificas llamadas Builder y Conexión\_BD, las cuales permiten la conexión a la base de datos, y con el método Database Metadata getMetadata() y los querys se seleccionan en la base de datos la información necesaria para la generación de las clases.

Para la conexión a la base de datos, esta se realiza mediante el driver de MySQL JDBC Driver, por sus siglas en ingles de Java Database Connectivity. JDBC es un conjunto de clases que nos permite acceder a diversos gestores de bases de datos en forma transparente.[[23]](#footnote-23)



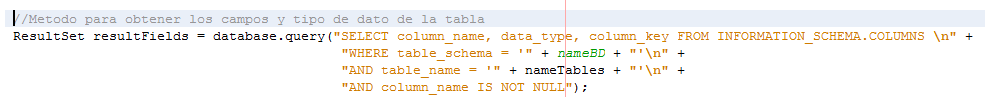
**Figura 15. Conector MySQL JDBC**

Así mismo se realizó distintos querys que permiten traer información más específica de la base de datos, para la obtención de los campos, tipo, llaves foráneas y primarias, se realizaron querys apuntando a INFORMATION\_SCHEMA, es la base de datos de información, que almacena información acerca de todas las otras bases de datos que mantiene el servidor MySQL .[[24]](#footnote-24)



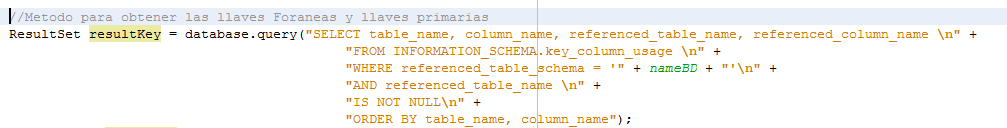
**Figura 16. INFORMATION\_SCHEMA**

Para la obtención de los campos y el tipo de dato de cada uno se realizó el siguiente Query



**Figura 17. Query obtención campos y tipo de dato**

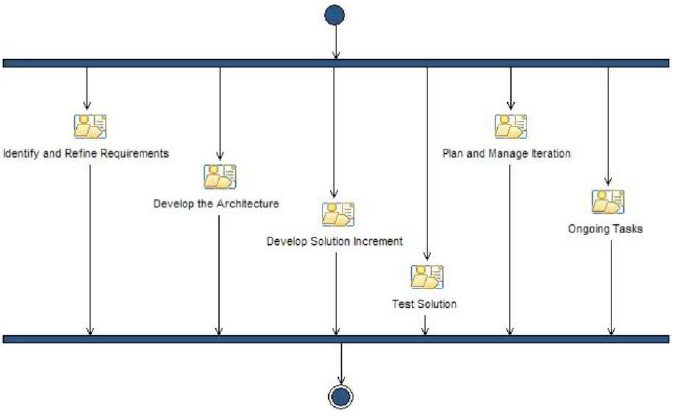
Para la obtención de las llaves primarias y llaves foráneas se realizó el siguiente Query



**Figura 18. Query obtención llaves foráneas y llaves primarias**

Con lo anteriormente descrito es como se obtiene la metadata de la base de datos MySQL, esta información posterior a esto se envía al modelo el cual se explica en la fase de elaboración

* + 1. **Elaboración**
* Especificación y realización de casos de uso detallando cada uno de los requerimientos.
* Diseña mediante diagramas UML (Casos de Uso, Diagrama de actividad,
* Diagramas de secuencia), implementa, valida y realiza la línea base de la arquitectura, documento de arquitectura.
* Mitiga los riesgos esenciales, y produce un programa preciso y estimación de costos.

