

**UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS
FACULTAD 4
INGENIERÍA EN CIENCIAS INFORMÁTICA**



**TÍTULO: GENERADOR DE CÓDIGO PARA APLICACIONES EN PHP APLICADO AL
ERPFAR.**

**TRABAJO PARA OPTAR POR EL TÍTULO
DE INGENIERO INFORMÁTICO**

AUTOR.

Leevan Abon Cepeda.

TUTOR:

Ing. Rolando Ramírez Concepción

Ciudad de la Habana
Junio del 2007

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los _____ días del mes de _____ del _____.

Leevan Abon Cepeda

Ing. Rolando Ramirez Concepción



Hay hombres que luchan un día y son buenos. Hay otros que luchan un año y son mejores. Hay quienes luchan muchos años y son muy buenos. Pero hay los que luchan toda la vida: esos son los imprescindibles.

Bertoldt Brecht

AGRADECIMIENTOS:

A Fidel, por confiar en esta tropa del futuro de la cual formo parte.
A Tte. Cor. Raúl Miguel Triana Díaz por confiar en mí en todo momento.
A mi tutor, por ayudarme y apoyarme en todo lo que me ha hecho falta.
A las personas que me han apoyado y ayudado en la elaboración de mi tesis.
A las personas que me han soportado en mi tormentoso andar por la vida.

DEDICATORIA:

Quisiera dedicarles esta tesis a mis padres, por la simple razón de haberme dado el placer de vivir y disfrutar de este hermoso mundo. Así como a todas aquellas personas que de un modo u otro han formado y forman parte de mi vida y mi corazón....

RESUMEN:

La generación de código en el mundo se ha hecho algo imprescindible, pues el ahorro de tiempo, la eficiencia en la programación y la estandarización de código son los pilares fundamentales para la construcción de un proyecto. El siguiente trabajo pretende desarrollar una herramienta para generar la mayor cantidad de código posible en las aplicaciones programadas en PHP.

La automatización de estos procesos trae consigo un cambio radical en la construcción de proyectos, pues el programador solo tendría que concentrarse en la Lógica de Negocio, pues el resto de las tareas pueden ser generadas, ya que en tiempo anteriores se perdía un mucho tiempo creando plantillas y programando la Capa de Acceso a Datos, cosa que con el Establecimiento de este software se erradica, pues queda generado todo este código con solo especificarle algún que otro parámetro.

Esta aplicación, la cual se comprenderá de varios módulos, los cuales son Generación de Acceso a Datos, Generación de Interfaz, Generación de Proyecto y Generación de Conexión, esta bien concebida para un fácil manejo por el programador, ayudándolo grandemente a la hora de programar y siguiendo los estándares de código que se aplican en el ERP FAR, así como una arquitectura sólida y robusta que permite una aplicación eficiente.

El resultado más relevante de este software consiste en el ahorro del tiempo y alcance de un estadio superior en la calidad de software pues se estandarizaría el código de las aplicaciones generadas. Además de ser una herramienta hecha para los propios programadores, ayudando a facilitar su trabajo, el cual suele ser en ocasiones engorroso y difícil, constituyendo un paso de avance en el mejoramiento de las condiciones de trabajo.

Palabras claves: Generador de Código, ahorro de tiempo, Arquitectura.

INDICE:

AGRADECIMIENTOS.....	I
DEDICATORIA.....	II
RESUMEN.....	III
INTRODUCCIÓN.....	1
CAPITULO I: FUNDAMENTACIÓN TEÓRICA.....	4
Introducción	4
1.1- ¿Que es la generación de código?.....	4
1.2- Tipos de generación de código	5
1.3- Creación de generadores de código.	5
1.4- Softwares Generadores de Código	6
1.5- Desventajas de los mismos.....	7
1.6- Arquitectura Generada.....	8
1.7- Tecnologías utilizadas.....	22
1.8- Herramientas utilizadas.....	26
Conclusiones	28
CAPITULO II: CARACTERÍSTICAS DEL SISTEMA.....	29
Introducción	29
2.1 Análisis crítico de los procesos actuales	29
2.2 Objeto de automatización.....	29
2.3 Descripción general de la propuesta de sistema	30
2.4 Modelo del Negocio.....	30
2.5 Descripción del caso de uso del negocio.....	32
2.6 Diagramas de Actividades del Negocio	37
2.7 Modelo de Objeto.....	41
2.8 Requerimientos Funcionales	42
2.9 Requerimientos no funcionales	42
2.10 Casos de uso y actores del sistema	45
2.10.1 Definición de los actores.....	45
2.10.2 Listado de casos de uso.	45
2.11 Diagrama de casos de uso.....	47
2.12 Descripción de los Casos de uso expandida	48
Conclusiones	63

CAPITULO III: ANÁLISIS Y DISEÑO DEL SISTEMA	64
Introducción:	64
3.1 Análisis.....	64
3.1.1 ¿Qué es el Análisis?.....	64
3.1.2 Diagramas de clases del Análisis	64
3.2 Diseño.....	68
3.2.1 ¿Qué es el diseño?	68
3.2.2 Diagramas de clases del Diseño	69
3.3 Mecanismos de Diseño:	77
3.3.1 Mecanismos de Seguridad	77
3.3.2 Mecanismos de Acceso a Datos:.....	77
3.4 Diagramas de Secuencia	78
3.4.1 Diagrama de secuencia.....	78
3.4.2 Diagramas de secuencia de la aplicación.....	79
3.5 Estándares de diseño.....	83
3.6 Tratamiento de Errores	83
3.7 Descripción de las clases	84
Conclusiones:	89
CAPITULO IV: IMPLEMENTACIÓN Y PRUEBA.....	90
Introducción	90
4.1 Diagrama de Despliegue.....	90
4.2 Diagrama de Componentes.....	91
4.3 Prueba	97
Conclusiones	100
CONCLUSIONES	101
RECOMENDACIONES.....	102
BIBLIOGRAFIA.....	103
GLOSARIO.....	104

INTRODUCCIÓN.

Actualmente el MINFAR está enfrascado en el proceso de informatización de sus principales entidades, desplegando un avanzado soporte de infocomunicaciones con considerables capacidades de almacenamiento y procesamientos de información permitiéndole distribuir, desarrollar e integrar sus principales sistemas informativos. En el ámbito de desarrollo encontramos un fuerte vínculo con la Universidad de las Ciencias Informáticas, donde una de sus principales líneas de trabajo esta enfocada fundamentalmente a la construcción de aplicaciones en entorno Web, utilizando como lenguaje de programación fundamental PHP, debido a las potencialidades que el mismo brinda y ofrece.

En las aplicaciones construidas sobre un entorno de desarrollo PHP aparecen una serie de patrones comunes en cualquiera de las capas existentes así como a la hora de la programación Orientada a Objeto, por lo que muchas veces los programadores tienen que repetir el mismo código varias veces cuando desarrollan una aplicación siendo esto una **situación problemática** en cualquier proyecto perdiéndose tiempo innecesariamente en su construcción y a pesar de que el código es el mismo, se cometan errores. Todo esto trae consigo una falta de eficiencia muy grande a la hora de la programación.

En el mundo existen varios software con una idea similar a la planteada, pero sin una arquitectura bien definida a la hora de generar el código. De aquí que el **Problema Científico** podemos describirlo de la siguiente manera: ¿Cómo lograr erradicar los problemas de pérdida de tiempo y de errores cometidos que se les presenta a los programadores al desarrollar una aplicación PHP?

Este trabajo tiene como **Objeto de Estudio** el proceso de desarrollo de software automatizado utilizando código PHP en la creación del ERPFAR para aplicaciones de gestión.

Encontrándose dentro de los **Objetivos de la Investigación:**

Generales:

- Diseño e implementación de una herramienta generadora de código PHP en su primera versión para ser utilizada en el ERPFAR.

Específicas:

- Investigación de una metodología que facilite y garantice confeccionar un sistema informático para la generación de código.
- Estudio de los programas generadores de código en el mundo para obtener sus experiencias y ventajas, así como para analizar sus desventajas para prever problemas en la propuesta planteada.
- Desarrollar una serie de Patrones estándares a la hora de programar una aplicación Web.
- Realizar el Diseño de la herramienta propuesta, así como la implementación de su primera versión.
- Generar la estructura de un proyecto, así como su portal principal, su fichero de conexión, las clases de acceso a datos, sus estilos básicos y sus formularios.

El **Campo de acción** es muy amplio y del Objeto de estudio analizado, podemos definir que el Campo de acción es el desarrollo de software automatizados en la creación del ERP FAR en la Universidad de las Ciencias Informáticas.

Ahora bien como **Hipótesis** podemos plantear que si se desarrolla una aplicación que genere código con una arquitectura bien definida se logrará trabajar con mucha más eficiencia y se disminuiría la cantidad de errores cometidos por los programadores.

Dentro de las **Tareas de la investigación** encontramos:

- Selección de las herramientas para llevar a cabo el proyecto y la elección de la plataforma en la que se desarrollará la aplicación. Fundamentando su elección.
- Selección de la metodología de Análisis y Diseño de sistemas informáticos, que facilite y garantice la creación con calidad del sistema.
- Implementación de la aplicación hasta obtener una primera versión la cual puede ser mejorada y ampliada con nuevos módulos.
- Investigar los programas generadores de código en el mundo para la obtención de experiencias y mejoras en una nueva versión.

Definiéndose en este trabajo de tesis cuatro capítulos:

- **Capítulo I.** Fundamentación teórica: Se expone el estado del arte del problema antes mencionado abarcando las tendencias existentes, las técnicas utilizadas, la tecnología empleada, la metodología de desarrollo y los Softwares existentes en el mundo en la actualidad.
- **Capítulo II.** Características del sistema: Se trata el objeto de informatización, logrando una propuesta del sistema, modelando el negocio, especificando requisitos funcionales y no funcionales que debe cumplir el sistema, y definiendo los casos de uso.
- **Capítulo III.** Análisis y diseño del sistema: Se define el modelo de análisis, el modelo de clases de análisis, la descripción de las clases, el modelo de clases del Diseño y los diagramas de interacción por cada realización de casos de uso la descripción de las clases.
- **Capítulo IV.** Implementación y prueba: Se define los diagramas de Despliegue y Componentes, así como una serie de pruebas realizadas a la aplicación.

CAPITULO I: FUNDAMENTACIÓN TEÓRICA.

Introducción:

En el presente capítulo podemos encontrar todo lo relacionado con los generadores de código, sus características, y una descripción de los principales conceptos asociados al problema y que son necesarios para entender el negocio y la propuesta de solución.

Se especifican y argumentan las principales técnicas y herramientas usada para la construcción de dicho software. Además de analizar profundamente los proyectos y aplicaciones que se relacionan con la problemática planteada.

1.1- ¿Que es la generación de código?

En términos más generales, la generación de código es usada para construir programas de una manera automática evitando que los programadores tengan que escribir el código a mano. Constituyendo un ahorro de tiempo en el desarrollo de proyectos y aplicaciones.

La generación de código data desde la existencia de los primeros compiladores. Hasta la aparición de los primeros generadores de código comercial u orientado a "usuarios finales"; la generación de código era exclusividad de programas compiladores especializados.

En tiempos más recientes la generación de código, gracias al avance de la ingeniería del software, se ha llevado a un nivel diferente; donde se encuentran programas generadores de pantallas, reportes y consultas, estas son herramientas de gran utilidad; pero se debe, en la mayoría de los casos, pagar una gran cantidad de dinero por ellos. [1]

Todavía el mundo se encuentra avanzando y dando pasos firmes en el tema de generación de código, pues no se puede hablar de una herramienta eficaz que cumpla con todos los requerimientos necesarios en la actualidad, pero tampoco existe una solida preparación en cuanto al tema, pues todo programador se enfrasca en el código que debe confeccionar y no en la búsqueda de una herramienta que le puede ayudar a generar parte de ese código.

1.2- Tipos de generación de código:

- **Templating.** Se genera un armazón (o esbozo) de código fuente no funcional para ser editado, con el que se evita tener que escribir la parte más repetitiva del código (generalmente poco compleja). Suele ser una opción recomendable.
- **Parcial.** Se genera código fuente que implementa parcialmente la funcionalidad requerida, pero que el programador usará como base para modificar, integrar y/o adaptar a sus necesidades. No suele ser recomendable. Por ejemplo: generamos una aplicación para el mantenimiento de tablas de una Base de datos.
- **Total.** Se genera código fuente funcionalmente completo pero que no va a ser modificado por el programador, sino que si es necesario se vuelve a regenerar. Por lo general tampoco suele ser un código excesivamente complejo. Recomendable.

1.3- Creación de generadores de código.

Para la creación de generadores de código se deben considerar los siguientes aspectos:

- La arquitectura de software para la cual se va a desarrollar el generador.
- Las características específicas del lenguaje de programación.
- El lenguaje con el que se desarrollará el propio generador.
- Responder las interrogantes: ¿La generación de código se realizará a partir de modelos como Uml? ¿La generación de código se hará a partir de las tablas de una base de datos?, ¿Se realizará un generador de código que su resultado sea fragmentos de código que son de uso más frecuente en el software? ¿Se creará un generador genérico que "genere" código para diferentes lenguajes?
- Las reglas de utilización del generador, en otras palabras, la forma adecuada para que los usuarios del generador obtengan el mayor provecho.

En síntesis para crear un generador de código se deben hacer muchas de las tareas que realizan los compiladores; algunas de estas tareas son: la búsqueda de patrones, la escritura de código, el análisis sintáctico, el análisis léxico y la optimización de código. Estas tareas las realiza el desarrollador una vez, para una arquitectura específica.

1.4- Softwares Generadores de Código:

Dentro de los programas generadores de códigos podemos encontrar el **VISUALWADE**, **MAKE ME FEEL GOOD**, el **CODECHARGE STUDIO** y **CLARION**.

VISUALWADE, generador visual de código PHP

Es un generador de código PHP gratuito que produce aplicaciones web desde una interface de diseño de modelos. Puede realizar conexiones a base de datos relacionales como MySQL, PostgreSQL u Oracle. Aunque sólo para Windows y requiriendo de nuestra registración para descárgalo.

CODECHARGE STUDIO

Es una solución muy productiva para visualmente crear aplicaciones Web con la cantidad mínima de codificación. Es bastante fácil y amigable. Genera conexión con base de datos Access, MS SQL, MySQL, Oracle. Pero además es capaz de generar los códigos en PHP, ASP, JSP, Perl, ColdFusion y ASP.NET. Este sistema tiene un gran problema, pues es software propietario y tiene un costo en el mercado de **\$139.95**. [2]

MAKE ME FEEL GOOD

Es un script que genera todo un sistema de panel de control, con formularios de inserción, edición y listados, creando al vuelo los documentos necesarios a partir de una base de datos en MySQL y sin tener que tocar ni una sola línea de código PHP y todo realizado mediante asistentes.

La utilización de esta herramienta para la realización de una página web o aplicación supone ahorrar horas de trabajo y todo ello sin tener que pelearse con código PHP, HTML o CSS. [3]

CLARION

Es un generador rápido de aplicaciones de base de datos, que permite generar aplicaciones tanto para ambiente Windows, como para Internet / Intranet. CLARION es un producto de SoftVelocity y está disponible en dos versiones: Profesional y Enterprise. Es una aplicación de escritorio que corre sobre cualquier Windows, pero no es multiplataforma.

En cuanto a la conectividad con bases de datos, CLARION viene integrado a una base de datos TopSpeed que provee de gran funcionalidad, robustez y acceso a datos. Así mismo CLARION permite conectarse nativamente con la mayoría de las base de datos Xbase como Dbase, Fox, Clipper, etc., así como a motores de base de Datos SQL como MS-SQL, Sybase, Oracle entre otros, y vía ODBC a cualquier base de datos. Para generar el código utiliza el método de asistente. [4]

1.5- Desventajas de los mismos.

Todos los programas antes descritos son muy buenos y están hechos con el objetivo de generar código, pero presentan problemas como son:

- Generan código solo referente a los métodos básicos en las conexiones a la Base de datos, es decir, insertar, modificar, eliminar y en casos muy específicos consultas.
- No están orientados a una arquitectura, y para especificarle una se debe realizar todos los modelos correspondientes a la misma, siendo esto un trabajo muy engorroso y difícil de modelar.
- A veces deben ser demasiado detallados los diagramas para generar un simple código y otras veces no es posible a través de diagramas especificar lo que se quiere hacer.
- La mayoría de los generadores de código son aplicaciones de escritorio.

El software propuesto estará basado en un asistente y no en diagramas. Desde los años 70 se nos viene hablando de la generación de código a partir de modelos visuales, lo que ha degenerado en los lenguajes gráficos tipo UML, las herramientas CASE, y demás. Es posible generar código a partir de un lenguaje gráfico con UML, pero es tan práctico como escribir un libro mediante jeroglíficos. Cualquiera que haya usado UML para generar código sabe que al final tienes que detallar tanto el modelo gráfico para que genere código útil, que habría resultado más fácil hacerlo mediante un lenguaje no gráfico.

Con esta aplicación se puede generar código basado en una arquitectura específica la cual también puede ser modificada por una persona que tenga conocimientos avanzados del funcionamiento de la aplicación.

Esta nueva herramienta esta diseñada sobre plataforma Web, con una metodología de trabajo en equipo y generando una arquitectura solida y bien definida, la cual a sido el resultado de un estudio profundo de un conjunto de especialistas. Con su terminación y puesta en funcionamiento, el programador solo tendría que preocuparse en mayor medida por el trabajo en la Lógica de Negocio, pues el resto de los componentes de la arquitectura serian generados de forma dinámica.

1.6- Arquitectura Generada.

Una aplicación o un servicio se compone de varios componentes, así como el modo en que cada uno de los cuales realiza una tarea diferente. Todas las soluciones de software contienen tipos de componentes similares, independientemente de las necesidades que deban cubrir. Por ejemplo, la mayoría de las aplicaciones contienen componentes que tienen acceso a datos, encapsulan reglas empresariales y controlan la interacción con el usuario, entre otros. La identificación de los tipos de componentes que se encuentran normalmente en las soluciones de software facilitará la elaboración de un plano técnico para el diseño de aplicaciones o servicios.

Tipos de componentes

El análisis de la mayoría de las soluciones basadas en modelos de componentes por capas muestra que existen varios tipos de componentes habituales. En la figura 1.1 se muestra una ilustración completa en la que se indican estos tipos de componentes.

Aunque la lista que se muestra en la figura no es completa, representa los tipos de componentes de software más comunes encontrados en la mayoría de las soluciones con una arquitectura en capas, orientada a objetos y servicios. A lo largo del documento describiremos en profundidad cada uno de estos.

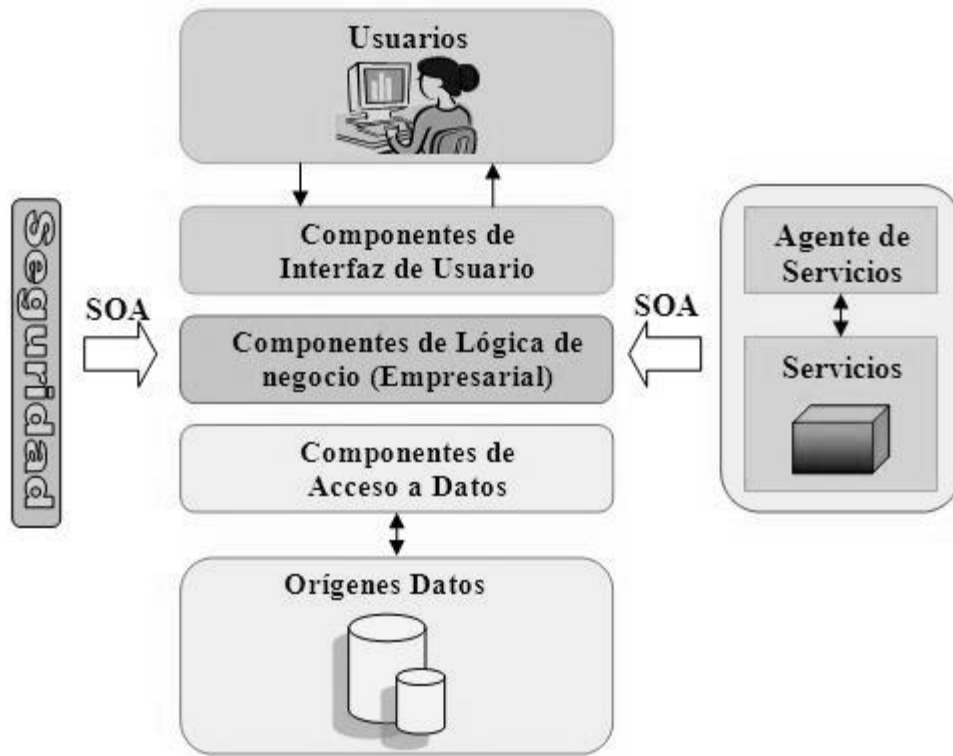


Figura 1.1. Arquitectura en capas. Tipos de componentes utilizados.

Los tipos de componentes identificados en el escenario de diseño son:

1. **Componentes de interfaz de usuario (IU).** La mayor parte de las soluciones necesitan ofrecer al usuario un modo de interactuar con la aplicación. Las interfaces de usuario se implementan utilizando formularios, controles u otro tipo de tecnología que permita procesar y dar formato a los datos de los usuarios, así como adquirir y validar los datos entrantes procedentes de éstos. En un gran número de casos, la interacción del usuario con el sistema se realiza de acuerdo a un proceso predecible.
2. **Componentes de lógica de negocio.** Independientemente de si el proceso consta de un único paso o de un flujo de trabajo organizado, la aplicación requerirá probablemente el uso de componentes que implementen reglas empresariales y realicen tareas empresariales. Por ejemplo, en la aplicación comercial, deberá implementar una funcionalidad que calcule el precio total del pedido y agregue el costo adicional correspondiente por el envío del mismo. Los componentes empresariales implementan la lógica empresarial de la aplicación.

3. **Componentes de acceso a datos.** La mayoría de las aplicaciones y servicios necesitan obtener acceso a un almacén de datos en un momento determinado del proceso empresarial. Por ejemplo, la aplicación empresarial necesita recuperar los datos de los productos de una base de datos para mostrar al usuario los detalles de los mismos, así como insertar dicha información en la base de datos cuando un usuario realiza un pedido. Por tanto, es razonable abstraer la lógica necesaria para obtener acceso a los datos en una capa independiente de componentes lógicos de acceso a datos, ya que de este modo se centraliza la funcionalidad de acceso a datos y se facilita la configuración y el mantenimiento de la misma.
4. **Servicios.** Cuando un componente empresarial requiere el uso de la funcionalidad proporcionada por un servicio externo, tal vez sea necesario hacer uso de código para administrar la semántica de la comunicación con dicho servicio. Los agentes de servicios permiten aislar las idiosincrasias de las llamadas a varios servicios desde la aplicación y pueden proporcionar servicios adicionales, como la asignación básica del formato de los datos que expone el servicio al formato que requiere la aplicación.
5. **Componentes de seguridad:** La directiva de seguridad se ocupa de la autenticación, autorización, comunicación segura, auditoria y administración de perfiles.