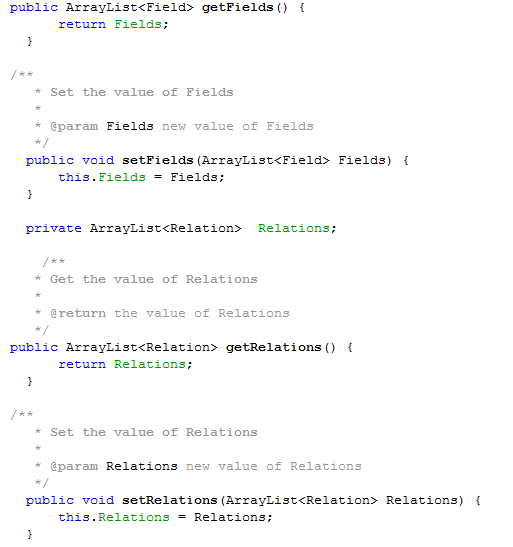


**Figura 19 Fase - Elaboración[[25]](#footnote-25)**

* + - 1. **CARGA DE EL MODELO CON LA INFORMACIÓN OBTENIDA**

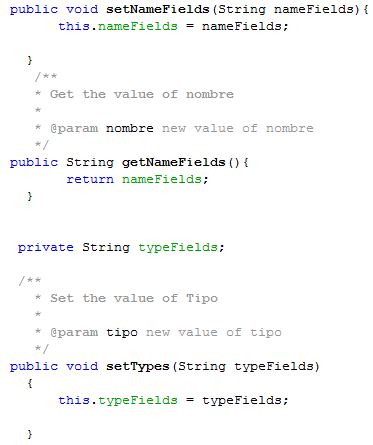
Para la carga del modelo desde el Builder se implementaron 4 clases (Entidad, Campo, Relación y Modelo), cada clase tiene sus Set y Get´s correspondiente que permite el envío de la información de una clase a la otra así:

**CLASE ENTIDAD (ENTITY)**



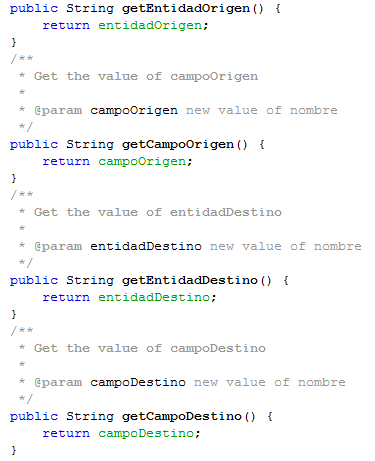
**Figura 20. Clase Entidad**

**CLASE CAMPO (FIELD)**

****

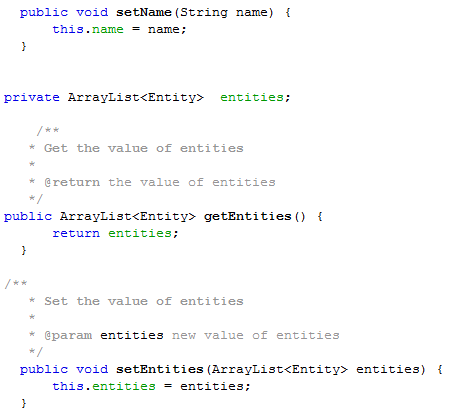
**Figura 21. Clase Campo**

**Clase Relación (RELATION)**

****

**Figura 22. Clase Relación**

**Clase Modelo (MODEL)**

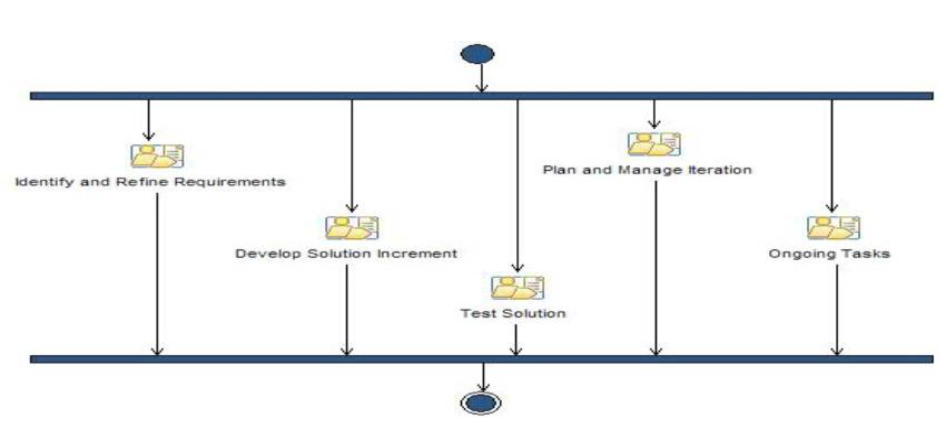
****

**Figura 23. Clase Modelo**

En este modelo es el centro principal del sistema en donde se almacena toda la información extraída y desde la cual se generarán todas las demás clases aplicadas para el patrón MVC y la arquitectura N-TIERS.

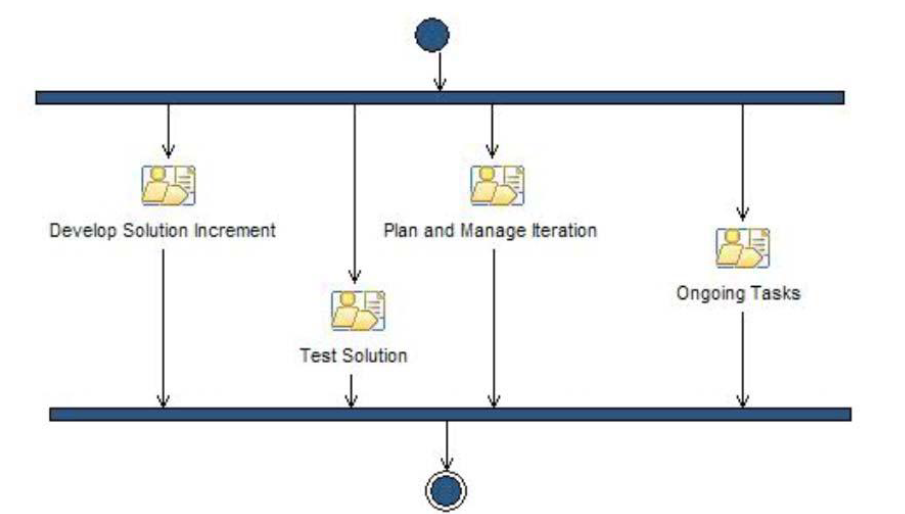
* + 1. **Construcción**

Iterativamente se desarrolla el producto completo, se determina su operatividad y capacidad de uso de acuerdo a la arquitectura definida y se verifica que cumple todas las condiciones para proceder con la entrega final.



**Figura 24. Fase de construcción[[26]](#footnote-26)**

* + 1. **Transición**
* Prueba la versión Beta e implantación de la solución
* Se elaboran los manuales de funcionalidad del proyecto
* Implementación.

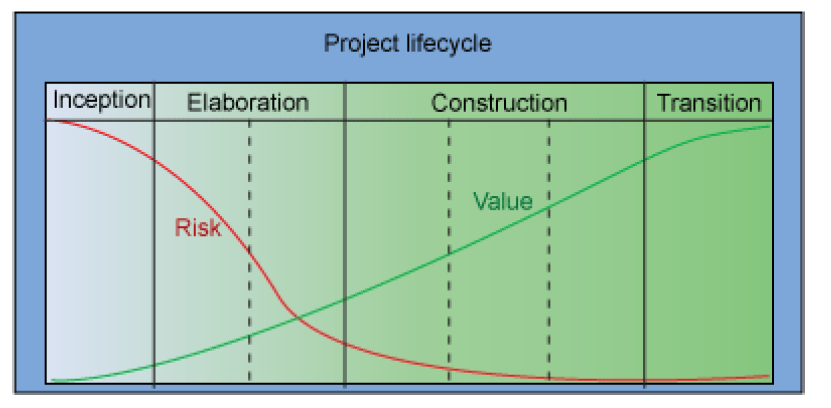


**Figura 25. Fase De Transición**

* + 1. **Riesgos**

En cada elemento de cada una de las fases se deben realizar registros de revisión, para definir si el proyecto continua o no dependiendo del progreso y su evolución. Uno de los objetivos del ciclo de vida del proyecto es centrarse en dos elementos la reducción del riesgo y la creación de valor, en la siguiente figura se puede observar una grafica que ilustra su comportamiento a lo largo del ciclo de vida del proyecto.

Reducción de Riesgo (curva roja) y Creación de Valor (curva verde)



**Figura 26. Riesgo**

El riesgo es una manifestación de la probabilidad que cosas inesperadas sucedan en el desarrollo normal del proyecto, este interfiere en el camino de la creación de valor. Es directamente proporcional a la incertidumbre en las estimaciones, y los interesados quieren saber más temprano que tarde que el proyecto puede entregarse en el plazo estipulado.

1. **MARCO LEGAL Ó NORMATIVO**

“En el mundo ya existen estándares de accesibilidad web. Tal es el caso de la W3C, asociación internacional de carácter neutro fundada en 1994 y cuya finalidad es lograr una web universal, accesible, confiable y mejor.