Diseño de capas de presentación

La capa de presentación contiene los componentes necesarios para habilitar la interacción del usuario con la aplicación. Las capas de presentación más simples contienen componentes de interfaz, como formularios Web. Las interacciones más complejas conllevan el diseño de componentes de proceso de usuario que permiten organizar los elementos de la interfaz y controlar la interacción con el usuario. Los componentes de proceso de usuario resultan especialmente útiles cuando la interacción del usuario sigue una serie de pasos predecibles, como al utilizar un asistente para realizar una tarea determinada.

Las interfaces de usuario constan normalmente de una página o formulario con varios elementos que permiten mostrar datos y aceptar la entrada del usuario. Por ejemplo, una aplicación puede contener un control DataGridView que muestre una lista de categorías de productos y un control de botón de comando que indica que el usuario desea ver los productos de la categoría seleccionada. Cuando un usuario interactúa con un elemento de la interfaz, se genera un evento que llama al código de una función de control. Esta, a

su vez, llama a componentes empresariales, componentes lógicos de acceso a datos o componentes de proceso de usuario para implementar la acción deseada y recuperar los datos que se han de mostrar. A continuación, la función de control actualiza los elementos de la interfaz.



Figura 1.2. Diseño de interfaz de usuario

Funcionalidad de los componentes de interfaz de usuario

Los componentes de la interfaz de usuario deben mostrar datos al usuario, obtener y validar los datos procedentes del mismo e interpretar las acciones de éste que indican que desea realizar una operación con los datos. Asimismo, la interfaz debe filtrar las acciones disponibles con el fin de permitir al usuario realizar sólo aquellas operaciones que le sean necesarias en un momento determinado.

Los componentes de interfaz de usuario:

- No inicializan, participan ni votan en transacciones.
- Presentan una referencia al componente de proceso de usuario actual si necesitan mostrar sus datos o actuar en su estado.
- Pueden encapsular tanto la funcionalidad de visualización como un controlador.

Al aceptar la entrada del usuario, los componentes de la interfaz:

- Adquieren los datos del usuario y atienden su entrada utilizando guías visuales (como informaciones sobre herramientas) y sistemas de validación, así como los controles necesarios para realizar la tarea en cuestión.
- Capturan los eventos del usuario y llaman a las funciones de control para indicar a los elementos de la interfaz de usuario que cambien el modo de visualización de los datos, bien inicializando una acción en el proceso de usuario actual, o bien, modificando los datos del mismo.

- Restringen los tipos de entrada del usuario. Por ejemplo, un campo puede limitar las entradas del usuario a valores numéricos.
- Realizan la validación de entrada de datos, por ejemplo, restringiendo el intervalo de valores que se pueden escribir en un campo determinado, o garantizando que se escriben los datos obligatorios.
- Llevan a cabo la asignación y transformación simple de la información proporcionada por los controles del usuario en los valores necesarios para que los componentes subyacentes realicen su trabajo (por ejemplo, un componente de interfaz de usuario puede mostrar el nombre de un producto pero pasar el Id. del mismo a los componentes subyacentes).
- Interpretar las acciones del usuario (como las operaciones de arrastrar y colocar o los clics de botones) y llamar a una función de control.
- Pueden utilizar un componente de utilidad para el almacenamiento de datos en caché.
- Pueden utilizar un componente de utilidad para realizar la paginación. Es frecuente, especialmente en las aplicaciones Web, mostrar largas listas de datos como conjuntos paginados. Asimismo, se suele disponer de un componente de ayuda para realizar el seguimiento de la página actual en la que se encuentra el usuario y, por tanto, invocar a las funciones de consulta paginada de los componentes lógicos de acceso a datos con los valores adecuados relativos al tamaño de página y página actual. La paginación se puede realizar sin la interacción del componente de proceso de usuario.

Capa de Lógica de Negocios.

La parte más importante de la aplicación es la funcionalidad que proporciona. Una aplicación realiza un proceso empresarial que consta de una o varias tareas. En los casos más simples, cada tarea se puede encapsular en un método de un componente y llamar de forma sincrónica o asincrónica. Para los procesos empresariales más complejos que requieren varios pasos y transacciones de ejecución larga, la aplicación necesita disponer de un modo de organizar las tareas empresariales y almacenar el estado hasta que el proceso se haya completado.

Puede diseñar la lógica en las capas empresariales para su uso directo por parte de componentes de presentación o su encapsulación como servicio y llamada a través de una interfaz de servicios, que coordina la conversación asincrónica con los llamadores del servicio. La parte principal de la lógica empresarial se suele denominar lógica de *dominio o negocio*. Los componentes empresariales también

pueden realizar solicitudes de servicios externos, en cuyo caso tal vez sea preciso implementar agentes de servicios para administrar la conversación requerida para la tarea empresarial específica realizada por cada uno de los servicios que necesita utilizar.

Al implementar lógica empresarial, es necesario decidir si es preciso organizar o no el proceso empresarial, o si será suficiente con disponer de un conjunto de componentes empresariales.

Diseño de componentes empresariales

Los componentes empresariales pueden ser la raíz de las transacciones atómicas. Éstos implementan las reglas empresariales en diversos patrones y aceptan y devuelven estructuras de datos simples o complejas. Los componentes empresariales deben exponer funcionalidad de modo que sea independiente de los almacenes de datos y los servicios necesarios para realizar la tarea, y se deben componer de forma coherente desde el punto de vista del significado y transaccional.

Si el proceso empresarial invocará a otros procesos empresariales en el contexto de una transacción atómica, todos los procesos invocados deben garantizar que sus operaciones participan en la transacción existente de modo que las operaciones se deshagan en caso de que la lógica empresarial que realiza las llamadas se interrumpa. Una técnica muy segura es volver a intentar una operación atómica si ésta da error, sin miedo a que los datos pierdan su coherencia. Considere el límite de una transacción determinada como un límite de reintento.

En la siguiente lista se resumen las recomendaciones relativas al diseño de componentes empresariales:

- Básese lo más que pueda en la comunicación basada en mensajes.
- Asegúrese de que los procesos expuestos en las interfaces de servicios no se pueden alterar y que, por tanto, el estado de la aplicación o el servicio no perderá su coherencia si se recibe el mensaje dos veces.
- Elija con cuidado los límites de la transacción de modo que se puedan realizar reintentos y composiciones. Esto se aplica tanto a las transacciones atómicas como a las de ejecución larga. Asimismo, debe considerar el uso de reintentos para sistemas basados en mensajes, sobre todo al exponer la funcionalidad de la aplicación como un servicio.

- Los componentes empresariales se deben poder ejecutar en el contexto de cualquier usuario de servicio; no necesariamente suplantando a un usuario de una aplicación específica. Esto permite su invocación con mecanismos que ni transmiten ni delegan la identidad del usuario.
- Elija y mantenga un formato de datos coherente (como XML, conjunto de datos, etc.) para los parámetros de entrada y los valores de devolución.

Los componentes empresariales son llamados por los siguientes clientes:

- Interfaces de servicios.
- Componentes de proceso de usuario.
- Flujos de trabajo empresariales.
- Otros componentes empresariales.

En la figura se muestra un componente empresarial típico que interactúa con los componentes lógicos de acceso a datos, las interfaces y los agentes de servicios y otros componentes empresariales.

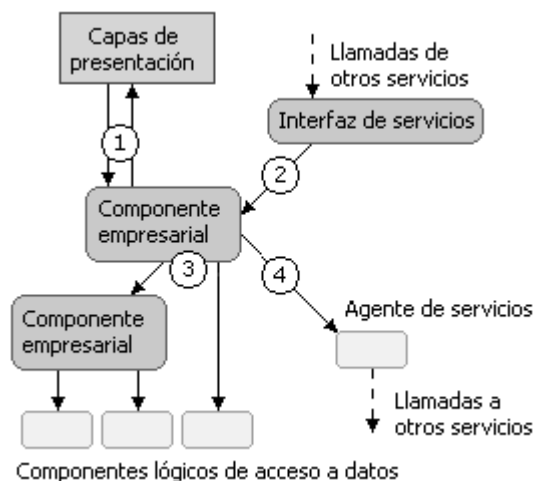


Figura 1.3. Componentes empresariales

Observe los siguientes puntos:

1. Los componentes empresariales pueden ser invocados por los componentes de las capas de presentación.

2. Los componentes empresariales pueden ser invocados por interfaces de servicios (por ejemplo, un servicio Web XML).
3. Los componentes empresariales pueden llamar a componentes lógicos de acceso a datos para recuperar y actualizar datos, y pueden invocar a otros componentes empresariales.
4. Los componentes empresariales también pueden invocar a agentes de servicios. Tenga especial cuidado al diseñar la lógica de compensación en caso de que el servicio al que desea tener acceso no esté disponible o lleve demasiado tiempo devolver una respuesta.

Nota Las flechas que aparecen en la figura representan el flujo de control, no el flujo de datos.

Diseño de capas de acceso a datos y almacén de datos.

Casi todas las aplicaciones y servicios necesitan almacenar y obtener acceso a un determinado tipo de datos. Por ejemplo, la aplicación comercial descrita en esta guía necesita almacenar datos de productos, clientes y pedidos.

Al trabajar con datos debe determinar:

- El almacén de datos que utiliza.
- El diseño de los componentes utilizados para obtener acceso al almacén de datos.
- El formato de los datos pasados entre componentes y el modelo de programación necesario para ello.

La aplicación o servicio puede disponer de uno o varios orígenes de datos, los cuales pueden ser de tipos diferentes. La lógica utilizada para obtener acceso a los datos de un origen de datos se encapsulará en *componentes lógicos de acceso a datos* que proporcionan los métodos necesarios para la consulta y actualización de datos. Los datos con los que la lógica de la aplicación debe trabajar están relacionados con *entidades* del mundo empresarial que forman parte de la empresa. En determinados escenarios, puede disponer de componentes personalizados que representan estas entidades, mientras que en otros puede decidir trabajar con datos utilizando directamente conjuntos de datos o documentos XML.

La mayoría de las aplicaciones utilizan una base de datos relacional como almacén principal de los datos de la aplicación.

Cuando la aplicación recupera datos de la base de datos, puede hacerlo utilizando un formato de conjunto de datos (PDO). A continuación los datos se transfieren entre las capas y los distintos niveles de la aplicación y, finalmente, uno de los componentes los utilizará. Tal vez desee utilizar formatos de datos diferentes para recuperar, pasar y utilizar datos; por ejemplo, puede utilizar los datos de un conjunto de datos para llenar las propiedades de un objeto de entidad personalizado. No obstante, debería intentar mantener una coherencia en cuanto al tipo de formato utilizado, ya que mejorará probablemente el rendimiento y la facilidad de mantenimiento de la aplicación para presentar sólo un conjunto limitado de formatos, evitando así la necesidad de capas de traducción adicionales y de familiarizarse con API diferentes.

Diseño de la directiva de seguridad

La directiva de seguridad se ocupa de la autenticación, autorización, comunicación segura, auditoría y administración de perfiles, tal como muestra la figura 1.4.



Figura 1.4. Aspectos de la directiva de seguridad

Principios generales sobre seguridad

Existen ciertos principios generales sobre seguridad que se deben tener en cuenta a la hora de desarrollar una directiva de seguridad. Hay que tener en cuenta las siguientes directrices:

- Siempre que sea posible, se debe recurrir a sistemas de seguridad que se hayan comprobado y demostrado su eficacia en lugar de generar su propia solución personalizada.
- Nunca confiar en las aportaciones externas. Deberá validar todos los datos que introduzcan los usuarios o envíen otros servicios.

- Considerar por principio que los sistemas externos no son seguros. Si su aplicación recibe datos confidenciales sin cifrar desde un sistema externo, asuma que dicha información no es segura.
- Aplicar el principio del menor privilegio. No habilitar más atributos en las cuentas de servicios que los que resulten estrictamente necesarios para la aplicación. Obtener acceso a los recursos con cuentas que tengan los mínimos permisos necesarios.
- Reducir el área de superficie. El riesgo se incrementa según aumenta el número de componentes y datos que haya expuesto a través de la aplicación y, por lo tanto, se deberá exponer únicamente la funcionalidad que se crea que otros van a utilizar.
- Establecer como predeterminado un modo seguro. No habilitar servicios, tecnologías y derechos de cuenta que no sean absolutamente necesarios. Cuando se implemente la aplicación en equipos cliente o servidor, la configuración predeterminada de esta deberá ser segura.
- No confiar en la seguridad a través del ocultamiento. El cifrado de los datos implica disponer de claves y de un algoritmo de cifrado demostrado. El almacenamiento de los datos seguros evitará el acceso a ésta en cualquier circunstancia. No se puede considerar seguridad la mezcla de diversas cadenas, el almacenamiento de la información en rutas de archivo inesperadas y demás técnicas similares.
- Seguir los principios de STRIDE. (STRIDE responde a las siglas inglesas de Simulación, Alteración, Repudio, Revelación de información, Denegación de servicio y Elevación de privilegios). Todas estas son clases de vulnerabilidades de la seguridad contra los que un sistema se debe proteger.
- Realizar la comprobación desde la misma puerta. No permitir que los procesos vayan más allá del lugar para el que los usuarios están autorizados.
- Bloquear su sistema interna y externamente: los usuarios y operadores internos pueden representar un riesgo igual que los intrusos externos.

Autenticación

La autenticación se define como identificación segura, que básicamente quiere decir que dispone de un mecanismo para identificar con seguridad a los usuarios que se adecua a los requisitos de seguridad de su aplicación.

La autenticación se implementa en la capa de la interfaz de usuario para proporcionar funciones de autorización, auditoría y personalización. Esto generalmente requiere que el usuario escriba sus credenciales (como por ejemplo, nombre y contraseña) para demostrar su identidad. Entre otros tipos de credenciales se incluyen las lecturas biométricas, tarjetas inteligentes, claves físicas, certificados digitales, etc.

Si la aplicación se expone como servicio, se debe autenticar también en ciertas interfaces de servicio para asegurarse de que se compromete en un intercambio conocido y de confianza, así como que otros servicios externos no simulan su aplicación para hacer creer que quien llama es otra aplicación.

En el diseño, se debe perseguir como objetivo disponer de una lógica empresarial que sea transparente en el proceso de autenticación. Por ejemplo, no es aconsejable tener un parámetro adicional en los métodos de componentes sólo para pasar información del usuario, a menos que la función empresarial lo requiera.

Autorización

El aspecto de la autorización de la directiva de seguridad se ocupa de la identificación de las acciones permitidas para cada principal de seguridad autenticado. En otras palabras, la directiva de seguridad determina quién puede hacer qué. Para determinar la directiva de autorización, deberá tener en cuenta dos factores principales:

- Los permisos y derechos de usuario.
- La seguridad de acceso al código.

Los permisos y derechos de usuario determinan lo que se permite hacer en una cuenta de usuario en el contexto de la aplicación. Técnicamente, el término "permisos" se refiere a las acciones permitidas en un recurso (como por ejemplo, un archivo o una tabla de base de datos), mientras que los "derechos" hacen referencia a las tareas del sistema que se permite realizar al usuario (como por ejemplo, configurar la hora del sistema o apagar el equipo). Los permisos y derechos de usuario se pueden asignar de forma individual para cada usuario, si bien resultan más fáciles de administrar cuando los usuarios se organizan de una manera lógica en grupos o funciones. La mayor parte de los recursos tienen algún tipo de lista de

permisos relacionada, en la que se indican los permisos asignados a los usuarios para ese determinado recurso.

La seguridad de acceso al código, ofrece a los desarrolladores y administradores una dimensión adicional de control de acceso, así como la posibilidad de comprobar de nuevo la configuración de seguridad correcta.

Auditoría

En muchos casos, se necesitará implementar la funcionalidad de auditoría para realizar el seguimiento del usuario y la actividad empresarial en la aplicación por motivos de seguridad. Para realizar la auditoría de las actividades empresariales, necesita una ubicación de almacenamiento segura; de hecho, la auditoría se puede considerar como un "registro seguro". Al implementar una solución de auditoría, se deberá asegurar de que las entradas de auditoría no se puedan modificar o al menos permitan reconocer si han sido alteradas (mediante el uso de firmas digitales) y de que la ubicación de almacenamiento sea segura (por ejemplo, no se pueden modificar las cadenas de conexión o sustituir los archivos de almacenamiento). El mecanismo de auditoría puede utilizar la firma de documentos, la autenticación de plataforma y la seguridad de acceso al código para asegurarse de que no haya código malintencionado que registre entradas falsas.

La interfaz de auditoría en la aplicación se puede exponer como una función de utilidad o como un método del objeto Principal de la aplicación si la acción auditada se debe correlacionar con el usuario.

Auditoría en componentes de procesos empresariales

Los procesos empresariales son objetivos prioritarios de la auditoría. Se deseará saber quién realizó las actividades empresariales clave y cuándo tuvieron lugar dichas actividades.

Al realizar la auditoría dentro del contexto de una transacción, a un administrador de recursos transaccional, se deseará que el componente de auditoría inicie una nueva transacción de modo que los errores en el árbol de la transacción original no deshagan también la entrada de auditoría.

Auditoría en componentes de acceso a datos

Los componentes de acceso a datos constituyen la capa de lógica empresarial personalizada más cercana al almacén de datos. Al igual que ocurría para la autorización minuciosa, la capa de los componentes de acceso a datos es una ubicación idónea para implementar una auditoría minuciosa.

Distribución de los componentes en las capas.

Los sistemas o arquitecturas en capas constituyen uno de los estilos que aparecen con mayor frecuencia. Se define como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Instrumentan así una vieja idea de organización estratigráfica que se remonta a las concepciones formuladas por Edsger Dijkstra en la década de 1960, largamente explotada en los años subsiguientes.

Ventajas en el caso de la arquitectura de capas:

- Reutilización de capas
- Facilita la estandarización
- Dependencias se limitan a intra-capa
- Contención de cambios a una o pocas capas

Generalmente en los sistemas se mezclan un conjunto de estilos de arquitectura. Lo más frecuente es encontrar el estilo MVC, capas, basada en objetos y servicios.

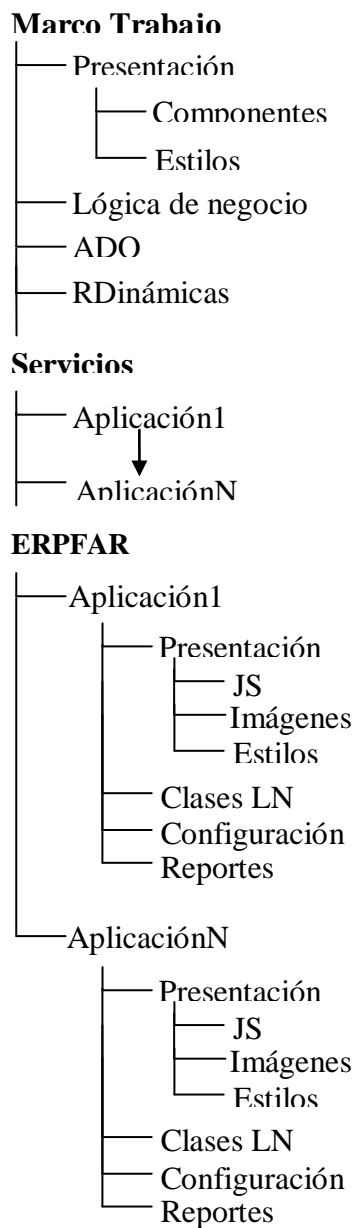
La arquitectura MVC es muy común en las aplicaciones web y tiene como objetivo fundamental separar el código cliente del servidor. Para el diseño de los componentes se utiliza una arquitectura basada en objetos donde se aplican fundamentalmente los patrones **Experto, Creador, Alta Cohesión, Bajo Acoplamiento y Controlador**.

La arquitectura basada en servicios surge como una solución para la comunicación interaplicaciones, aunque vale destacar que actualmente se está utilizando para realizar la comunicación intercapas.

El marco de trabajo se forma con los componentes de interfaz de usuario y estilo de aplicaciones, los componentes de lógica de negocio y acceso a datos principalmente.

Distribución física de los componentes.

En la presente sesión se pretende describir como distribuir físicamente los componentes del marco de trabajo y las aplicaciones físicamente. Los componentes del marco de trabajo son generales para todas las aplicaciones y los componentes de una aplicación pudieran o no ser utilizados en un momento determinado por otras aplicaciones los cual se realizará utilizando una interfaz de servicios.



1.7- Tecnologías utilizadas.

Como lenguaje de programación utilizamos PHP y como gestor de Base de Datos PostgreSQL. Utilizando para realizar la ingeniería de software la metodología formal RUP y como lenguaje de modelado UML.

Personal Home Page (PHP).

Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Es también un lenguaje interpretado y embebido en el HTML. Fue creado originalmente en 1994 por Rasmus Lerdorf, pero como PHP está desarrollado en política de código abierto, a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores.

El PHP corre en 7 plataformas, funciona en 11 tipos de servidores, ofrece soporte sobre unas 20 Bases de Datos y contiene unas 40 extensiones estables sin contar las que se están experimentando, además de que:

- Es software libre y abierto, lo que implica menos costes y servidores más baratos que otras alternativas.
- Su sintaxis está inspirada en C, ligeramente modificada para adaptarlo al entorno en el que trabaja, de modo que si se está familiarizado con esta sintaxis, resultará muy fácil aprender PHP.
- Su librería estándar es realmente amplia, lo que permite reducir los llamados "costes ocultos", uno de los principales defectos de ASP.
- PHP tiene una de las comunidades más grandes en Internet, por lo que no es complicado encontrar ayuda, documentación, artículos, noticias, y más recursos.
- Posee una potente variedad de extensiones para el acceso a la mayoría de los sistemas de gestión de bases de datos, por lo que una migración a otro sistema de gestión es mucho menos costosa que en otras plataformas.

PostgreSQL

PostgreSQL ofrece muchas ventajas respecto a otros sistemas de bases de datos:

Instalación ilimitada

Es frecuente que las bases de datos comerciales sean instaladas en más servidores de lo que permite la licencia. Algunos proveedores comerciales consideran a esto la principal fuente de incumplimiento de licencia. Con PostgreSQL, nadie puede demandarlo por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.

Esto tiene varias ventajas adicionales:

- Modelos de negocios más rentables con instalaciones a gran escala.
- No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.
- Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.
- Estabilidad y confiabilidad legendarias
- En contraste a muchos sistemas de bases de datos comerciales, es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Simplemente funciona.

Extensible

El código fuente está disponible para todos sin costo. Si su equipo necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin costos adicionales. Esto es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo que también extienden PostgreSQL todos los días.

Multiplataforma

PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows está actualmente en estado beta de pruebas.

Una lista breve de características técnicas que PostgreSQL ofrece:

- Diseñado para ambientes de alto volumen.
- Replicación (soluciones comerciales y no comerciales) que permiten la duplicación de bases de datos maestras en múltiples sitios de replica
- Interfaces nativas para ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python y Ruby
- Vistas
- Unicode
- Secuencias
- Procedimientos almacenados
- Soporte nativo SSL
- Lenguajes procedurales
- Respaldo en caliente
- Índices parciales y funcionales
- Autenticación Kerberos nativa
- Soporte para consultas con UNION, UNION ALL y EXCEPT
- Extensiones para SHA1, MD5, XML y otras funcionalidades
- Herramientas para generar SQL portable para compartir con otros sistemas compatibles con SQL
- Sistema de tipos de datos extensible para proveer tipos de datos definidos por el usuario, y rápido desarrollo de nuevos tipos
- Funciones de compatibilidad para ayudar en la transición desde otros sistemas menos compatibles con SQL.