En el caso de Colombia, por ley nacional, hay que tener sitios web accesibles, sobre todo si se trata de sitios web del estado. La ley existe aunque no está reglamentada aún pero sí lo será próximamente, según explica Martín.”[[27]](#footnote-27)

“El desarrollo que ha tenido Internet en Colombia, lo ubica como uno de los países de América Latina y el Caribe de mayor crecimiento en este sector, según datos del Ministerio de Comunicaciones entre 1997 y 1999, el crecimiento del número de usuarios fue del 220%y la densidad de conexiones (hosts) aumentó a una tasa promedio anual del 231% entre1993 y 1998. El número de proveedores de Internet llega a 220 registrados y a 100operando efectivamente al 2000.

Hasta el año de 1989, donde contaba con un monopolio estatal en el campo de las telecomunicaciones, la Ley 72 de 1989 y el decreto-Ley 1900 de 1990 rompen dicho monopolio y con el Decreto 1794 de 1991, se crea el escenario normativo para que empresas privadas puedan prestar los servicios de valor agregado.

En 1990 se conformo la Red Universitaria de Colombia, RUNCOL, en la cual participaban las principales universidades del país con apoyo del Sistema de Información de la Educación Superior – ICFES, y con una infraestructura para operar con la red BitNet.

Para 1993, Colciencias y el ICFES, deciden participar en la conformación de una Red bajo el protocolo de Internet, la cual logró unir el Sistema Nacional de Información Científica y Tecnológica de Colciencias, y el Sistema de Información de la Educación Superior del ICFES y con el propósito que otros sistemas se pudieran conectar, resultado de ello, es la conformación de la Red Colombiana de Ciencia y Tecnología CETCOL, administrada por la Corporación Inter red. Esta red comenzó a prestar el servicio de Internet desde 1996.”[[28]](#footnote-28)

**“**MARCO LEGAL DE LA INDUSTRIA DE CONTENIDOS DIGITALES EN COLOMBIA

LEY 1341 DE 2009: UN MARCO LEGAL PROPICIO PARA EL DESARROLLO DE LOS CONTENIDOS DIGITALES

La Ley se orienta por los siguientes principios: la prioridad del acceso y uso de lasTIC; la libre competencia; el uso eficiente de la infraestructura y de los recursos escasos; la protección de los derechos de los usuarios; la promoción de la inversión; la neutralidad tecnológica; el derecho a la comunicación, la información, la educación y los servicios básicos de las TIC; y la masificación del Gobierno en Línea.

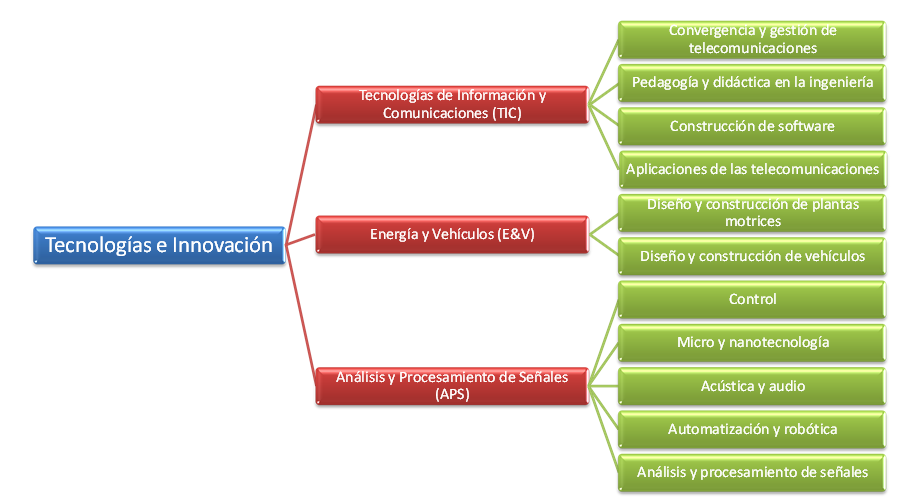
* 1. **Privacidad y Protección de Datos**

El correcto desarrollo de los contenidos digitales implica necesariamente salvaguardar la privacidad y datos personales de los usuarios de dichos contenidos.