Rational Unified Process (RUP):

El Rational Unified Process (RUP) es una metodología formal, a veces también llamada proceso, para desarrollo de software, documentada en hipertexto para ser consultada a través de un navegador.

El RUP describe a gran detalle todas las actividades, roles, responsabilidades, productos de trabajo y herramientas para definir quién hace qué y en qué momento en un proyecto de desarrollo de software.

El proceso ha sido elaborado en base a 6 ideales llamados principios clave que, a través del tiempo, han mostrado ser un conjunto de mejores prácticas para la industria del software convirtiéndose en estándares para desarrollo de software de aquí que el ciclo de vida de RUP, como se conoce al trazado de las actividades de desarrollo en el tiempo, está dividido en 4 fases: inicial, elaboración, construcción y transición, que corresponden a los 4 hitos principales de RUP: proyecto, arquitectura, versión β y release.

Con la utilización de RUP como metodología de desarrollo puede eliminarse una serie de problemas que surgen a la hora de crear un software como por ejemplo:

- Retrasos importantes en el tiempo de terminación.
- Gran número de errores aparecen cuando el software se pone en producción.
- Falta de tiempo para atender eficazmente proyectos de mantenimiento y nuevos desarrollos.
- Insatisfacción de los usuarios por el exceso de mantenimiento requerido o la falta de conformidad con los requerimientos.
- Pérdida de información entre proyectos por falta de documentación.
- Exceso de trabajo por no reutilizar material producido en proyectos anteriores.[5]

Lenguaje Unificado de Modelado (UML):

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

- Diagramas de Casos de Uso para modelar los procesos 'business'.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.

- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. Empezó como una consolidación del trabajo de Grade Booch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las metodologías orientadas a objetos más populares. [6]

UML será el lenguaje de modelado de software de uso universal pues existen razones para ello como:

- En el desarrollo han participado investigadores de reconocido prestigio.
- Ha sido apoyado por prácticamente todas las empresas importantes de informática.
- Se ha aceptado como un estándar por la OMG.
- Prácticamente todas las herramientas CASE y de desarrollo la han adaptado como lenguaje de modelado.

UML resuelve de forma bastante satisfactoria un viejo problema del desarrollo de software como es su modelado gráfico. Además, se ha llegado a una solución unificada basada en lo mejor que había hasta el momento, lo cual lo hace todavía más excepcional.

1.8- Herramientas utilizadas.

Visual Paradigm para UML

Esta herramienta multiplataforma es un modelador UML y herramienta CASE para programadores. Se escogió el Visual Paradigm porque es una herramienta CASE que ofrece un entorno de creación de diagramas para UML; diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad; uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación; capacidades de ingeniería directa (versión profesional) e inversa; modelo y código que permanece sincronizado en todo el ciclo de desarrollo; disponibilidad de múltiples versiones, para cada necesidad; disponibilidad de integrarse en los principales IDEs; disponibilidad en múltiples plataformas.

Proporciona modelar procesos de negocio así como un generador de esqueletos de clases para Java, .NET y PHP.

Dreamweaver 8

Dreamweaver 8 es un software fácil de usar que permite crear páginas web profesionales. Las funciones de edición visual de Dreamweaver 8 permiten agregar rápidamente diseño y funcionalidad a las páginas, sin la necesidad de programar manualmente el código HTML. Se puede crear tablas, editar marcos, trabajar con capas, insertar comportamientos JavaScript, etc., de una forma muy sencilla y visual.

Ventajas que proporciona con respecto a las versiones anteriores:

- **Interfaz mejorada:** Los usuarios con problemas visuales podrán acceder a una opción de Aumento de la pantalla en vista de diseño para analizar o trabajar con difíciles anidamientos de tablas. Además de la inclusión de información visual gracias a las guías que permitirán la medición píxel a píxel de todos los elementos.
- **Nueva barra de herramientas:** Se ha añadido una barra de herramientas a Dreamweaver 8, podrás encontrarla en la parte lateral izquierda del modo de Código, esta barra hace mucho más accesible el código al permitirnos la navegación por etiquetas y su contracción. Una de las nuevas novedades es la posibilidad de añadir comentarios con un sólo clic.
- **Compatibilidad:** La compatibilidad añadida en esta versión con PHP5, Coldfusion MX 7 y Video Flash.
- **Mejoras CSS:** esta última versión ha mejorado mucho respecto a la compatibilidad y manejo de estilos de cascada. De esta forma se ha mejorado el panel de estilos CSS, donde ahora podrás acceder a la configuración de cada uno de los estilos desde una lista mucho mejor dotado de una cuadrícula editable desde donde podrás modificar sus propiedades. Además, Dreamweaver 8, añade una nueva barra de herramientas que proporciona la reproducción inmediata de los estilos para diferentes medios (pantalla, impresora...).
- **Del lado del XML,** incluye interesantes herramientas visuales para incluir contenidos de este formato, integrándolos fácilmente en sitios web y aplicaciones.[7]

Conclusiones:

En este capítulo se detallaron las condiciones y problemas que rodean el objeto de estudio; y a través de los conceptos y definiciones planteadas. Se analizó profundamente el problema y sus variantes en el mundo, se realizó un análisis completo de las tecnologías que serán utilizadas a lo largo del desarrollo del sistema propuesto, y se fundamentaron las elecciones del lenguaje, el sistema gestor de bases de datos, y la metodología a utilizar. Una vez conocidas las herramientas optimas y los conceptos a utilizar se puede empezar a desarrollar la propuesta de sistema.

CAPITULO II: CARACTERÍSTICAS DEL SISTEMA.

Introducción:

En el presente capítulo se hace un análisis crítico del funcionamiento del negocio, así como su representación gráfica y su descripción. Se modela el Sistema y se plantean todos los conceptos relacionados con el mismo. Nos representa además los actores que interactúan con el sistema, así como su función en el mismo y enumera los requisitos funcionales y no funcionales que debe tener la aplicación, lo que nos da un primer acercamiento al sistema.

2.1 Análisis crítico de los procesos actuales:

Los procesos actualmente se ejecutan de una forma correcta, donde el Jefe de Proyecto o el Ingeniero de componente al frente de cada proyecto le entregan a cada programador el trabajo que debe hacer con su diagrama de caso de uso junto a una descripción detallada del mismo, así como el diagrama de clases y el diagrama de interacción. Cuando el programador termina le entrega la tarea asignada al Jefe de Proyecto o al Ingeniero de componente que esta vinculado a él y este la revisa, probando su calidad, funcionalidad y nivel de acabado.

El problema de todo esto consiste en que cuando el programador esta ejecutando la tarea asignada, pierde mucho tiempo a la hora de programarlo pues debe repetir en mucho de los casos las mismas líneas de código lo que variando detalles como nombre de la clase, consulta, campos y otra serie de parámetros que es la minoría del código, a veces estas pequeñas variaciones trae consigo que se cometan errores y se pierda tiempo también corrigiéndolos y encontrando donde esta el problema. Además a veces la arquitectura de la Aplicación es violada, al programador no tener preparación o habilidad a la hora de programar lo que se le asigne.

2.2 Objeto de automatización:

Muchos de estos procesos que realiza el programador pueden automatizarse como son generar formularios, crear interfaz, programar acceso a datos, programar lógica de negocio. Los cuales se encargarán de solo pedirle unos parámetros al programador y el software se encargara de generar el código, ahorrándole tiempo de programación.

2.3 Descripción general de la propuesta de sistema:

El sistema se encargará de toda la generación de código posible de un proyecto programado en PHP y siguiendo una arquitectura base robusta y eficiente. Esta aplicación funcionará de forma tal que permita a un programador generar código para el módulo en el que esta trabajando o asignado.

El programador al autenticarse en la aplicación, esta cargará un menú acorde a sus privilegios con las funcionalidades que este debe cumplir, y ajustándose al modulo en el cual el programador esta trabajando.

Después del programador haberse autenticado tiene derecho a Crear un Portal, a definir los estilos del Portal, Crear un Formulario y a crear todas las clases de acceso a datos.

Al Crear Portal el programador escogerá el banner que desea ponerle al Portal al igual que el menú que tendrá la aplicación. Después de esto deberá definir los estilos del Portal como son color así como imágenes de insertar, modificar, eliminar, aceptar y cancelar.

Al Crear un Formulario el programador deberá seleccionar los campos que estarán asociados al formulario y los componentes que estarán asociados a cada campo. Después de esto se generara un formulario el cual se puede guardar XML o en HTML.

Al generar el formulario se podrá generar también un XMI el cual no es mas que una especie de código XML que entienden las herramientas CASE como Visual Paradigm y Rational. Con esta propuesta de sistema, no solo se beneficia el programador, sino también el analista, pues se crearán diagramas de Ingeniería para facilitar el trabajo.

2.4 Modelo del Negocio:

Para conseguir sus objetivos, una empresa organiza su actividad por medio de un conjunto de procesos de negocio. Cada uno de ellos se caracteriza por una colección de datos que son producidos y manipulados mediante un conjunto de tareas, en las que ciertos agentes (por ejemplo trabajadores) participan de acuerdo a un flujo de trabajo determinado. Además, estos procesos se hallan sujetos a un

conjunto de reglas de negocio, que determinan la estructura de la información y las políticas de la empresa. Por tanto, la finalidad del modelado del negocio es describir cada proceso del negocio, especificando sus datos, actividades (o tareas), roles (o agentes) y reglas de negocio. La producción de software no es más que una gran empresa donde se cumplen todas estas reglas de negocio.

Los objetivos del modelamiento del negocio son:

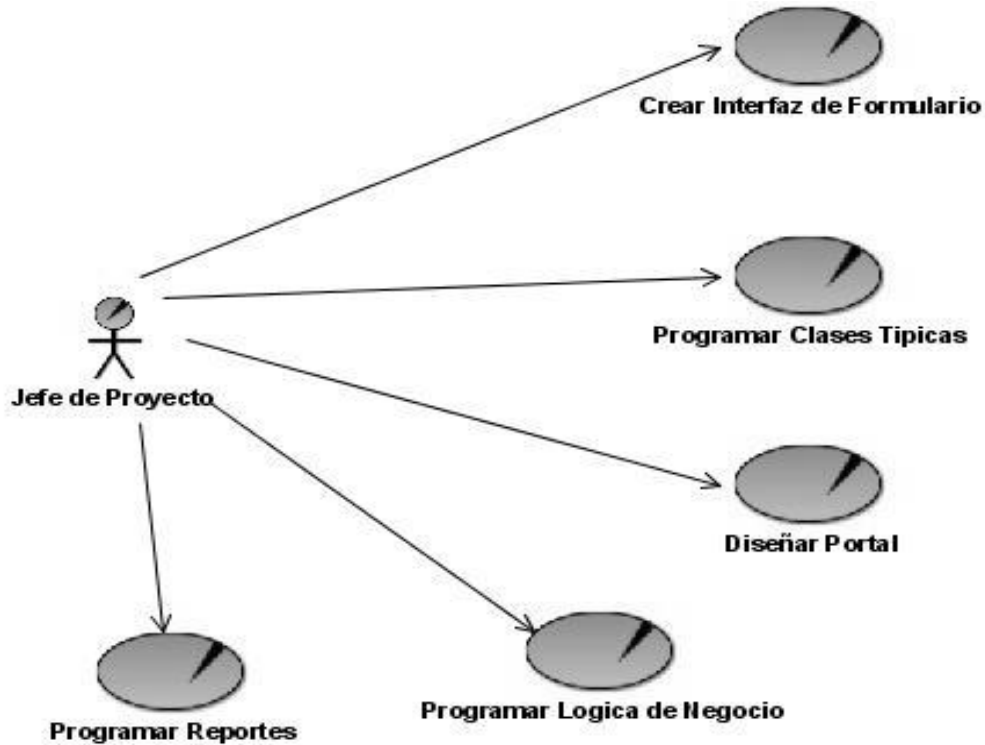
- Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema.
- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.
- Derivar los requerimientos del sistema que va a soportar la organización.

Una vez se han identificado los procesos de negocio, es preciso encontrar los agentes involucrados en su realización. Cada uno de estos agentes o actores del negocio desempeña cierto papel (juega un rol) cuando colabora con otros para llevar a cabo las actividades que conforman dicho caso de uso del negocio. De hecho, identificaremos los roles que son jugados por agentes de la propia empresa (que incluyen trabajadores y dispositivos físicos) o agentes externos (como clientes u otros sistemas).

Actores del negocio	Justificación
Jefe de proyecto	Distribuye y organiza que debe hacer cada programador. Es el encargado de decirle a cada persona de su modulo que va a programar y como debe hacerlo.

Trabajadores del negocio	Justificación
Programador	Se encarga de programar todas las tareas que le asigna el Especialista de su Proyecto.

Diagrama de casos de uso del negocio



2.5 Descripción del caso de uso del negocio.

Nombre del CU del Negocio:	Crear Interfaz de Formulario
Actores del negocio:	Jefe de Proyecto.
Trabajadores del negocio:	Programador.
Resumen: Se inicia cuando el jefe de proyecto le entrega a un programador una interfaz para que este la programe, incluyendo una serie de datos como los componentes que debe usar y el nombre de la interfaz. El programador la programa y se la entrega al especialista.	
Acción del actor	Respuesta del negocio
1-El jefe de proyecto le entrega una descripción de Interfaz al programador para que este la programe.	2- El programador recibe la descripción de la Interfaz.
	3- Programa la Interfaz

	4- Le entrega la Interfaz programada al jefe de proyecto para su posterior uso.
5-El jefe de proyecto revisa la Interfaz.	
6- El jefe de proyecto integra la interfaz al proyecto.	
Prioridad:	-
Mejoras:	Al automatizar este proceso se podrá realizar la Creación de Interfaz de forma automática, en donde el programador solo tiene que especificarles los campos que debe tener y los componentes que va a estar asociado a cada campo.
Cursos alternos: Paso 3: Si el programador tiene dudas, acude al jefe de proyecto para que este le aclare su duda y vuelve al paso 3. Paso 5: Si el jefe de proyecto detecta algún error, vuelve al paso 3.	

Nombre del CU del Negocio:	Programar Clases Típicas.
Actores del negocio:	Jefe de Proyecto
Trabajadores del negocio:	Programador.
Resumen: Se inicia cuando el jefe de proyecto le entrega a un programador una serie de tablas de la Base de Datos para que este programe sus clases Típicas.	
Acción del actor	Respuesta del negocio
1-El jefe de proyecto le entrega las descripciones de las clases Típicas al programador para que este programe.	2- El programador recibe las descripciones de las clases Típicas.
	3- Programa las clases Típicas.
	4- Le entrega las clases Típicas programada al jefe de proyecto para su posterior uso.

CAPITULO II: CARACTERISTICAS DEL SISTEMA.

5-El jefe de proyecto revisa las clases Típicas.	
6- El jefe de proyecto integra las clases Típicas al proyecto.	
Prioridad:	-
Mejoras:	Al automatizar este proceso se podrá realizar la programación de clases Típicas de forma automática, en donde el programador solo tiene que especificarle las tablas de la Base de Datos.
Cursos alternos:	
Paso 3: Si el programador tiene dudas, acude al jefe de proyecto para que este le aclare su duda y vuelve al paso 3.	
Paso 5: Si el jefe de proyecto detecta algún error, vuelve al paso 3.	

Nombre del CU del Negocio:	Diseñar Portal.
Actores del negocio:	Jefe de Proyecto.
Trabajadores del negocio:	Programador.
Resumen: Se inicia cuando el jefe de proyecto le entrega a un programador la tarea de diseñar el Portal y este se encarga de hacer el diseño y programarlo.	
Acción del actor	Respuesta del negocio
1-El jefe de proyecto le entrega la descripción del Portal del Proyecto al programador.	2- El programador recibe la descripción del Portal.
	3- Diseña y programa el Portal.
	4- Le entrega el Portal al jefe de proyecto programado.
5-El jefe de proyecto revisa el diseño del Portal.	
6- El jefe de proyecto integra el Portal al proyecto.	

CAPITULO II: CARACTERISTICAS DEL SISTEMA.

Prioridad:	-
Mejoras:	Al automatizar este proceso se podrá realizar la programación del Portal de forma automática, en donde el programador solo tiene que especificar el Banner y el menú de la aplicación.
Cursos alternos:	
Paso 3: Si el programador tiene dudas, acude al jefe de proyecto para que este le aclare su duda y vuelve al paso 3.	
Paso 5: Si el jefe de proyecto detecta algún error, vuelve al paso 3.	

Nombre del CU del Negocio:	Programar Lógica de Negocio
Actores del negocio:	Jefe de Proyecto
Trabajadores del negocio:	Programador.
Resumen: Se inicia cuando el jefe de proyecto le entrega a un programador la tarea de programar una Lógica de Negocio y este se encarga de programarlo.	
Acción del actor	Respuesta del negocio
1-El jefe de proyecto le entrega la descripción de una Lógica de Negocio del Proyecto al programador.	2- El programador recibe la descripción de la Lógica de Negocio.
	3- Programa la Lógica de Negocio.
	4- Le entrega la Lógica de Negocio al jefe de proyecto programada.
5-El jefe de proyecto revisa la Lógica de Negocio.	
6- El jefe de proyecto integra la Lógica de Negocio al proyecto.	
Prioridad:	-
Mejoras:	Al automatizar este proceso se podrá realizar la programación de la Lógica de Negocio de forma automática, en donde

	el programador solo tiene que especificar las clases típicas que se vana relacionar y la relación entre estas.
Cursos alternos:	
Paso 3: Si el programador tiene dudas, acude al jefe de proyecto para que este le aclare su duda y vuelve al paso 3.	
Paso 5: Si el jefe de proyecto detecta algún error, vuelve al paso 3.	

Nombre del CU del Negocio:	Programar Reportes
Actores del negocio:	Jefe de Proyecto
Trabajadores del negocio:	Programador.
Resumen: Se inicia cuando el jefe de proyecto le entrega a un programador la tarea de programar un Reporte y este se encarga de programarlo.	
Acción del actor	Respuesta del negocio
1-El jefe de proyecto le entrega la descripción de un Reporte del Proyecto al programador.	2- El programador recibe la descripción del Reporte.
	3- Programa el Reporte.
	4- Le entrega el Reporte al jefe de proyecto programado.
5-El jefe de proyecto revisa el Reporte.	
6- El jefe de proyecto integra el Reporte al proyecto.	
Prioridad:	
Mejoras:	Al automatizar este proceso se podrá realizar la programación de los Reportes de forma automática, en donde el

	programador solo tiene que algunos datos de la consulta que lleva el reporte.
<p>Cursos alternos: Paso 3: Si el programador tiene dudas, acude al jefe de proyecto para que este le aclare su duda y vuelve al paso 3. Paso 5: Si el jefe de proyecto detecta algún error, vuelve al paso 3.</p>	

2.6 Diagramas de Actividades del Negocio:

Diagrama Actividad de programar interfaz:

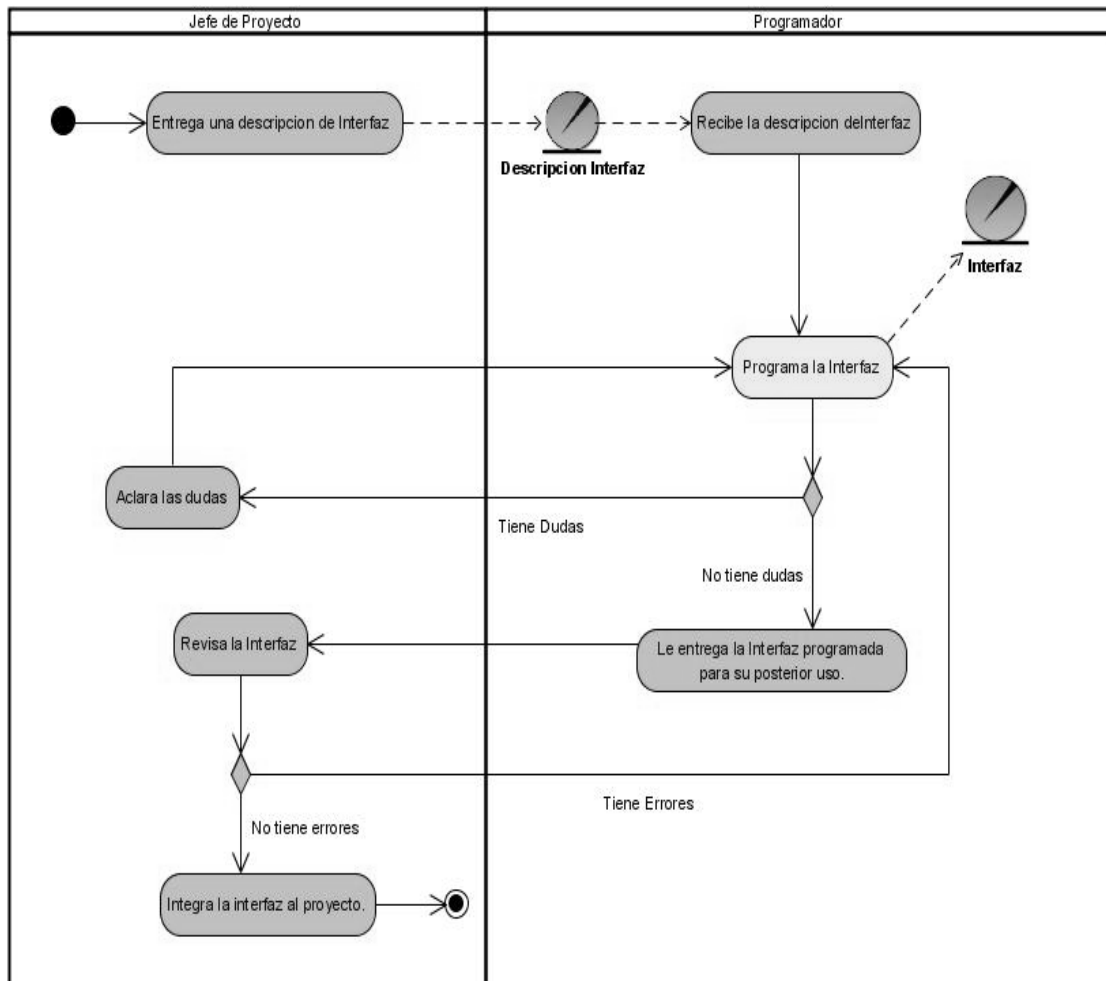


Diagrama Actividad de Programar Clases Típicas.