Es necesario impulsar una armonización de las leyes de protección de datos incluido el tema de la transferencia internacional de los mismos dado el contexto global de la Red. Para ello, en este momento se está tramitando una ley de protección de datos personales que actualmente cursa su análisis constitucional en la Corte Constitucional.”[[29]](#footnote-29)

1. **LÍNEA DE INVESTIGACIÓN DE USB / SUB-LÍNEA DE FACULTAD / CAMPO TEMÁTICO DEL PROGRAMA**

Este proyecto está adscrito a la línea de investigación institucional denominada “Tecnologías e innovación”, que se encuentra en la sub-línea de la facultad de ingeniería de “Tecnologías de información y comunicaciones (TIC)”, en el campo temático del programa de “Construcción de software”.

****

**Figura 27. Línea de Investigación de USB / Sub-Línea de Facultad / Campo Temático del Programa**

1. **GLOSARIO**

**DEFINICIONES MAS CLARAS**

**MVC:**Es un patrón arquitectónico (Modelo – Vista - Controlador), la cual utiliza varios patrones de diseño, ayuda a crear sistemas confiables y fáciles de modificar.

**ARQUITECTURA DE SOFTWARE:**Proporcionará una pauta para definir la forma en que todos los componentes trabajen juntos para llevar a cabo la funcionalidad que se pretende conseguir con la aplicación.

**N-TIERS:**Una arquitectura multi- niveles, de tres niveles o N-tier son arquitecturas lógicas sobrepuestas en un modelo físico distribuido. En donde los servidores de aplicaciones pueden tener acceso a otros servidores de aplicaciones cliente.

**METADATOS:** Son datos sobre datos, en general un objeto que describe o dice algo sobre otro objeto de información.

**FRAMEWORK:** Conjunto de clases que componen un ambiente reutilizable, con una serie de funcionalidades que brindan apoyo a un desarrollo (Gamma, 2002).

**MANTENIBILIDAD:**Facilidad con la que un sistema o componente software puede ser modificado para corregir fallos, mejorar su funcionamiento u otros atributos o adaptarse a cambios en el entorno. (IEEE (19990)).

**CICLO DE VIDA:** Describe el desarrollo del software, desde la fase inicial hasta la fase final.

**APLICACIÓN WEB:**Es cualquier aplicación que es accedida vía web por una res como internet o una intranet, también se utiliza para designar aquellos programas informáticos que son ejecutados en el entorno del navegador, o codificado como algún lenguaje soportado por un navegador (Alegsa, diccionario).

**NETBEANS:** NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE). (NETBEANS)

**IDE:** Entorno integrado de desarrollo, Aplicación compuesta por un conjunto de herramientas útiles para un programador.Un entorno IDE puede ser exclusivo para un lenguaje de programación o bien, poder utilizarse para varios. Suele consistir de un editor de código, un compilador y un constructor de interfaz gráfica GUI.

**GUI:** (*Graphic User Interface o Interfaz Gráfica de Usuario*). Conjunto de formas y métodos que posibilitan la interacción de un sistema con los usuarios utilizando formas gráficas e imágenes. Con formas gráficas se refiere a botones, íconos, ventanas, fuentes, etc. los cuales representan funciones, acciones e información.

**REUTILIZACIÓN:**Es una técnica común que intenta ahorrar tiempo y energía, reduciendo el trabajo redundante.

**FLEXIBILIDAD:** Consiste en reconocer el hecho de que todo lo que puede hacerse con una máquina, puede hacerse en el lenguaje ensamblador de esta máquina; los lenguajes de alto nivel tienen en una u otra forma limitantes para explotar al máximo los recursos de la máquina.

**BASES DE DATOS:** Es un “almacén” que permite guardar grandes cantidades de información de forma organizada. (PEARSON, 2003).

**VULNERABILIDAD:** Una vulnerabilidad del software es un fallo de seguridad en una aplicación a través de la cual, un atacante puede llegar a comprometer la seguridad de todo el sistema sobre el que se ejecuta esa aplicación. (INTECO).

**JAVA:** Java es un lenguaje de programación y la primera plataforma informática creada por Sun Microsystems en 1995. Es la tecnología subyacente que permite el uso de programas punteros, como herramientas, juegos y aplicaciones de negocios. (JAVA)

**LENGUAJE DE PROGRAMACIÓN:** Lenguaje artificial que se utiliza para expresar programas de ordenador (Jesús J. Rodríguez, 2003).

**SOSTENIBILIDAD:**Cualidad por la que un elemento, sistema, o proceso, se mantiene activo con el transcurso del tiempo. Cualidad por la que un elemento se sostiene.

**CÓDIGO ABIERTO:** Código abierto es un software que pone a disposición de cualquier usuario su código fuente, es decir no es necesario pagar por utilizar dicho software.

**UML:**(Unified Modeling Language). Lenguaje único para el modelamiento de aplicaciones (OMG, 2007).

**W3C:** El Consorcio (World Wide Web) (W3C) es una comunidad internacional donde las organizaciones Miembro, personal a tiempo completo y el público en general trabajan conjuntamente para desarrollar estándares Web (W3C).

**TIC:**(Tecnologías de la información y la comunicación) nuevas tecnologías de la información y comunicación empleadas para la transmisión de contenidos a través de internet.

**SALVAGUARDAR:** Realizar una copia de seguridad.

**SERVLET:** Es un programa que se ejecuta en el contenedor web de un servidor de aplicaciones, el servlet acepta peticiones de un cliente, procesa la información relativa a la petición realizada por el cliente y le devuelve a éste los resultados que podrán ser mostrados mediante applets, páginas HTML, etc.

**ESCALABILIDAD:** Habilidad de adicionar recursos para soportar mayores números de usuarios sin modificar las aplicaciones.

**EXTENSIBILIDAD**: Habilidad de incrementar la funcionalidad de una aplicación sin alterar lo ya existente.

**SEGURIDAD:**Capacidad de extender y propagar la autenticación en el “front-end” hacia las capas intermedias, salvaguardando la información de los niveles de datos.

**COMPLEJIDAD:**En la medida en que existan mayores elementos en hardware y software en momentos de falla la complejidad de detectarla es elevada.

**COMUNICACIONES:**En la mayoría de los casos, cada capa se encuentra distribuida en la red. Lo cual genera mayor consumo de ancho de banda

1. **BIBLIOGRAFÍA**

Funcionamiento de Struts [En línea] [Consulta Octubre 2012] Disponible en <http://www.docstoc.com/docs/121292689/Patr%EF%BF%BDn-MVC-con-STRUTS>

CodeSmith – [En línea] [Consulta Agosto 2012] Disponible en <http://www.codesmithtools.com/aboutus>

Genexus - [En línea] [Consulta Octubre 2012] Disponible en [http://www.exa.unicen.edu.ar/catedras/modysim/tutoriales/manual\_de\_genexus.pd](http://www.exa.unicen.edu.ar/catedras/modysim/tutoriales/manual_de_genexus.pdf)

Patrones de Software – [En línea] [Consulta Agosto 2012] Disponible en <<http://www.xnoccio.com/es/7-una-de-patrones-parte-i/>>

Esquema de Arquitectura Centrada en páginas – [En línea] [Consulta Octubre 2012] Disponible en <<http://www.tamps.cinvestav.mx/~vjsosa/clases/sd/Arquitectura_MVC.pdf>>

Arquitecturas de software – [En línea] [Consulta agosto 2012] Disponible en <<http://www.tamps.cinvestav.mx/~vjsosa/clases/sd/Arquitectura_MVC.pdf>

Cómo funciona el patrón MVC – [En línea] [Consulta agosto 2012] Disponible en <<http://www.eugeniabahit.com/mvc/>>

Modelo de arquitectura de 3 niveles [ En línea] [Consulta agosto 2012] Disponible en <<http://msdn.microsoft.com/en-us/library/windows/desktop/ms685068(v=vs.85).aspx>>

HEIKO SCHULDT - Database and Information Systems Group,Department of Computer Science, University of Basel,Basel, Switzerland [Docuemento en PDF]

Metadatos [En línea] [Colsulta Agosto 2012] Disponible en <<http://www.scielo.br/pdf/ci/v32n2/17038.pdf/>>

NETBEANS [En línea] [Consulta Agosto 2012] Disponible en <<http://mogoyita.blogspot.com/2011/02/definicion-de-de-netbeans.html>>

UMl [En línea] [Consulta Octubre 2012] Disponible en < http://books.google.com.co/books?id=\_tKTpr4Ah88C&pg=PA37&dq=modelo+estatico+de+clases&hl=es&sa=X&ei=DpCMUKsnkubxBNKagMgH&ved=0CCoQ6AEwAA#v=onepage&q&f=false >