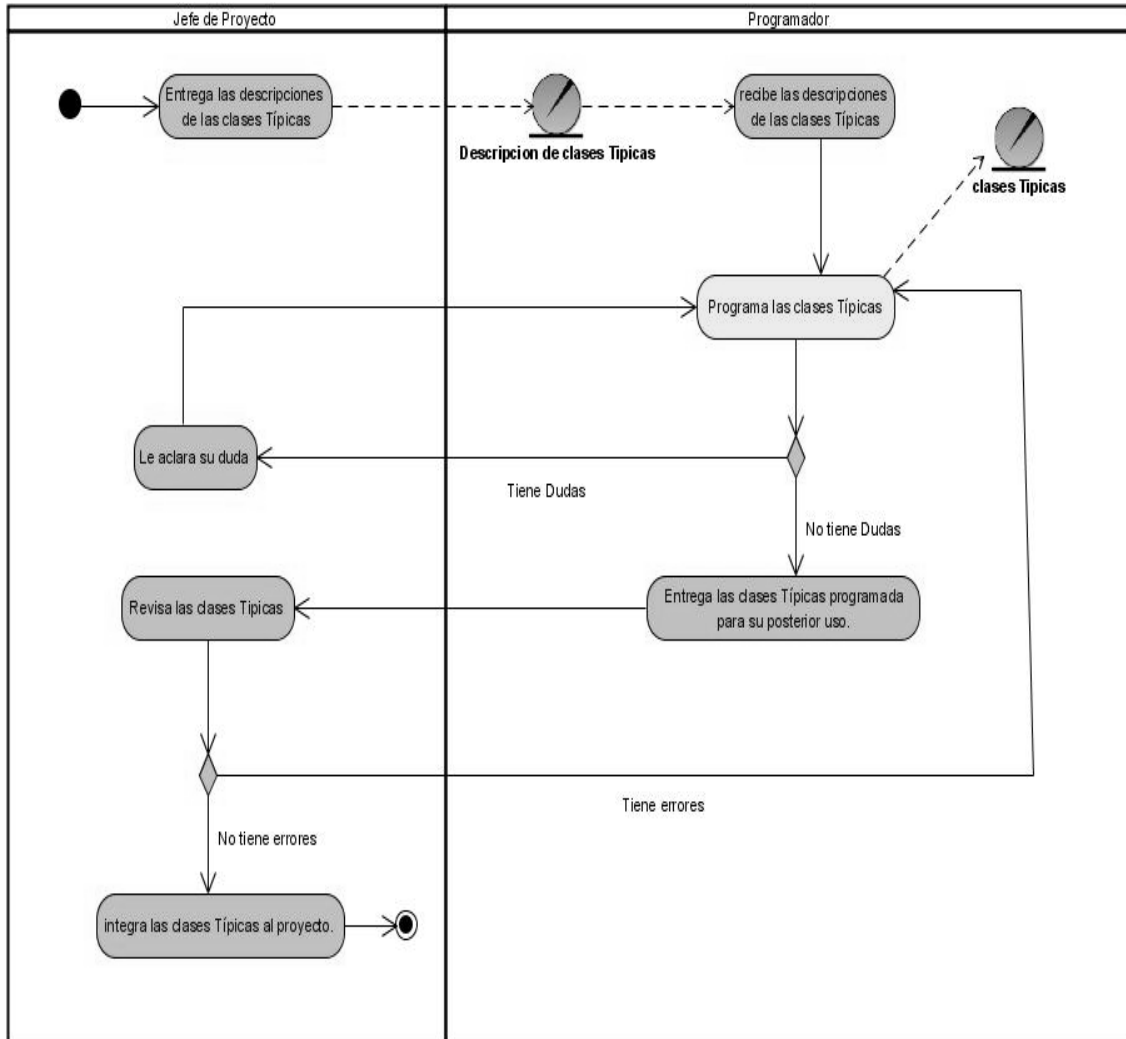


Diagrama Actividad de Diseñar Portal.

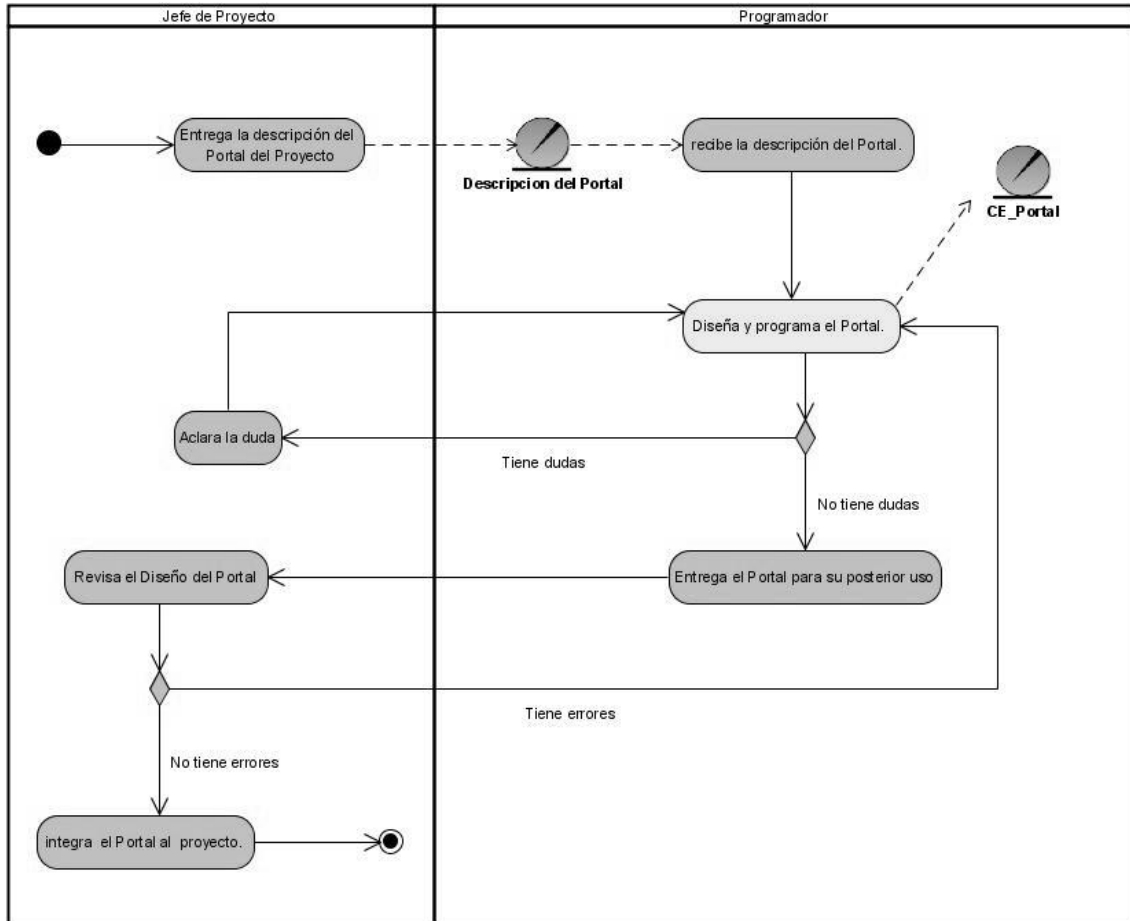
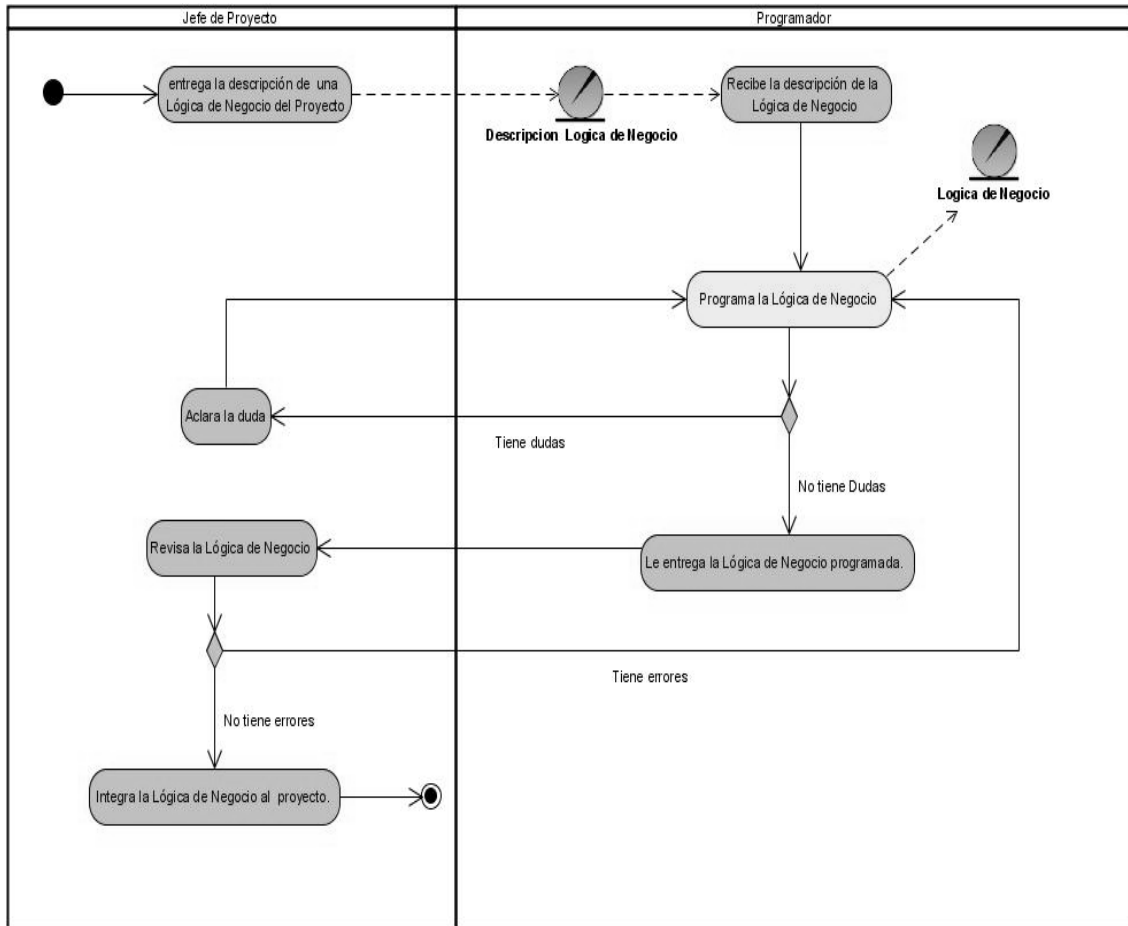
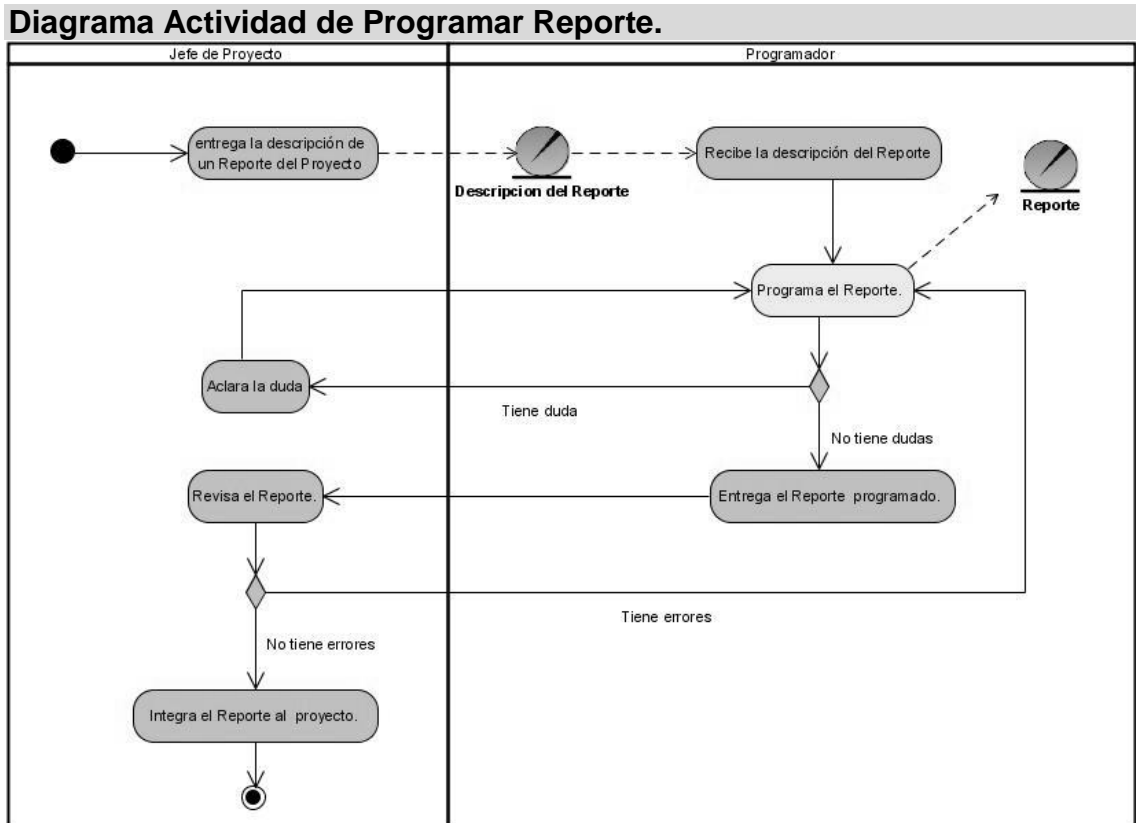


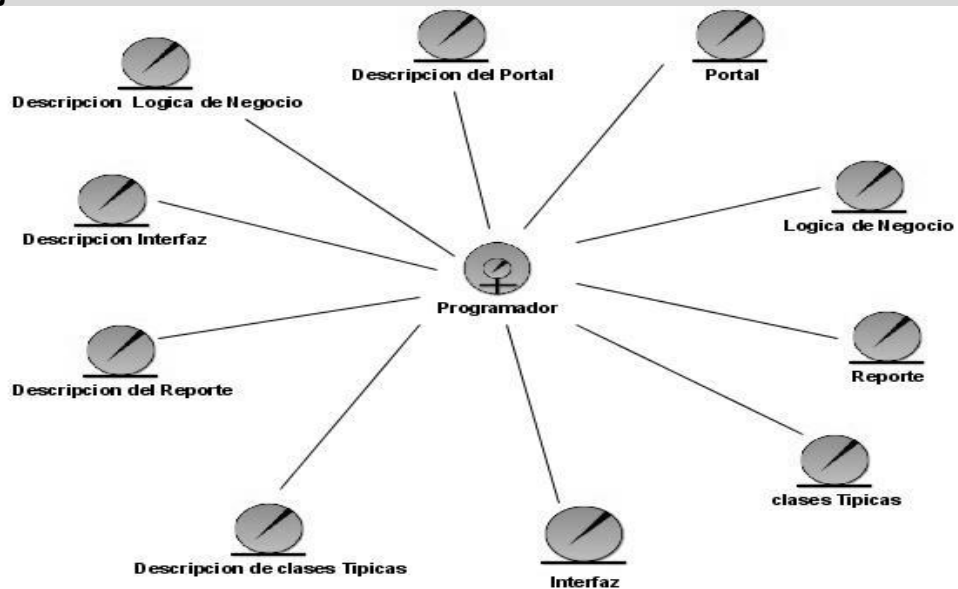
Diagrama Actividad de Programar Lógica de Negocio





2.7 Modelo de Objeto:

Modelo de Objeto



2.8 Requerimientos Funcionales:

RF1: Generar Portal.

- RF1.1 Mostrar Portal.

RF2: Generar Proyecto.

- RF2.1 Crear Menú XML.
- RF2.2 Cargar Menú XML.

RF3: Generar Estilos.

- RF3.1 Seleccionar Imágenes Toolbar.
- RF3.2 Seleccionar Imágenes de los Botones.

RF4: Generar Formulario.

- RF4.1 Seleccionar tablas y campos para el Formulario.

RF5: Generar Formulario XML.

RF6: Generar XMI.

RF7: Generar Típicas.

- RF7.1 Seleccionar tablas y campos para generar las clases típicas.

RF8: Generar Conexión.

RF9: Modificar Arquitectura.

RF10: Agregar Componentes.

2.9 Requerimientos no funcionales:

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Apariencia o interfaz externa:

- Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.
- Empleo de los colores: gris, blanco y azul principalmente, que son los definidos en los estándares del proyecto.

Usabilidad:

- El software tendrá siempre la posibilidad de ayuda disponible para cualquier tipo de usuario, lo que le permitirá un avance considerable en la explotación de la aplicación en todas sus funcionalidades.

Rendimiento:

- Tiempos de respuestas rápidos al igual que la velocidad de procesamiento de la información, no mayor a los 5 segundos en las actualizaciones y no mayor de 20 para las recuperaciones.

Soporte:

- Se requiere un servidor de bases de datos con las siguientes características:
 - Soporte para grandes volúmenes de datos y velocidad de procesamiento.
 - Tiempo de respuesta rápido en accesos concurrentes.
- Versión de PHP 5.0.
- Por parte del cliente se requiere un navegador capaz de interpretar JavaScript.

Portabilidad:

- Necesidad de que el sistema sea multiplataforma.

Seguridad:

- Autenticación (contraseña de acceso)
- Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de usuario que este activo.
- Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- Verificación sobre acciones irreversibles (eliminaciones).

Confiableidad:

- La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.

Funcionalidad:

- Mínima cantidad de páginas para ejecutar todas las funciones posibles (preferentemente que estén relacionadas).

Implantación

- Entregar toda la documentación asociada al proyecto.
- Organizar el adiestramiento de los usuarios.

Software:

En secciones anteriores se ha mencionado que la construcción de nuestra aplicación funcionará bajo los conceptos de arquitectura cliente/servidor. Por tanto el servidor del usuario final debe tener como requerimientos mínimos de software:

- Una computadora personal con plataforma del sistema operativo Windows Advancer Server 2000 o superior; o Linux.
- Apache 2.0 o superior como servidor Web, con módulo PHP 5 disponible y debe estar configurado con la extensión pgsql incluida.
- PostgreSQL como Sistema Gestor de Base de Datos.

Y la máquina cliente del usuario debe tener como requerimiento mínimo:

- El navegador Mozilla FireFox.

Hardware

Partiendo del mismo supuesto que los requerimientos de software, nuestro modelo ideal (cliente/servidor), para los requerimientos mínimos de hardware, el usuario final debe tener un servidor con las siguientes características:

- Tarjeta de red.
- 128 Mb. de RAM o superior.
- 40 Gb. de disco duro o superior.
- Pentium II a 133 MHz de velocidad en su procesador o más.

Una computadora que sirva de cliente:

- Pentium a 200 MHz. de velocidad de procesamiento o superior.
- 32 Mb. de memoria RAM superior.
- Tarjeta de red.

2.10 Casos de uso y actores del sistema:

2.10.1 Definición de los actores.

Actores	Justificación
Programador	Es el programador que trabaja en cada modulo y que recibe instrucciones del especialista del proyecto para ejecutar una tarea. Utiliza la arquitectura que ya esta montada para realizar casos de usos de su proyecto
Integrador de aplicación	Es el programador que tienes fuertes conocimientos de programación y un alto dominio de la Arquitectura. Además es el encargado de integrar la aplicación y el trabajo de los demás programadores.

2.10.2 Listado de casos de uso.

CU1	Generar Portal.
Actor	Programador
Descripción	Para Generar el Portal se le pide como datos al programador el Título de la Aplicación y el Banner que va a usar, así como el menú que desea que tenga la aplicación.
Referencia	RF1

CU2	Generar Estilos.
Actor	Programador
Descripción	Después de haber generado el Portal se generan los estilos que va a tener la aplicación. Para generar los Estilos se le piden una serie de parámetros al programador como son color de fondo e imágenes.
Referencia	RF2

CU3	Generar Formulario.
Actor	Programador
Descripción	Para Generar el Formulario se le pide como datos al programador los campos de la Base de Datos y los componentes asociados a cada campo que se van a usar, así como el patrón que va a seguir.
Referencia	RF3

CU4	Generar Formulario XML.
Actor	Programador
Descripción	Para Generar el Formulario se puede generar un XML con todos los datos de ese formulario generado, para un posterior uso.
Referencia	RF4

CU5	Generar Formulario XMI.
Actor	Programador
Descripción	Para Generar el Formulario se puede generar un XMI con todos los datos de ese formulario generado, para un posterior uso en herramientas CASE como Visual Paradimg.
Referencia	RF5

CU6	Generar Típicas.
Actor	Programador
Descripción	Para Generar las clases Típicas se le pide como datos al programador la Base de Datos y a partir de ahí se generan todas las clases asociadas a cada tabla de la Base de Datos.
Referencia	RF6

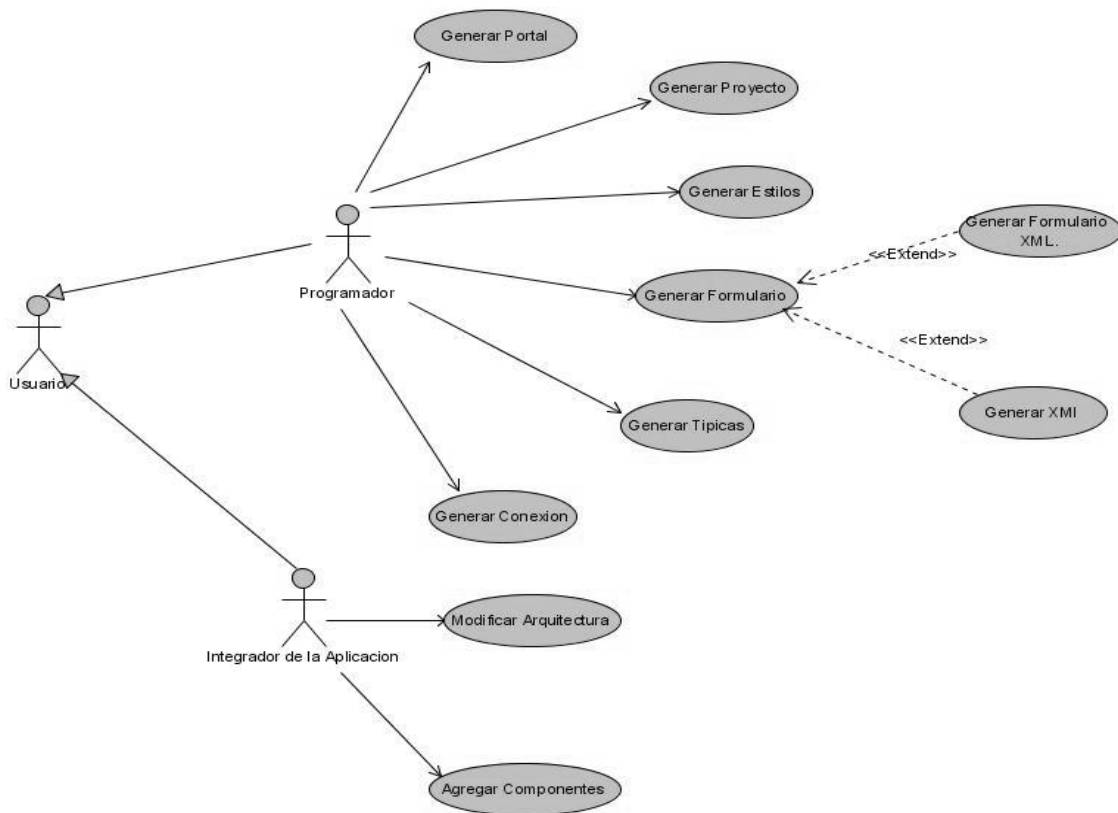
CU7	Generar Conexión.
Actor	Programador
Descripción	Para Generar la Conexión de la Aplicación a la Base de datos se le pide una serie de datos al programador como host, usuario, contraseña y Base de datos a donde el desea que se conecte la aplicación.
Referencia	RF7

CU8	Modificar Arquitectura.
Actor	Integrador de aplicación
Descripción	La aplicación generadora de código, genera el código para una arquitectura específica, la cual va a poder ser modificada por un programador avanzado para especificar su propia arquitectura.
Referencia	RF8

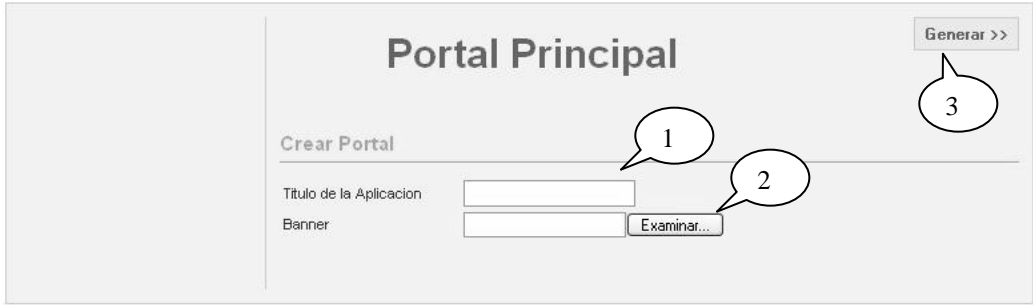
CU9	Agregar Componentes.
Actor	Integrador de aplicación
Descripción	La aplicación generadora tendrá una Base de Datos d componentes que pueden ser utilizados en generar código. A este Base de Datos se le podrá introducir nuevos componentes para su posterior utilización.
Referencia	RF9

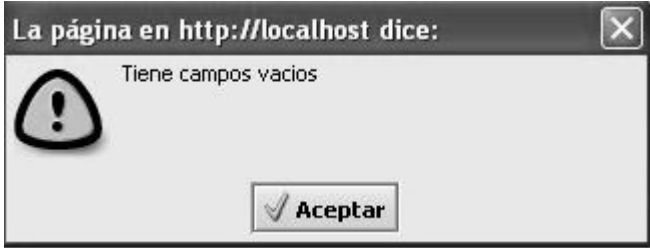
2.11 Diagrama de casos de uso:

DIAGRAMA DE CASOS DE USO



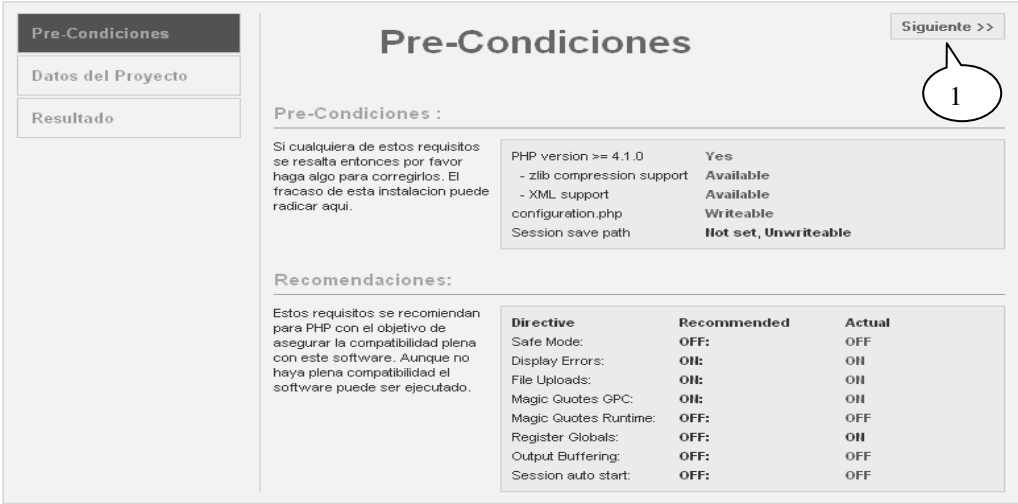
2.12 Descripción de los Casos de uso expandida:

Caso de uso:	Generar Portal
Actores:	Programador (inicia).
Propósito:	Generar un Portal de forma dinámica
Resumen:	El caso de uso se inicia cuando el programador necesita crear un portal y le introduce una serie de datos al sistema para que este se lo genere de forma dinámica.
Precondiciones:	Se debe seleccionar primero el proyecto con el cual se va a trabajar.
Poscondiciones:	-
Tipo:	Real y expandido.
Responsabilidades:	RF1
Casos de uso relacionados:	-
Interfaz I	
	
<p>(1) Titulo que se le va a dar al portal generado. (2) Dirección de la imagen del banner que se desea poner en el portal.</p>	
Curso normal de eventos para el caso de uso.	
Acción del actor	Respuesta del sistema
1. El actor selecciona la opción Nuevo Portal del menú principal (flujo básico).	2. El sistema muestra una serie de datos que el usuario debe llenar para generar el Portal, dentro de los que se encuentra: Titulo de la Aplicación y Banner.
3. Llena el campo del Titulo de la Aplicación	
4. Busca donde se encuentra la imagen del Banner a través del navegador tiene la aplicación.	

5. Presiona el Botón Generar.	6. Con los datos introducidos genera el Portal
	7. Muestra como quedo el Portal.
Cursos alternos	
<p>Línea 5: Si el actor presiona Generar sin tener los datos llenos se envía el siguiente mensaje:</p> 	

Caso de uso:	Generar Proyecto
Actores:	Programador (inicia).
Propósito:	Generar un Proyecto de forma dinámica
Resumen	El caso de uso se inicia cuando el programador necesita crear un proyecto y le introduce una serie de datos al sistema para que este se lo genere de forma dinámica.
Precondiciones:	-
Poscondiciones:	-
Tipo:	Real y expandido.
Responsabilidades:	RF2
Casos de uso relacionados:	-

Interfaz I




UCI. © 2006 - 2007 All rights reserved.
 Generador es un software libre bajo licencia GNU/GPL.



(2) Nombre que se le va a dar al proyecto generado.


(3) Dirección del XML donde esta el menú que se desea poner en el proyecto, partiendo de la opción que los proyectos al ser creados no tienen seguridad integrada.



Curso normal de eventos para el caso de uso.	
Acción del actor	Respuesta del sistema
1. El actor selecciona la opción Nuevo Proyecto del menú principal (flujo básico).	2. El sistema muestra la Interfaz I que son las condiciones que debe tener el servidor en que se va a montar el proyecto.
3. Presiona el Botón Siguiente (1).	4. El sistema muestra la Interfaz II con los datos que deben ser introducidos para confeccionar el Proyecto, los cuales son: Proyecto (2) y menú (3).
5. Llena el campo de Proyecto.	
6. Busca donde se encuentra el menú del proyecto a través del navegador tiene la aplicación.	
7. Presiona el Botón Generar. (3)	8. Con los datos introducidos genera el Portal
	7. Muestra la Interfaz III al ser satisfactoria la creación del proyecto.
Cursos alternos	
Línea 7: Si el actor presiona Generar sin tener los datos llenos se envía el siguiente mensaje:	
	

Caso de uso:	Generar Estilos
Actores:	Programador (inicia).
Propósito:	Generar los estilos que tendrá una aplicación de forma dinámica
Resumen	El caso de uso se inicia cuando el programador necesita crear los estilos de su proyecto y le introduce una serie de datos al sistema para que este se lo genere de forma dinámica.
Precondiciones:	Se debe seleccionar primero el proyecto con el cual se va a trabajar.
Poscondiciones:	-
Tipo:	Real y expandido.
Responsabilidades:	RF3
Casos de uso relacionados:	-

Interfaz I



UCI. © 2006 - 2007 All rights reserved.
Generador es un software libre bajo licencia GNU/GPL.

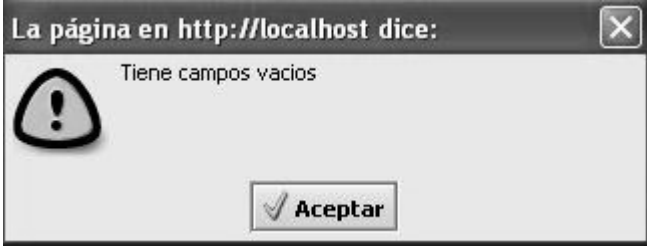
(1) Numero Hexadecimal del color que se le desea dar de fondo a la interfaz generada.
(2)) Número Hexadecimal del color que se le desea dar al marco de la interfaz generada.

Curso normal de eventos para el caso de uso.

Acción del actor	Respuesta del sistema
1. El actor selecciona la opción Nuevo Estilo del menú principal (flujo básico).	2. El sistema muestra una serie de datos que el usuario debe llenar para generar los estilos, dentro de los que se encuentra: Color de Fondo y Color del Marco.
3. Llena el campo del Color de Fondo	
4. Llena el campo del Color del Marco	
5. Presiona el Botón Generar (3).	6. Con los datos introducidos genera los estilos

Cursos alternos

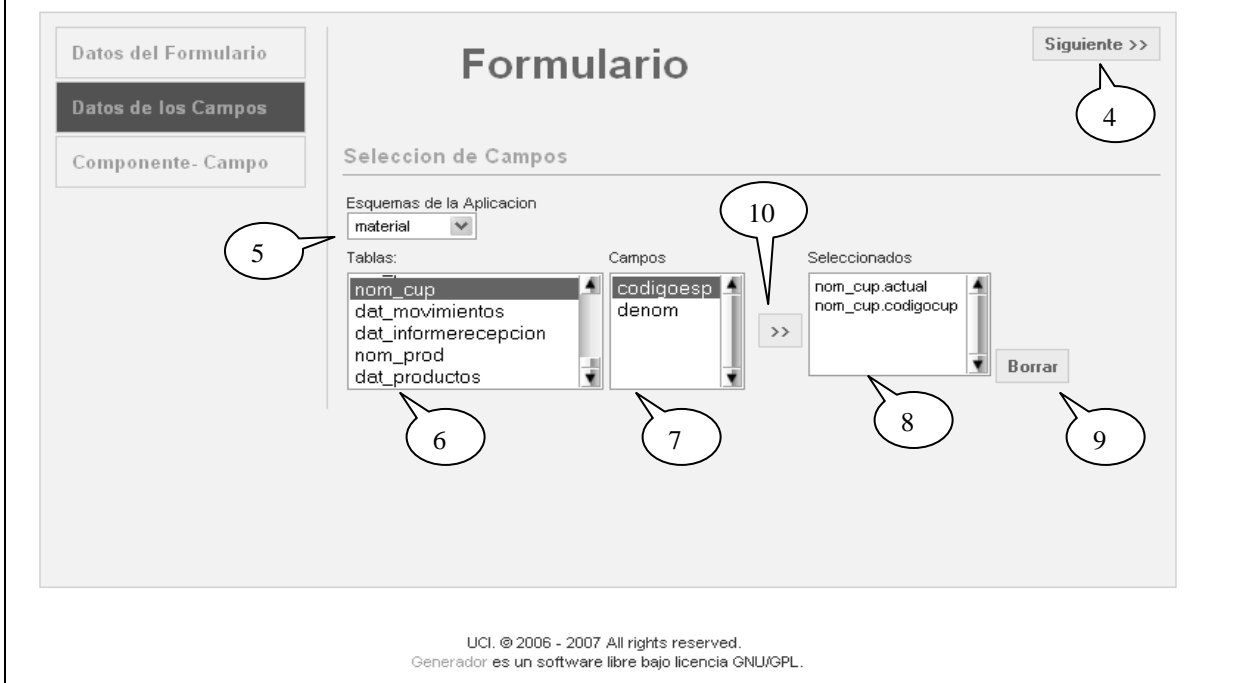
Línea 5: Si el actor presiona Generar sin tener los datos llenos se envía el siguiente mensaje:



Caso de uso:	Generar Formulario
Actores:	Programador (inicia).
Propósito:	Generar un Proyecto de forma dinámica
Resumen	El caso de uso se inicia cuando el programador necesita crear un formulario y le introduce una serie de datos al sistema para que este se lo genere de forma dinámica.
Precondiciones:	Se debe seleccionar primero el proyecto con el cual se va a trabajar.

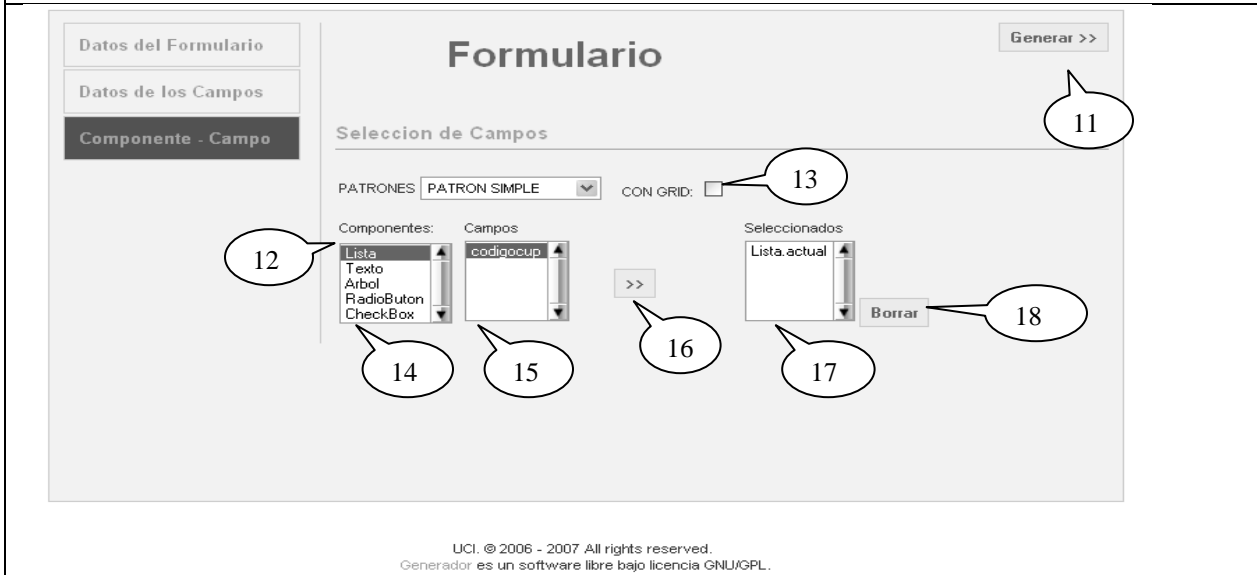
Poscondiciones:	-
Tipo:	Real y expandido.
Responsabilidades:	RF4
Casos de uso relacionados:	-
Interfaz I	
<p>UCL © 2006 - 2007 All rights reserved. Generador es un software libre bajo licencia GNU/GPL.</p>	
<p>(2) Nombre que va a recibir el formulario cuando se genere. (3) Titulo que va a recibir el formulario cuando se genere, es decir, encabezado del formulario.</p>	

Interfaz II



- (5) Nombre de los esquemas de la Base de Datos con la que se esta trabajando.
- (6) Tablas del esquema que se selecciono.
- (7) Campos de la tabla que se selecciono.
- (8) Campos seleccionados para conformar el formulario de la tabla que se selecciono.

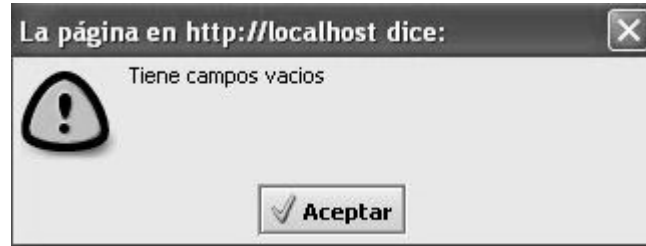
Interfaz III



<p>(12) Tipos de formularios que pueden ser generados, se denominan como patrones de Interfaz.</p> <p>(13) Si se desea colocar un gris en el formulario se debe marcar.</p> <p>(14) Componentes que pueden ser introducidos en el formulario.</p> <p>(15) Campos que fueron seleccionados en la Interfaz I y con los cuales se va a conformar el formulario.</p> <p>(17) Relación que va a existir entre los campos y los componentes, es decir, que componente va a estar asociado a cada campo.</p>	
Curso normal de eventos para el caso de uso.	
Acción del actor	Respuesta del sistema
1. El actor selecciona la opción Nuevo Formulario del menú principal (flujo básico).	2. El sistema muestra la Interfaz I que son los datos del formulario que se desea crear.
3. Llena el campo del nombre que va a tener el formulario (2).	
4. Llena el campo del título que va a tener el formulario (3).	
5. Presiona el Botón Siguiente (1).	6. El sistema muestra la Interfaz II con los datos que deben ser introducidos para confeccionar el formulario, los datos están compuestos por los campos que van a conformar el formulario.
7. El Actor selecciona el esquema del cual va a generar el formulario (5).	8. El sistema actualiza las tablas que pertenecen a ese esquema seleccionado (6).
9. Selecciona la tabla de la cual va a generar el formulario (6).	10. El sistema actualiza los campos que pertenecen a esa tabla seleccionada (7).
11. Selecciona los campos con los cuales se va a generar el formulario (7) utilizando un componente que se encarga de colocar los campos seleccionados en una nueva lista de campos seleccionados (8), auxiliándose del botón ">>"(10).	
12. Presiona el Botón siguiente (4).	13. Se muestra la Interfaz III para asignarle a cada campo el componente que lleva.
14. Selecciona el tipo de interfaz que desea generar (12).	
15. Decide si el formulario llevara un gris (13).	
16. Selecciona el tipo de componente que se le asignará a un campo (14), seleccionando también el campo (15) y auxiliándose del botón ">>" (16) incorpora en una nueva lista (17) que componente debe tener cada campo.	
17. Presiona el Botón Generar. (11)	18. El sistema genera el formulario.

Cursos alternos

Línea 5: Si el actor presiona Siguiente sin tener los datos llenos se envía el siguiente mensaje:



Línea 12: Si el actor presiona Siguiente sin tener los seleccionados los campos que debe llevar el formulario se envía el siguiente mensaje:



Caso de uso:	Generar Conexión
Actores:	Programador (inicia).
Propósito:	Generar el fichero de conexión que tendrá una aplicación de forma dinámica
Resumen	El caso de uso se inicia cuando el programador necesita crear un fichero de conexión de su proyecto y le introduce una serie de datos al sistema para que este se lo genere de forma dinámica.
Precondiciones:	Se debe seleccionar primero el proyecto con el cual se va a trabajar.
Poscondiciones:	-
Tipo:	Real y expandido.
Responsabilidades:	RF8
Casos de uso relacionados:	-

Interfaz I

UCI. © 2006 - 2007 All rights reserved.
 Generador es un software libre bajo licencia GNU/GPL.

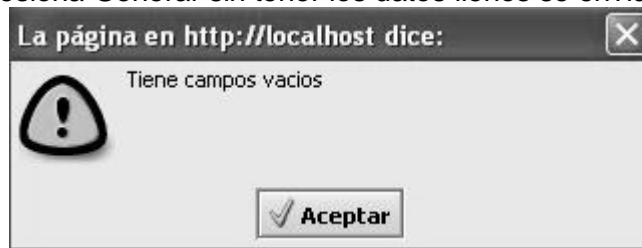
- (1) Servidor de Base de Datos al cual desea conectarse el usuario, se acepta IP o nombre del servidor.
- (2) Nombre de la Base de Datos donde se va a conectar el proyecto.
- (3) Nombre del usuario para poder conectarse al servidor.
- (4) Contraseña del usuario introducido para poder conectarse a la Base de Datos.

Curso normal de eventos para el caso de uso.

Acción del actor	Respuesta del sistema
1. El actor selecciona la opción Nueva Conexión del menú principal (flujo básico).	2. El sistema muestra una serie de datos que el usuario debe llenar para generar la conexión, dentro de los que se encuentra: Servidor (1), Base de Datos (2), usuario (3) y contraseña (4).
3. Llena el campo del Servidor(1)	
4. Llena el campo de la Base de Datos (2)	
5. Llena el campo del usuario (3).	
6. Llena el campo del contraseña (4).	
7. Presiona el Botón Generar (5).	6. Con los datos introducidos genera el fichero de conexión.

Cursos alternos

Línea 7: Si el actor presiona Generar sin tener los datos llenos se envía el siguiente mensaje:



Caso de uso:	Generar Típicas
Actores:	Programador (inicia).
Propósito:	Generar clases típicas de forma dinámica
Resumen	El caso de uso se inicia cuando el programador necesita crear una serie de clases de acceso a daros o típicas y le introduce una serie de parámetros del servidor de base de datos al sistema para que este se las genere de forma dinámica.
Precondiciones:	Se debe seleccionar primero el proyecto con el cual se va a trabajar.
Poscondiciones:	-
Tipo:	Real y expandido.
Responsabilidades:	RF7
Casos de uso relacionados:	-

Interfaz I

(1) Nombre de la base de datos de la cual se desea generar las clases típicas.

Interfaz II

http://localhost - I - Mozilla Firefox

GenTipFAR

Usuario

Contraseña

Esquema

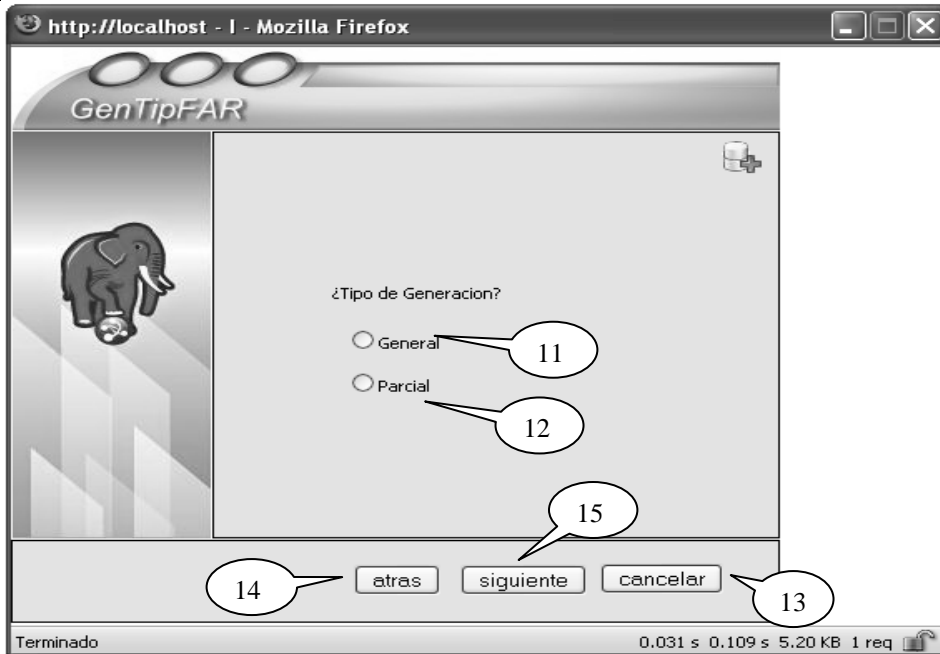
Host

atras siguiente cancelar

Terminado 0.063 s 0.094 s 5.17 KB 1 req

(4) Nombre del usuario con el cual se va a conectar a la Base de Datos.
(5) Contraseña del usuario con el cual se va a conectar a la Base de Datos.
(6) Esquema de la Base de Datos de la cual se va a generar las típicas.
(7) IP del servidor donde se encuentra la Base de Datos.