Modelo entidad-relacion [En línea] [Consulta Octubre 2012] Disponible en < http://www.desarrolloweb.com/articulos/modelo-entidad-relacion.html >

Ciclos de vida de un proyecto de grado [En línea] [Consulta octubre 2012] Disponible en <http://www.amazon.com/Agility-Discipline-Made-Easy-Practices/dp/0321321308>

Sitios Web accesibles [En línea] [Consulta Agosto 2012] Disponible en <http://comunicaciondigital.alieno.co/internet/%E2%80%9Cen-colombia-por-ley-es-obligatorio-tener-sitios-web-accesibles%E2%80%9D/>

**DE LA TORRE CESAR., ZORRILLA UNAI., CALVARRO JAVIER ., RAMOS MIGUEL ANGEL.,** Microsoft Arquitecture , (N-Layered Domain-Oriented Architecture Guide With,.Net 4.0), 1 ed, páginas (41-47).

Norma ISO 9126 [En línea] [Consulta agosto 2012] Disponible en <<http://normaiso9126.blogspot.com/>> QUITAR – COLOCAR NORMA APLICADA

Deitle M. Harvey, Deitel J. Paul – Cómo Programar en Java – pág 656

Contenidos Digitales [En línea] [Consulta Agosto 2012] Disponible en <http://www.vivedigital.gov.co/foros/contenidos/PoliticaContenidosDigitales.pdf>

**LISTA DE FIGURAS**

**Pág.**

Figura 1 Funcionamiento de Struts 3

Figura 2 Arquitectura de Software 5

Figura 3 Esquema de la arquitectura centrada en páginas 10

Figura 4 Esquema de la arquitectura MVC 10

Figura 5 Funcionamiento MVC 12

Figura 6 Estructura de una arquitectura 3 capas 13

Figura 7 Metadatos 15

Figura 8 UML 16

Figura 9 Modelo entidad – relación 17

Figura 10 JSP 18

Figura 11 Fases del Ciclo de Vida de un Proyecto 19

Figura 12 Fase de Inicialización 20

Figura 13 Fase – Elaboración 20

Figura 14 Fase de construcción 21

Figura 15 Fase De Transición 22

Figura 16 Riesgo 22

Figura 17 Línea de Investigación de USB / Sub-Línea 25

de Facultad / Campo Temático del Programa

1. Manual Struts 2, 06 noviembre 2009 -> [Documento en PDF] [↑](#footnote-ref-1)
2. Funcionamiento de Struts [En línea] [Consulta Octubre 2012] Disponible en <http://www.docstoc.com/docs/121292689/Patr%EF%BF%BDn-MVC-con-STRUTS> [↑](#footnote-ref-2)
3. Morras Jesús -Implementación MVC – Trabajo Fin De Carrera –J2EE [En línea ] [Consulta Agosto 2012] Disponible en < http://openaccess.uoc.edu/webapps/o2/bitstream/10609/669/1/00848tfc.pdf > [↑](#footnote-ref-3)
4. CodeSmith – [En línea] [Consulta Agosto 2012] Disponible en <http://www.codesmithtools.com/aboutus> [↑](#footnote-ref-4)
5. Genexus - [En línea] [Consulta Octubre 2012] Disponible en <http://www.exa.unicen.edu.ar/catedras/modysim/tutoriales/manual\_de\_genexus.pdf> [↑](#footnote-ref-5)
6. Patrones de Software – [En línea] [Consulta Agosto 2012] Disponible en < http://www.xnoccio.com/es/7-una-de-patrones-parte-i/> [↑](#footnote-ref-6)
7. Esquema de Arquitectura Centrada en páginas – [En línea] [Consulta Octubre 2012] Disponible en < http://www.tamps.cinvestav.mx/~vjsosa/clases/sd/Arquitectura\_MVC.pdf> [↑](#footnote-ref-7)
8. Arquitecturas de software – [En línea] [Consulta agosto 2012] Disponible en < http://www.tamps.cinvestav.mx/~vjsosa/clases/sd/Arquitectura\_MVC.pdf> [↑](#footnote-ref-8)
9. MVC – [En línea] [Consulta agosto 2012] Disponible en < http://www.tonymarston.net/php-mysql/design-patterns.html#architectural.patterns> [↑](#footnote-ref-9)
10. Cómo funciona el patrón MVC – [En línea] [Consulta agosto 2012] Disponible en < http://www.eugeniabahit.com/mvc/> [↑](#footnote-ref-10)
11. Modelo de arquitectura de 3 niveles [ En línea] [Consulta agosto 2012] Disponible en < http://msdn.microsoft.com/en-us/library/windows/desktop/ms685068(v=vs.85).aspx> [↑](#footnote-ref-11)
12. HEIKO SCHULDT - Database and Information Systems Group,Department of Computer Science, University of Basel,Basel, Switzerland [Docuemento en PDF] [↑](#footnote-ref-12)
13. Informáticos Generalitat Valenciana Grupos a Y B. Temario Bloque Específico Volumen I pág 125 [↑](#footnote-ref-13)
14. Metadatos [En línea] [Colsulta Agosto 2012] Disponible en < http://www.scielo.br/pdf/ci/v32n2/17038.pdf/> [↑](#footnote-ref-14)
15. ¿Qué son los Metadatos? [En línea] [Consulta Agosto 2012] Disponible en < http://users.dcc.uchile.cl/~cvasquez/introehistoria.pdf> [↑](#footnote-ref-15)
16. NETBEANS [En línea] [Consulta Agosto 2012] Disponible en < http://mogoyita.blogspot.com/2011/02/definicion-de-de-netbeans.html> [↑](#footnote-ref-16)
17. UMl [En línea] [Consulta Octubre 2012] Disponible en < http://books.google.com.co/books?id=\_tKTpr4Ah88C&pg=PA37&dq=modelo+estatico+de+clases&hl=es&sa=X&ei=DpCMUKsnkubxBNKagMgH&ved=0CCoQ6AEwAA#v=onepage&q&f=false > [↑](#footnote-ref-17)
18. Modelo entidad-relacion [En línea] [Consulta Octubre 2012] Disponible en < http://www.desarrolloweb.com/articulos/modelo-entidad-relacion.html > [↑](#footnote-ref-18)
19. JSP [En línea] [Consulta Octubre 2012] Disponible en < http://www.oracle.com/technetwork/java/overview-138580.html > [↑](#footnote-ref-19)
20. JSP [En línea] [Consulta Octubre 2012] Disponible en < http://onjava.com/pub/a/onjava/excerpt/jsp2\_3/index2.html > [↑](#footnote-ref-20)
21. Ciclos de vida de un proyecto de grado [En línea] [Consulta octubre 2012] Disponible en <http://www.amazon.com/Agility-Discipline-Made-Easy-Practices/dp/0321321308> [↑](#footnote-ref-21)
22. Ciclos de vida de un proyecto de grado [En línea] [Consulta octubre 2012] Disponible en <http://www.amazon.com/Agility-Discipline-Made-Easy-Practices/dp/0321321308> [↑](#footnote-ref-22)
23. Controlador JDBC MySQL – En línea [http://www.javaya.com.ar/detalleconcepto.php?codigo=132&inicio=40] Consultado en Septiembre 2013 [↑](#footnote-ref-23)
24. INFORMATION\_SCHEMA – En línea [http://dev.mysql.com/doc/refman/5.0/es/information-schema.html] Consultado en Septiembre 2013 [↑](#footnote-ref-24)
25. Ciclos de vida de un proyecto de grado [En línea] [Consulta octubre 2012] Disponible en <http://www.amazon.com/Agility-Discipline-Made-Easy-Practices/dp/0321321308> [↑](#footnote-ref-25)
26. Ciclos de vida de un proyecto de grado [En línea] [Consulta octubre 2012] Disponible en <http://www.amazon.com/Agility-Discipline-Made-Easy-Practices/dp/0321321308> [↑](#footnote-ref-26)
27. Sitios Web accesibles [En línea] [Consulta Agosto 2012] Disponible en <http://comunicaciondigital.alieno.co/internet/%E2%80%9Cen-colombia-por-ley-es-obligatorio-tener-sitios-web-accesibles%E2%80%9D/> [↑](#footnote-ref-27)
28. Internet en Colombia [En línea] [Consulta Agosto 2012] Disponible en < http://lac.derechos.apc.org/investigacion/tic\_colombia.pdf> [↑](#footnote-ref-28)
29. Contenidos Digitales [En línea] [Consulta Agosto 2012] Disponible en <http://www.vivedigital.gov.co/foros/contenidos/PoliticaContenidosDigitales.pdf> [↑](#footnote-ref-29)