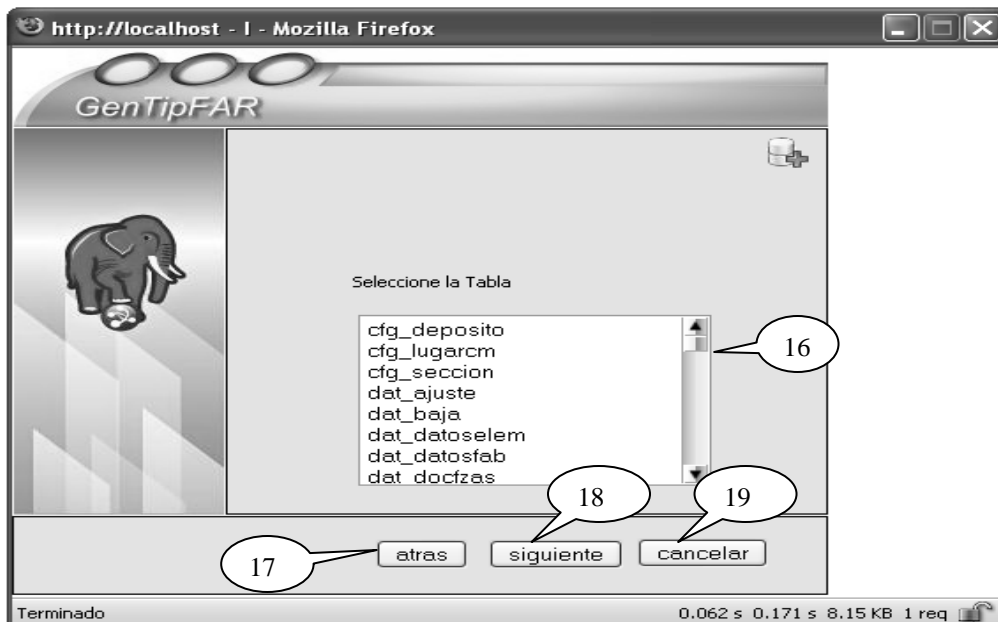
Interfaz III



(11) Si se desea generar todas las clases típicas de ese esquema se debe marcar esta opción.

(12) Si se desea generar algunas las clases típicas de algunas tablas del esquema se debe marcar esta opción.

Interfaz IV



(16) Lista con todas las tablas de la base de datos, donde se debe seleccionar a las que se desea generarle las clases típicas

Interfaz V



Curso normal de eventos para el caso de uso.

Acción del actor	Respuesta del sistema
1. El actor selecciona la opción Crear Clases Típicas del menú principal (flujo básico).	2. El sistema muestra la Interfaz I donde se debe introducir el Nombre de la Base de Datos a la cual se va a conectar la aplicación para generar las clases típicas.
3. Llena el campo del nombre de la base de datos (1).	
4. Presiona el Botón Siguiente (2).	5. El sistema muestra la Interfaz II con el resto de los datos necesarios para poder conectarse a la base de datos y generar las clases típicas.
6. Llena el campo del nombre del usuario para conectarse de la base de datos (4).	
7. Llena el campo de la contraseña del usuario para conectarse de la base de datos (5).	8. El sistema actualiza las tablas que pertenecen a ese esquema seleccionado (6).
9. Llena el campo del esquema de la base de datos (6) con el cual se va a trabajar.	10. El sistema actualiza los campos que pertenecen a esa tabla seleccionada (7).
11. Llena el campo del host (7) donde se encuentra la base de datos con el cual se va a trabajar.	
12. Presiona el Botón siguiente (9).	13. Se muestra la Interfaz III para elegir que tipo de generación de código se desea hacer.

14. Selecciona el tipo de generación de típica parcial (11).	
15. Presiona el Botón siguiente (15).	16. El sistema muestra la interfaz IV donde el usuario debe escoger que tablas desea crearle las clases típicas.
17. Decide que tablas utilizará usar (16).	
18. Presiona el Botón siguiente (18).	19. El sistema muestra la Interfaz V donde expresa que las clases típicas fueron creadas satisfactoriamente.

Cursos alternos

Línea 4: Si el actor presiona Siguiente sin llenar el nombre de la base de datos envía el siguiente mensaje:



Línea 12: Si el actor presiona Siguiente sin tener los datos llenos se envía el siguiente mensaje:



Línea 12: Si el actor presiona Atrás (8) el sistema muestra nuevamente la Interfaz I.

Línea 15: el actor presiona Atrás (14) el sistema muestra nuevamente la Interfaz II.

Línea 18: el actor presiona Atrás (17) el sistema muestra nuevamente la Interfaz III.

Línea 14: Si el actor decide seleccionar generar típicas de forma general, el sistema muestra la Interfaz V donde expresa que las clases típicas fueron creadas satisfactoriamente.

Conclusiones:

En este capítulo se comenzó a desarrollar la propuesta de solución, obteniéndose a partir del análisis de los procesos del negocio, un listado con las funciones que debe tener el sistema, que se representaron mediante un Diagrama de Casos de Uso, y finalmente se describieron paso a paso todas las acciones de los actores del sistema con los casos de uso con los que interactúan. Gracias a esto ahora se puede empezar a construir el sistema, tratando de que se cumplan todos los requerimientos y las funciones que han sido consideradas necesarias en este capítulo.

CAPITULO III: ANÁLISIS Y DISEÑO DEL SISTEMA.

Introducción:

En este capítulo se detallarán los artefactos necesarios para la construcción de la propuesta solución, se transitará por una fase importantísima de la ingeniería del software en la cual se desarrolla el análisis y el diseño del sistema, además de establecer patrones de arquitectura, de diseño y mecanismos de diseño que permiten obtener un producto de calidad.

3.1 Análisis:

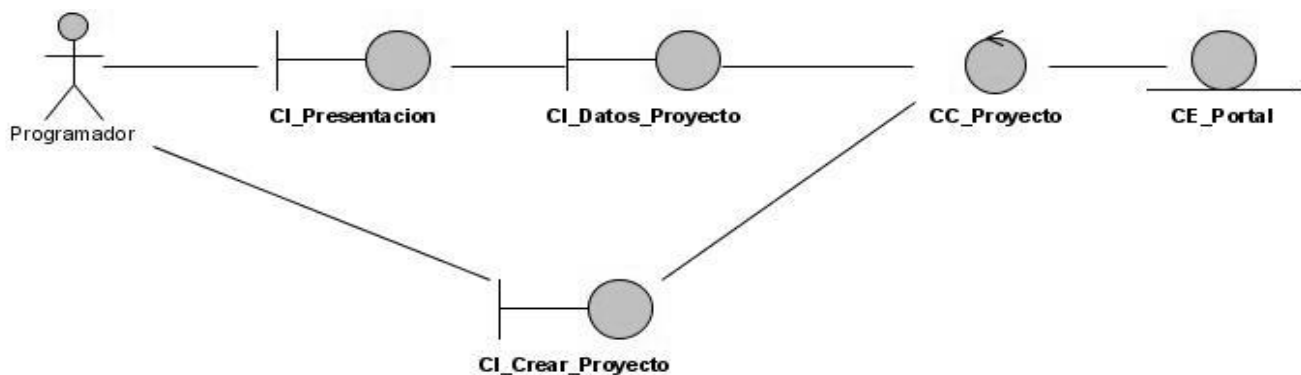
3.1.1 ¿Qué es el Análisis?

Durante las dos pasadas décadas, se han desarrollado un gran número de métodos de modelado. Los investigadores han identificado los problemas del análisis y sus causas y han desarrollado varias notaciones de modelado y sus correspondientes conjuntos de heurísticas para solucionarlos. Cada método de análisis tiene su punto de vista. Sin embargo, todos los métodos de análisis se relacionan por un conjunto de principios operativos:

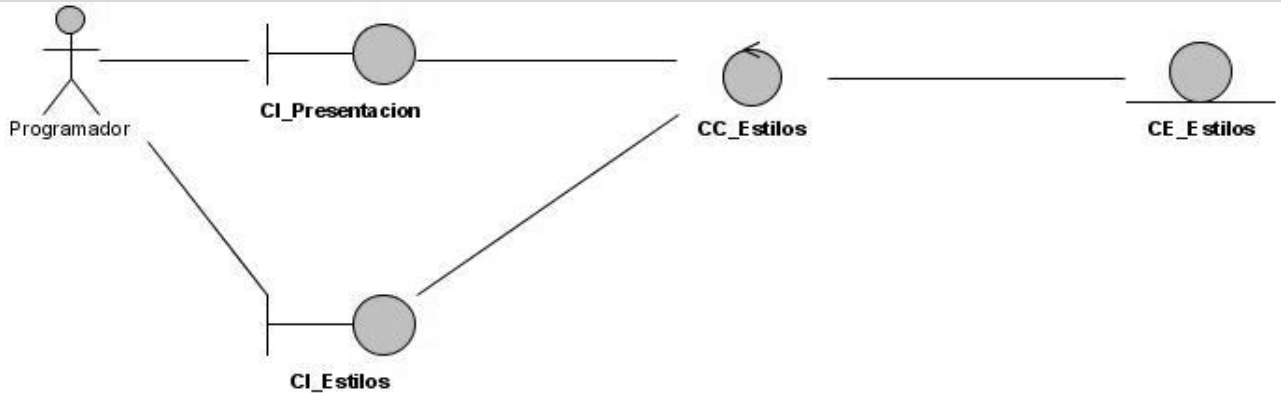
1. Debe representarse y entenderse el dominio de información de un problema.
2. Deben definirse las funciones que debe realizar el software.
3. Debe representarse el comportamiento del software (como consecuencia de acontecimientos externos).
4. Deben dividirse los modelos que representan información, función y comportamiento de manera que se descubran los detalles por capas (o jerárquicamente).
5. El proceso de análisis debería ir desde la información esencial hasta el detalle de la implementación.

3.1.2 Diagramas de clases del Análisis:

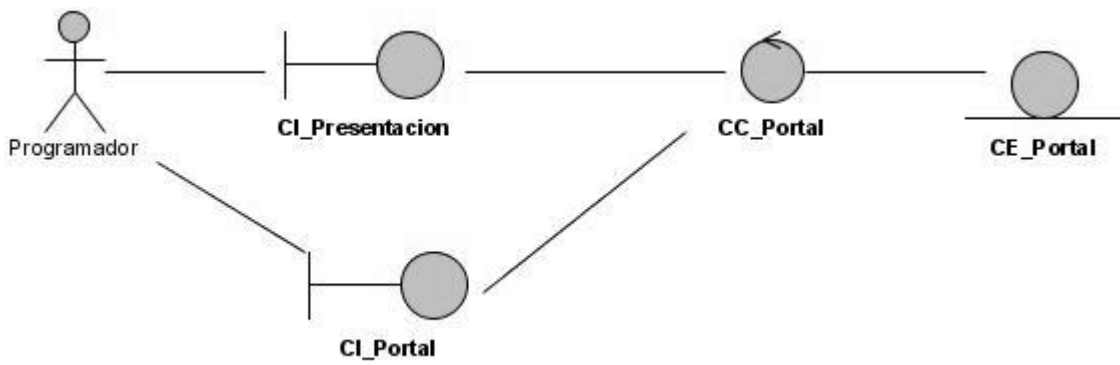
Generar Proyecto:



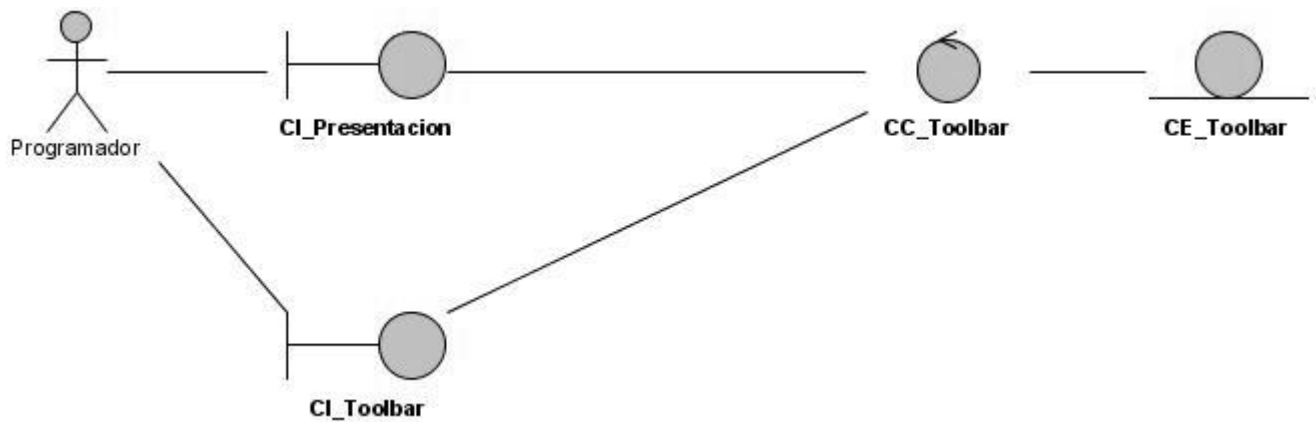
Generar Estilos:



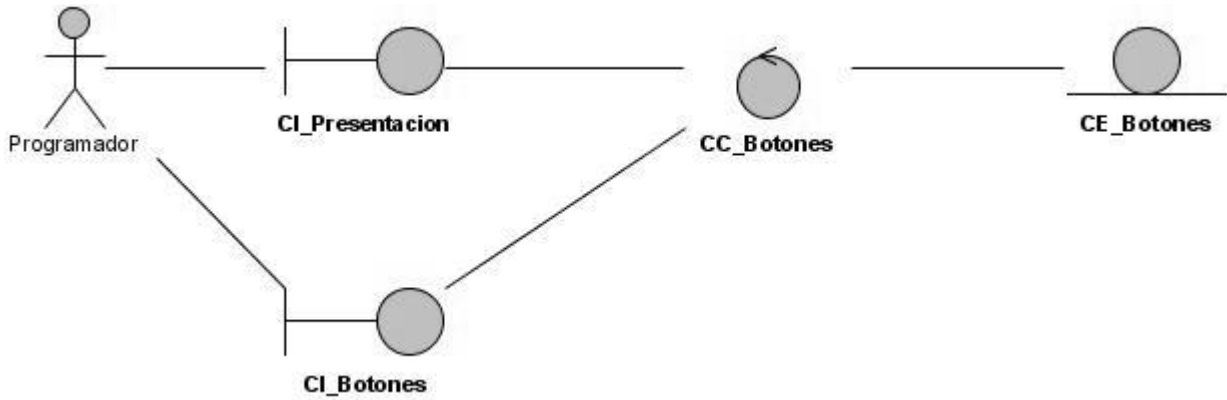
Generar Portal:



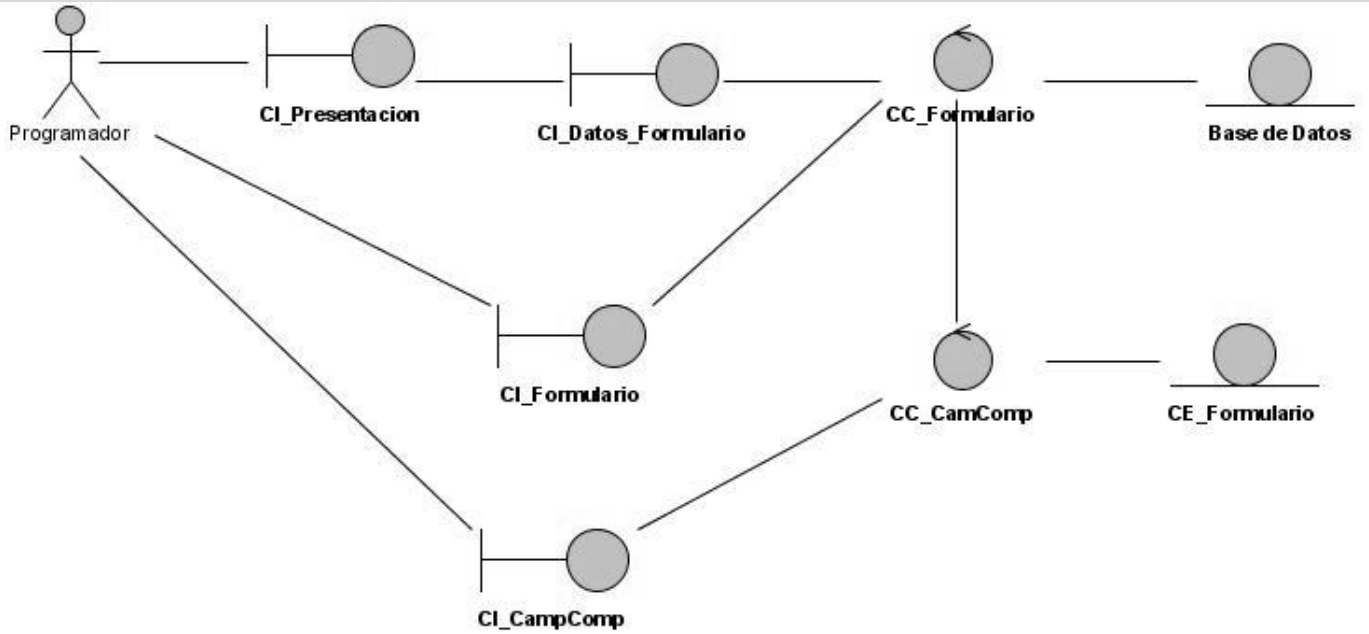
Crear Toobar



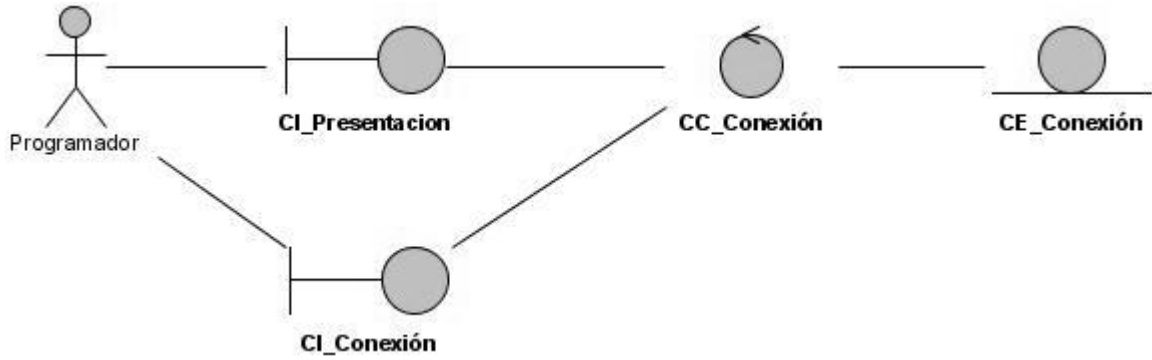
Crear Botones:



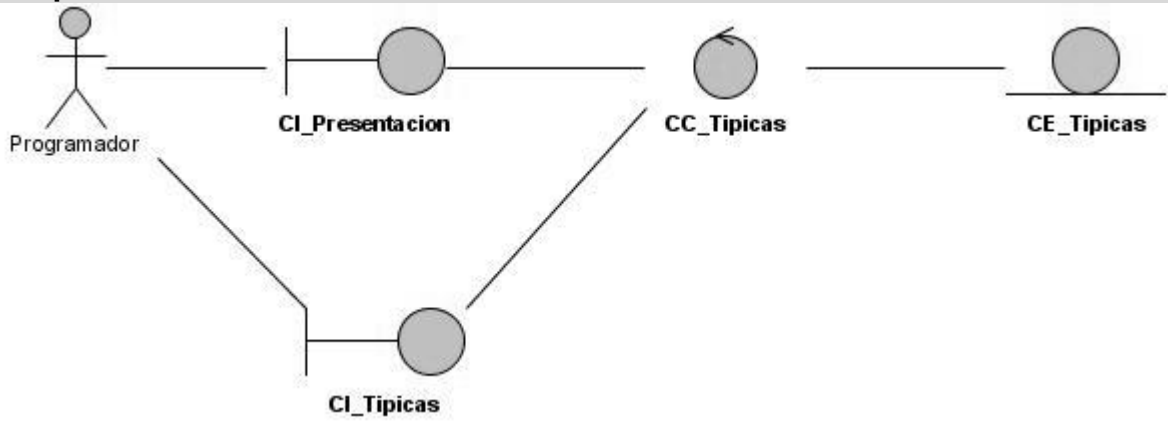
Generar Formulario:



Generar Conexión:



Generar Típicas:



3.2 Diseño:

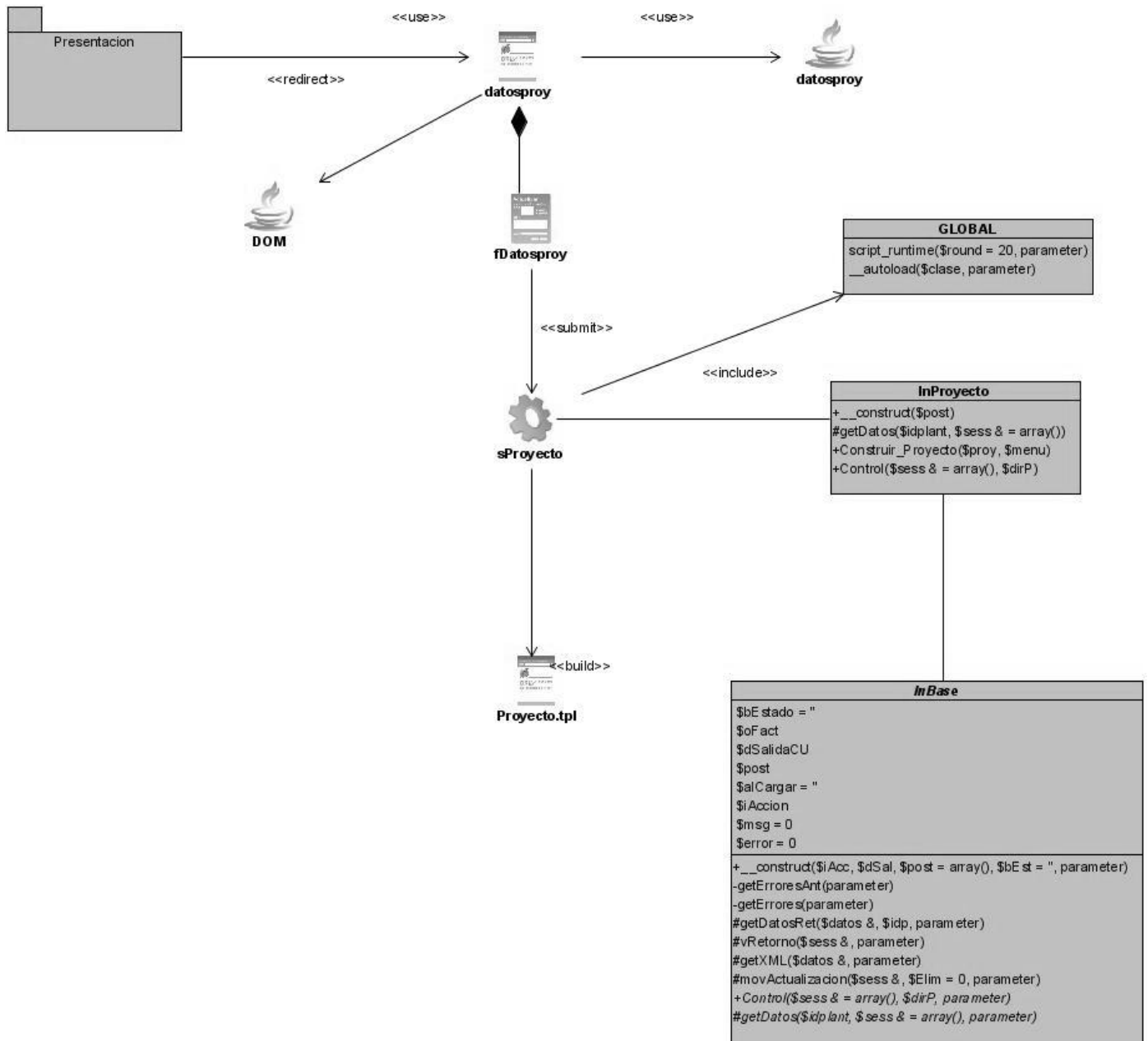
3.2.1 ¿Qué es el diseño?

El diseño del software se encuentra en el núcleo técnico de la ingeniería del software y se aplica independientemente del modelo de diseño de software que se utilice. Una vez que se analizan y especifican los requisitos del software, el diseño del software es la primera de las tres actividades técnicas (diseño, generación de código y pruebas) que se requieren para construir y verificar el software. Cada actividad transforma la información de manera que de lugar por ultimo a un software de computadora validado.

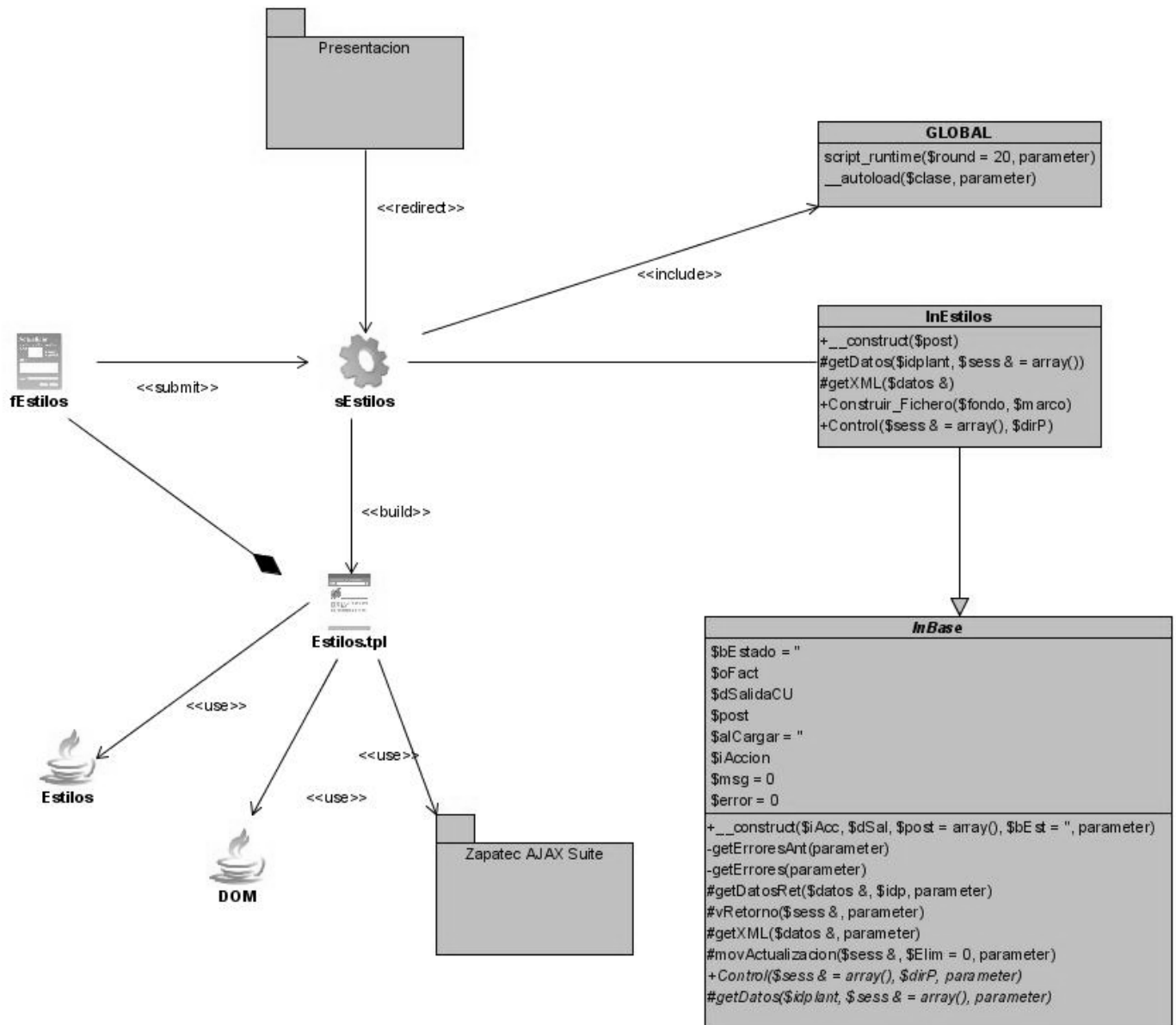
La importancia del diseño del software se puede describir con una sola palabra: *calidad*. El diseño es el lugar en donde se fomentan la calidad en la ingeniería del software. El diseño proporciona las representaciones del software que se pueden evaluar en cuanto a calidad. El diseño es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado. El diseño del software sirve como fundamento para todos los pasos siguientes del soporte del software y de la ingeniería del software. Sin un diseño, corremos el riesgo de construir un sistema inestable, un sistema que fallará cuando se lleven a cabo cambios; un sistema que puede resultar difícil de comprobar; y un sistema cuya calidad no puede evaluarse hasta muy avanzado el proceso, sin tiempo suficiente. [8]

3.2.2 Diagramas de clases del Diseño:

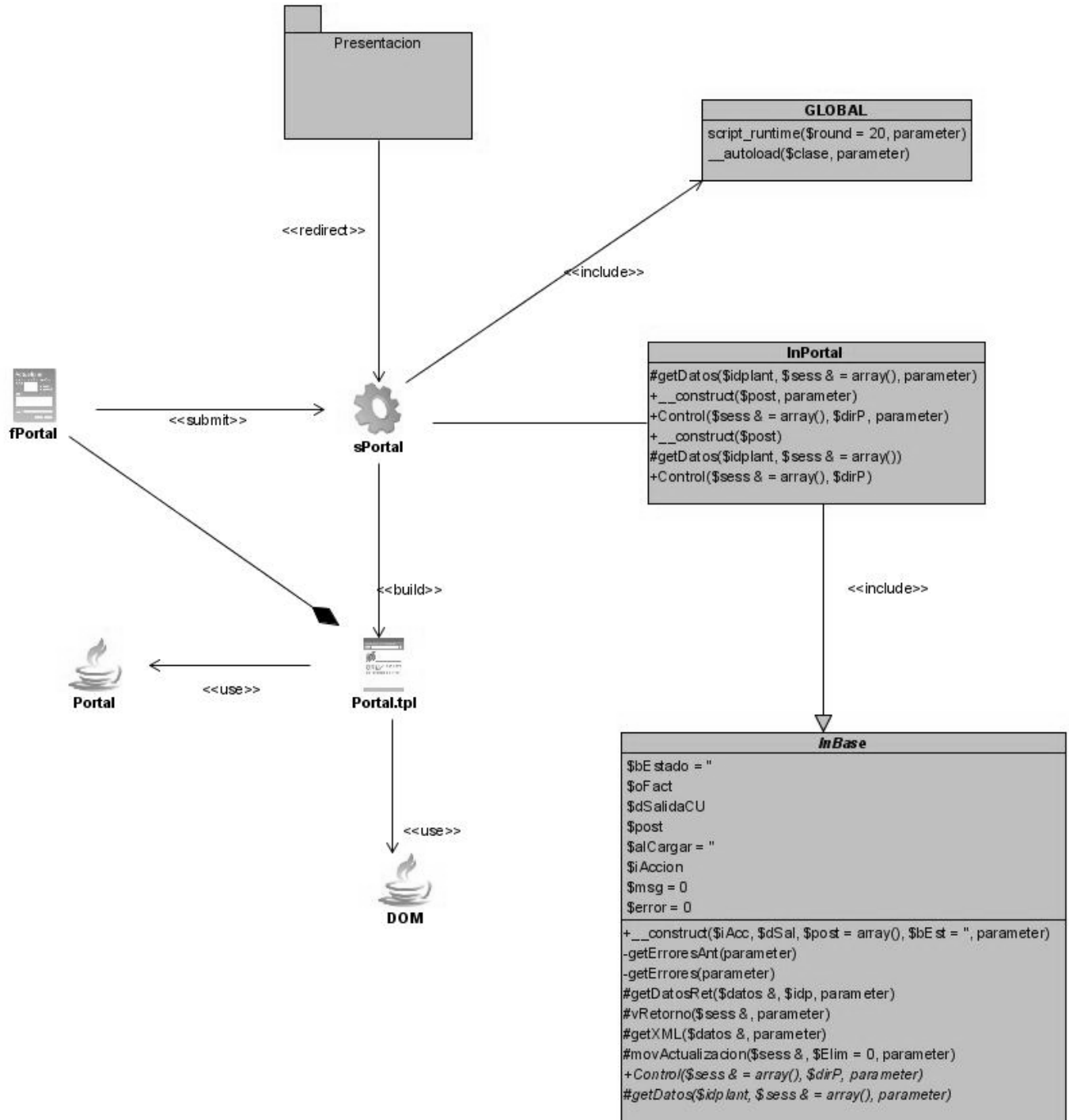
Generar Proyecto:



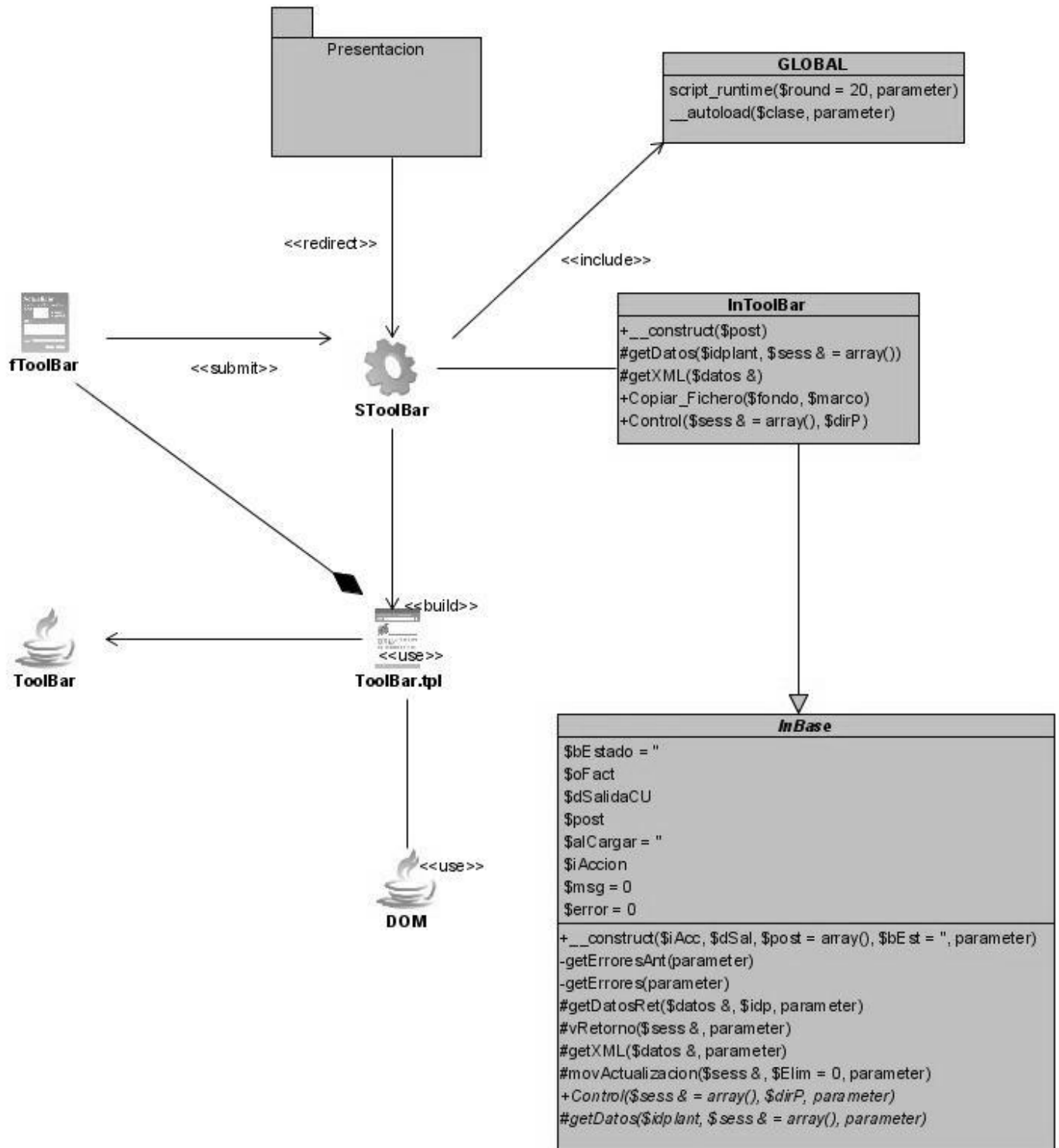
Generar Estilos:



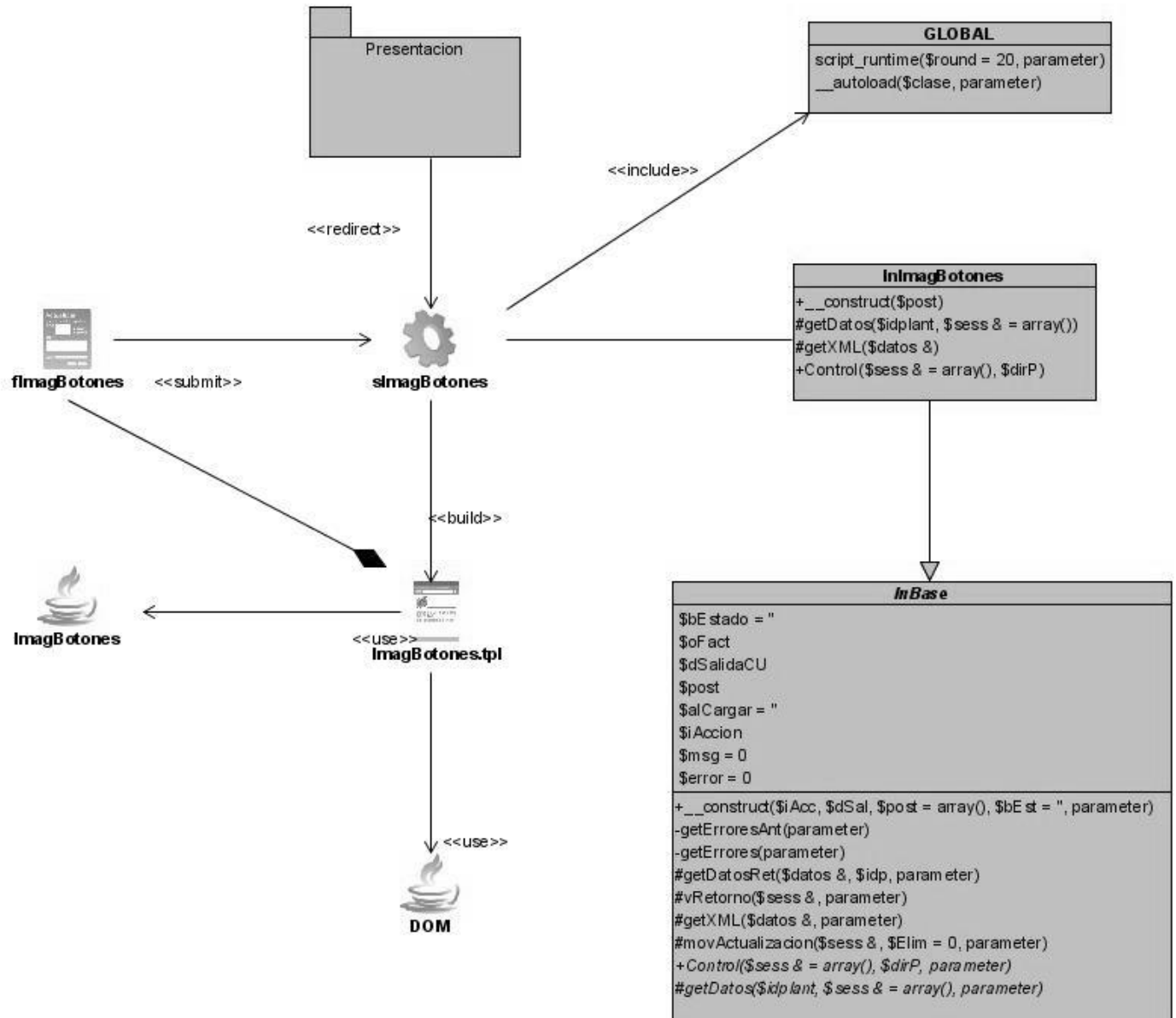
Generar Portal:



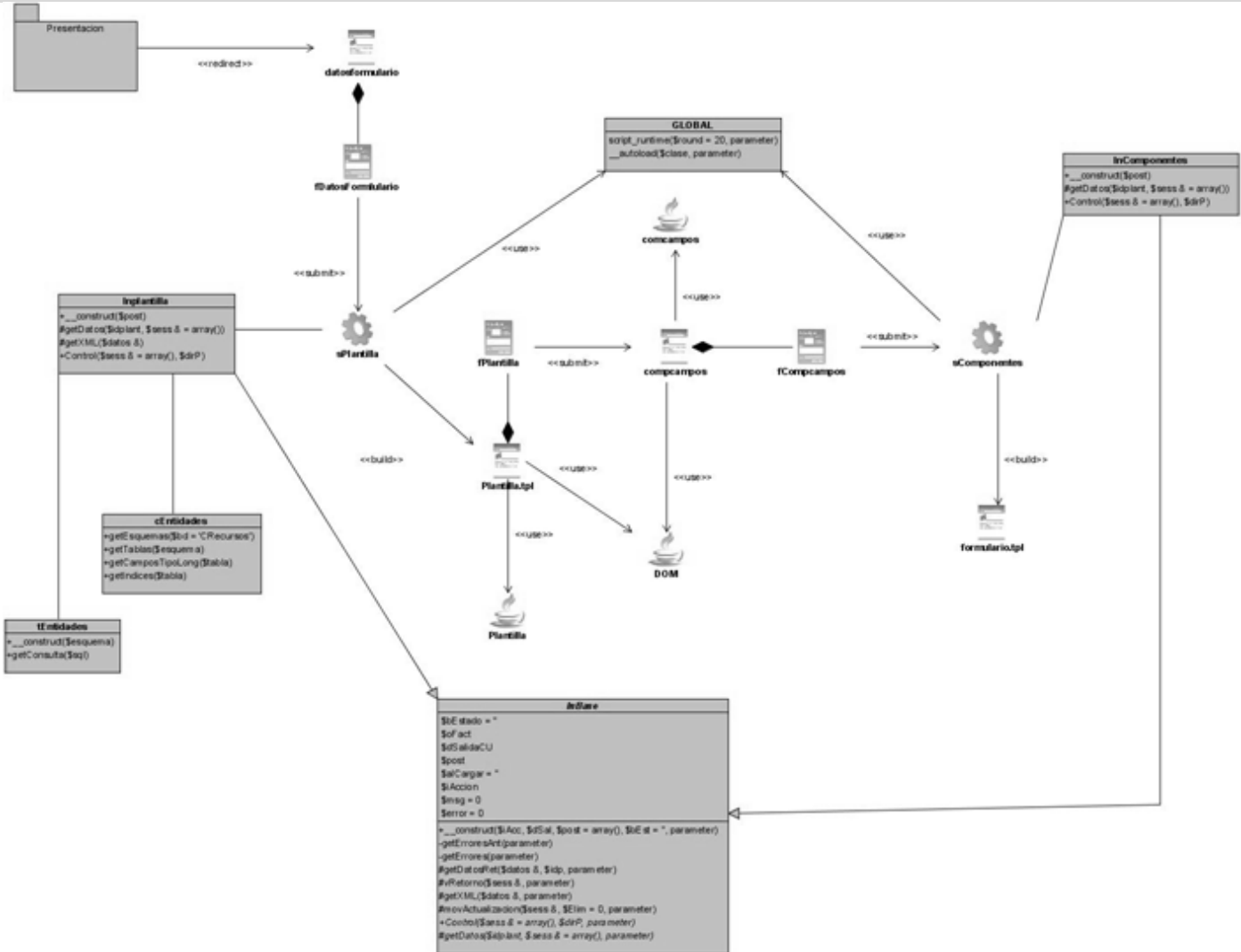
Crear Toobar:



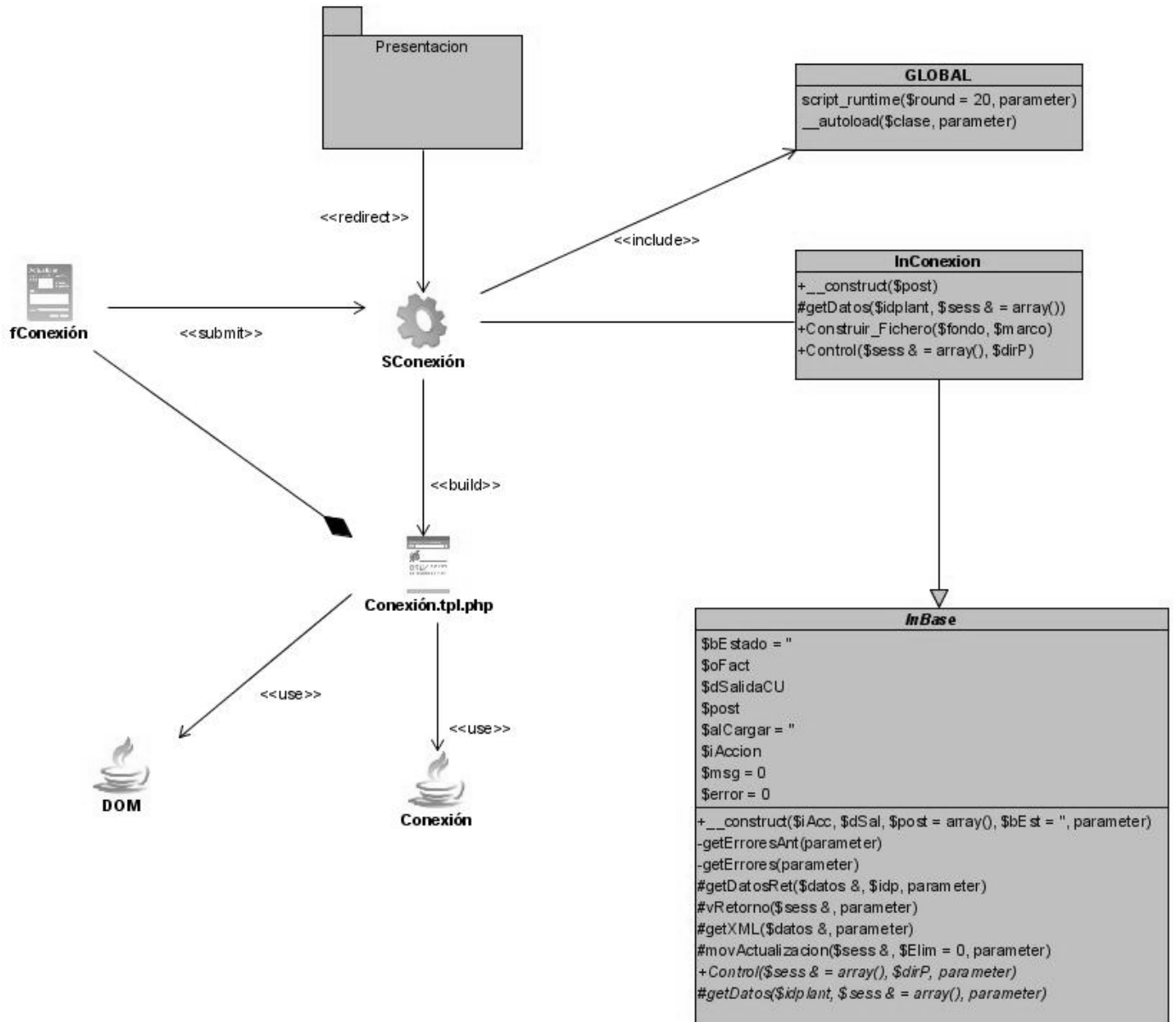
Crear Botones:



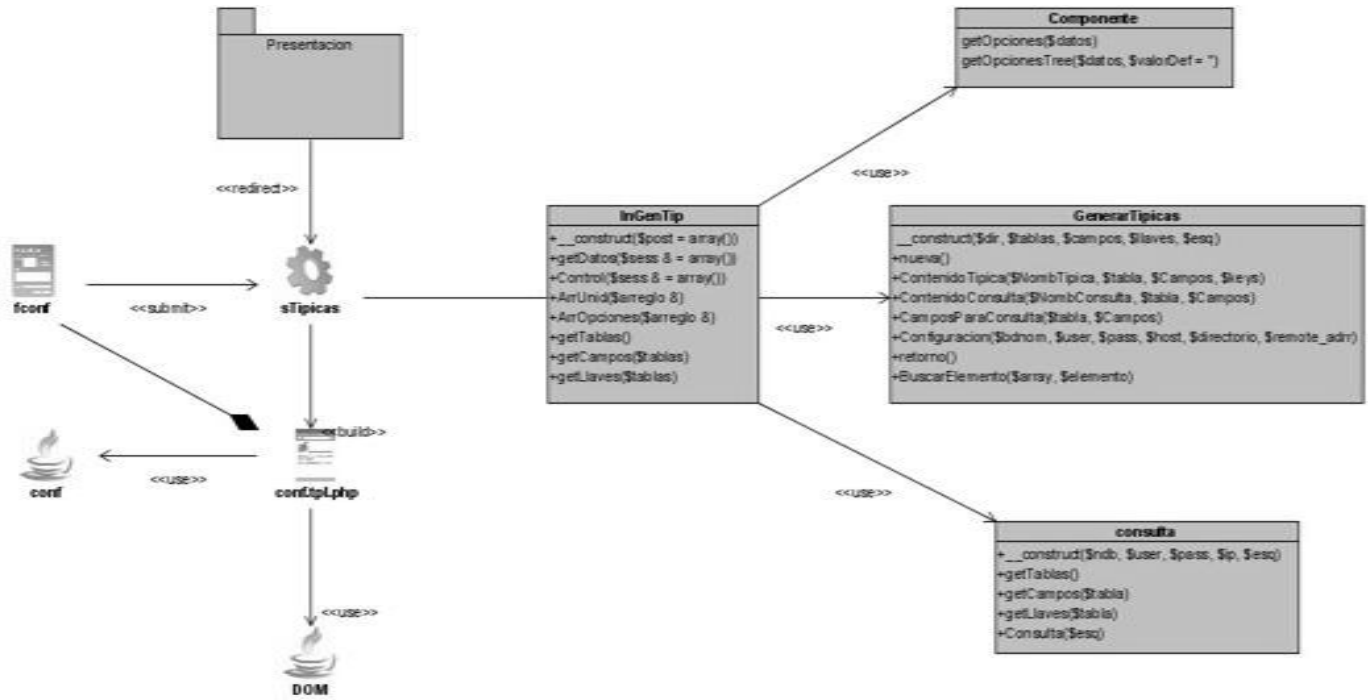
Generar Formulario:



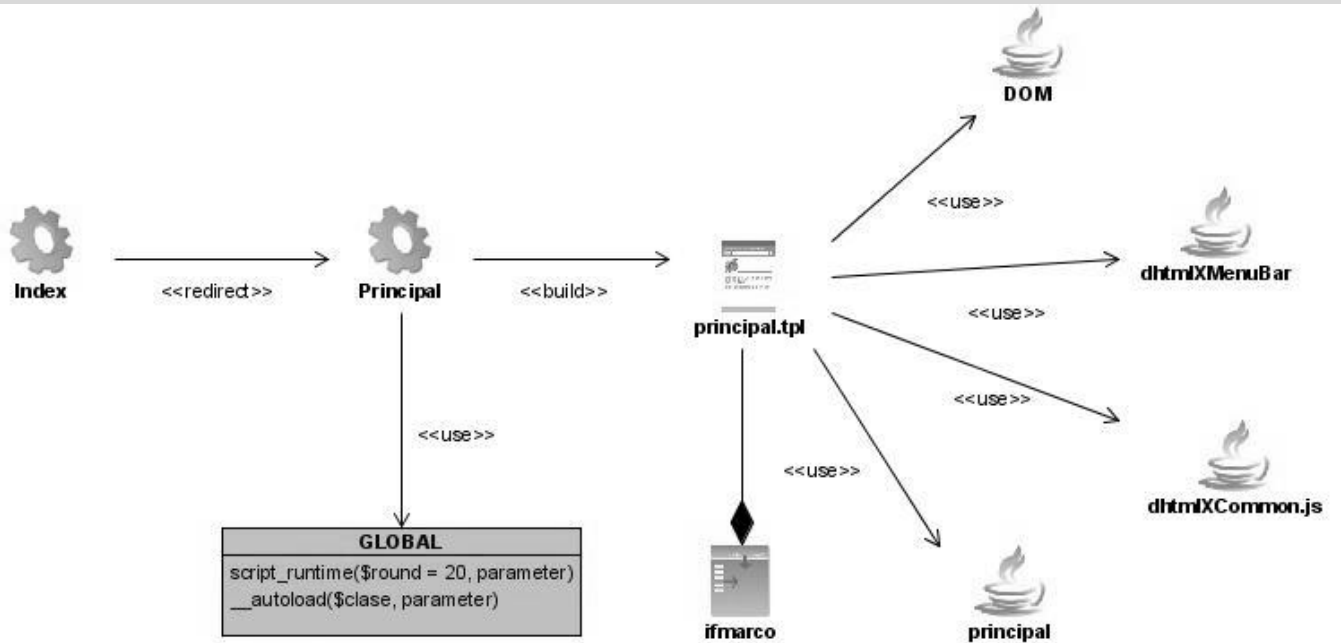
Generar Conexión:



Generar Clases Típicas:



Presentación:

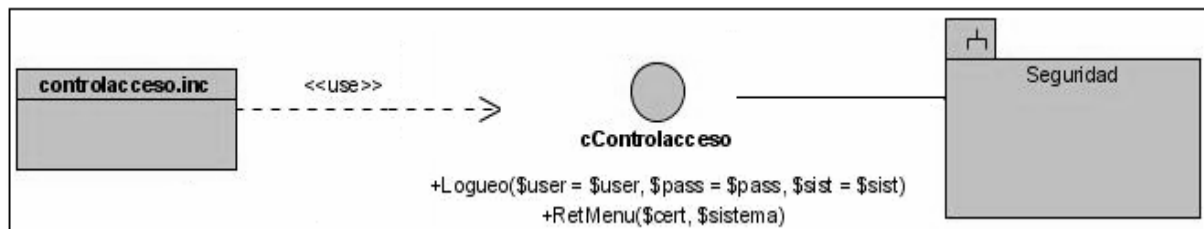


3.3 Mecanismos de Diseño:

3.3.1 Mecanismos de Seguridad:

Los mecanismos de seguridad de la aplicación son similares a los de la aplicación generada, donde se define, autenticación, autorización, comunicación segura y administración de perfiles.

A continuación se presenta el diagrama de clases genérico para la parte de seguridad, GLOBAL tiene una relación de include con controlacceso.inc. El subsistema Seguridad proporciona una clase de tipo interfaz llamada cControlacceso, la cual implementa dos métodos que son utilizados por controlacceso.inc.



3.3.2 Mecanismos de Acceso a Datos:

Este mecanismo surge por el mismo motivo que el de Seguridad, para acceder a los datos siempre estaban involucrados los mismos objetos y se efectuaban un conjunto de operaciones comunes en la mayoría de los casos de uso.

La vista estática de este mecanismo de acceso a datos muestra un conjunto de clases que interactúan para dar acceso y manipulación de los datos de la persistencia desde el nivel más bajo, es decir, utilizando los objetos nativos brindados por el entorno de desarrollo PHP como son PDO y PDOStatement, siguiendo así hasta la abstracción del acceso a datos a través de mEntidad de la cual heredan las clases particulares de nuestro sistema como Típicas.

¿Qué es Típicas?

Son clases que surgen también de la aplicación de patrones, en este caso, patrones de arquitectura y dentro de estos el patrón TABLE DATA GATEWAY, que tiene como supuesto realizar una clase para instanciar cada tabla de la base de datos.

3.4 Diagramas de Secuencia:

3.4.1 Diagrama de secuencia:

El Diagrama de Secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia se modela para cada caso de uso. Mientras que el diagrama de caso de uso permite el modelado de una vista 'business' del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos.

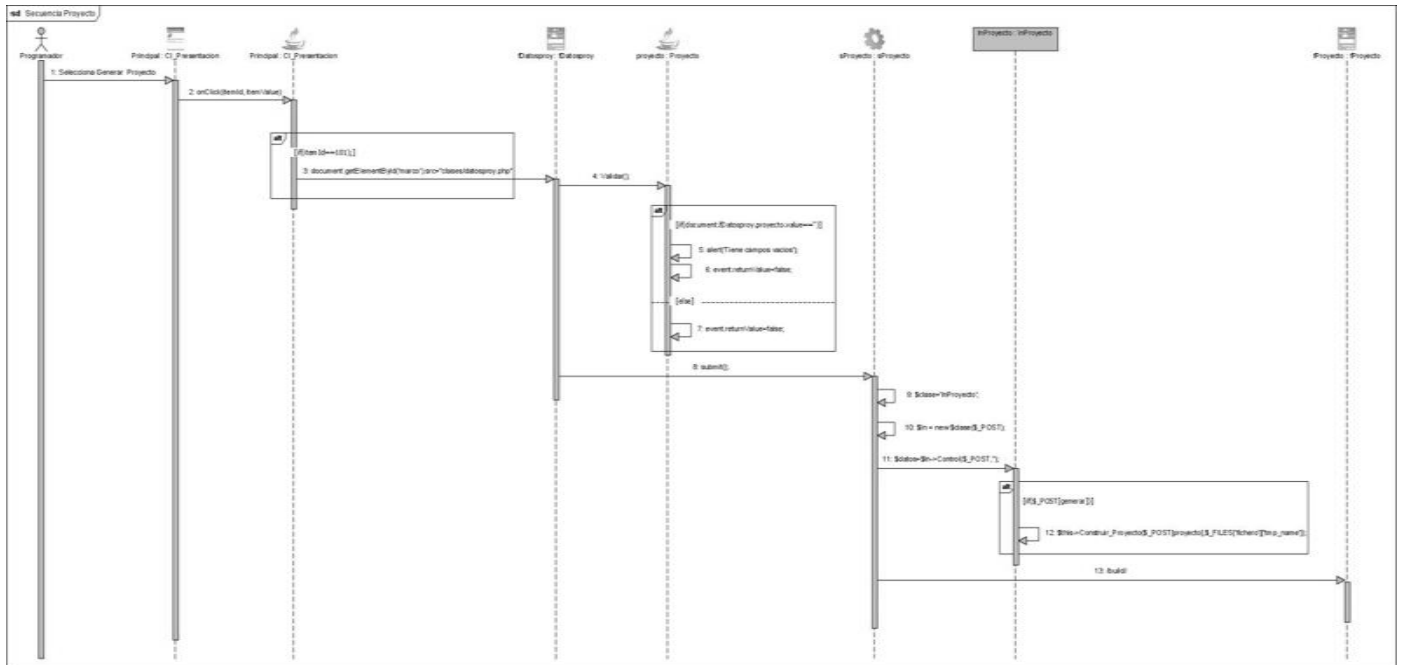
Típicamente uno examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si tienes modelada la descripción de cada caso de uso como una secuencia de varios pasos, entonces puedes "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos.

Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como vectores horizontales. Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria.

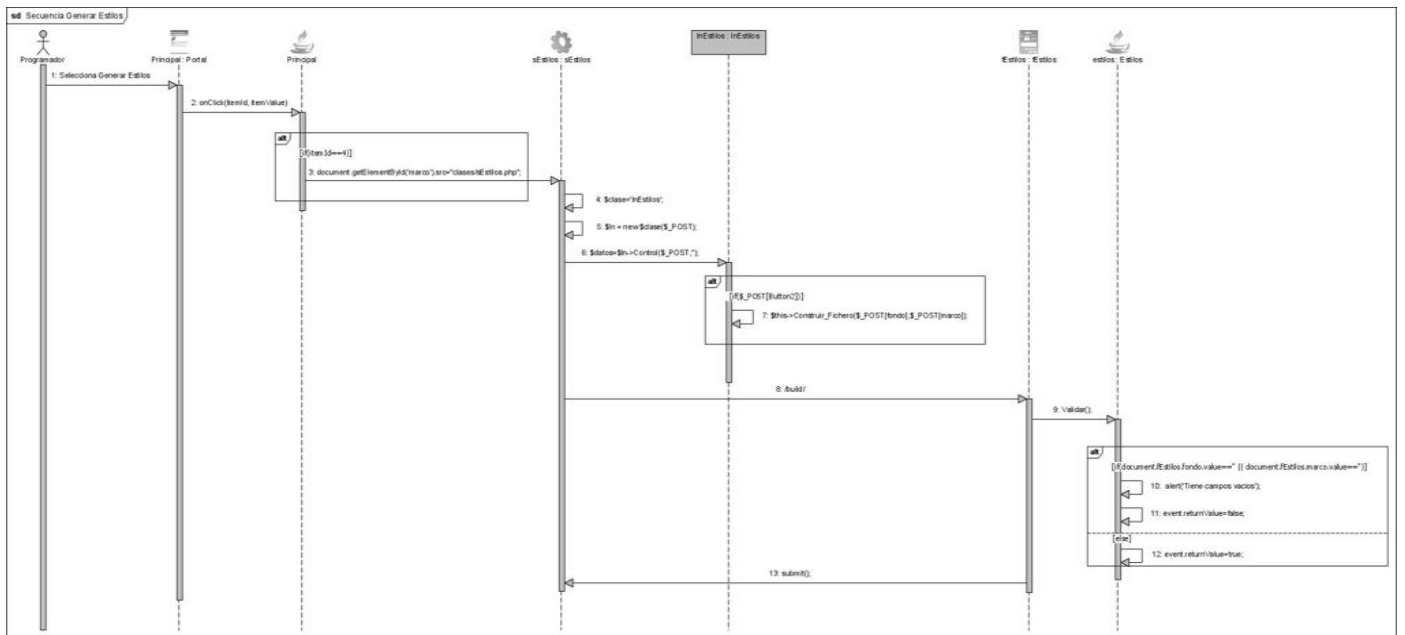
Durante el análisis inicial, el modelador típicamente coloca el nombre 'business' de un mensaje en la línea del mensaje. Más tarde, durante el diseño, el nombre 'business' es reemplazado con el nombre del método que está siendo llamado por un objeto en el otro. El método llamado, o invocado, pertenece a la definición de la case instanciada por el objeto en la recepción final del mensaje. [9]

3.4.2 Diagramas de secuencia de la aplicación:

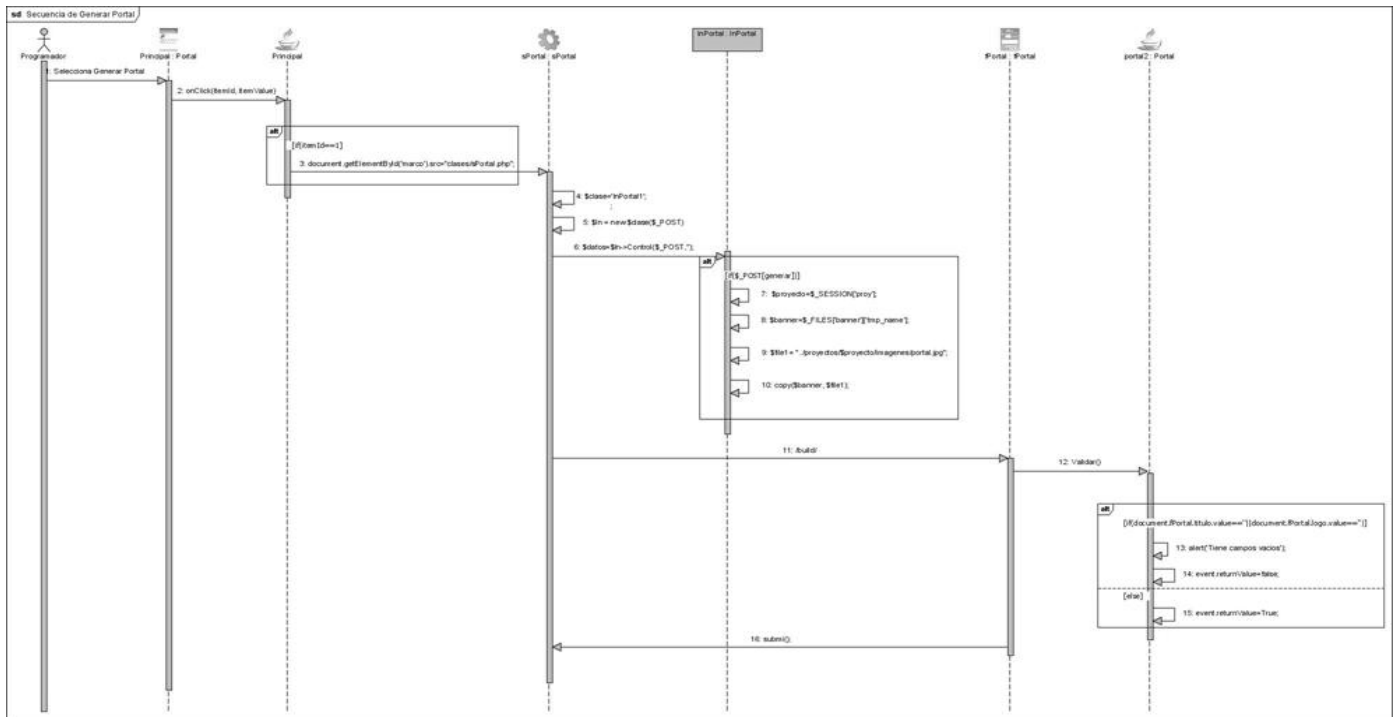
Generar Proyecto:



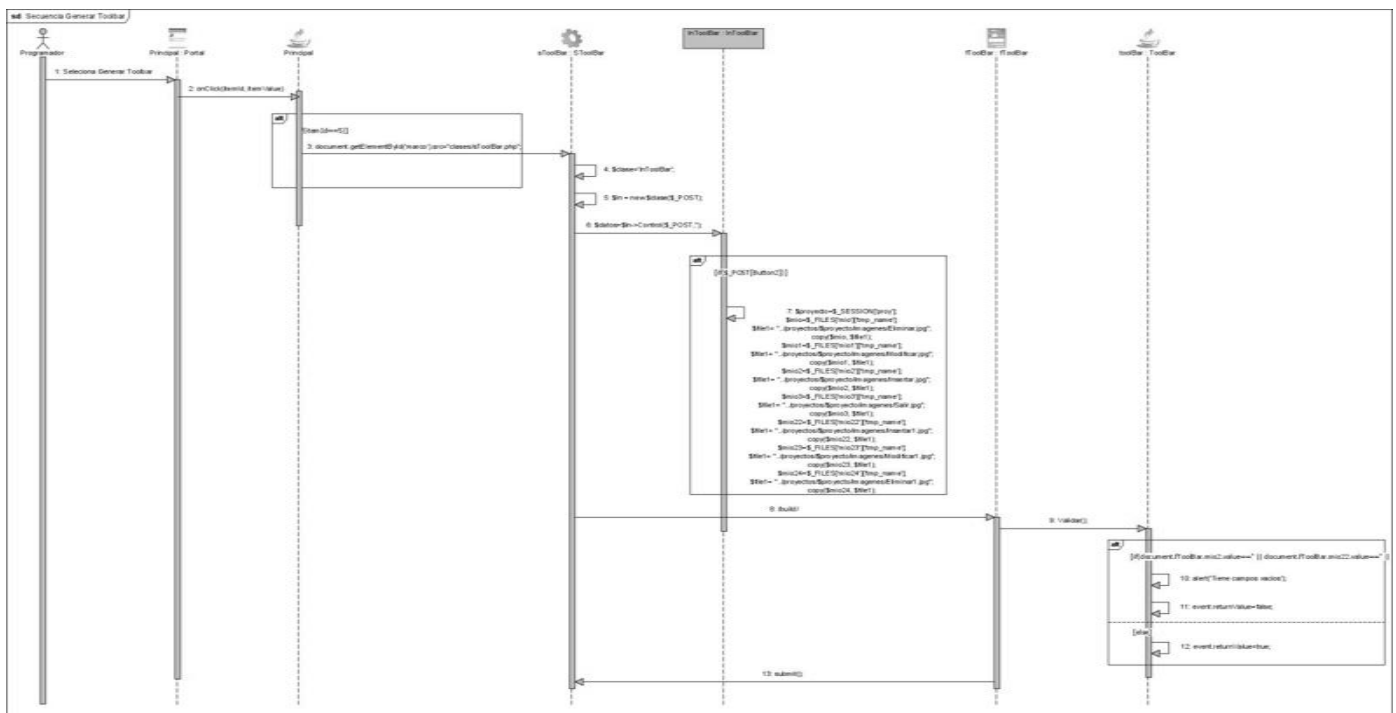
Generar Estilos:



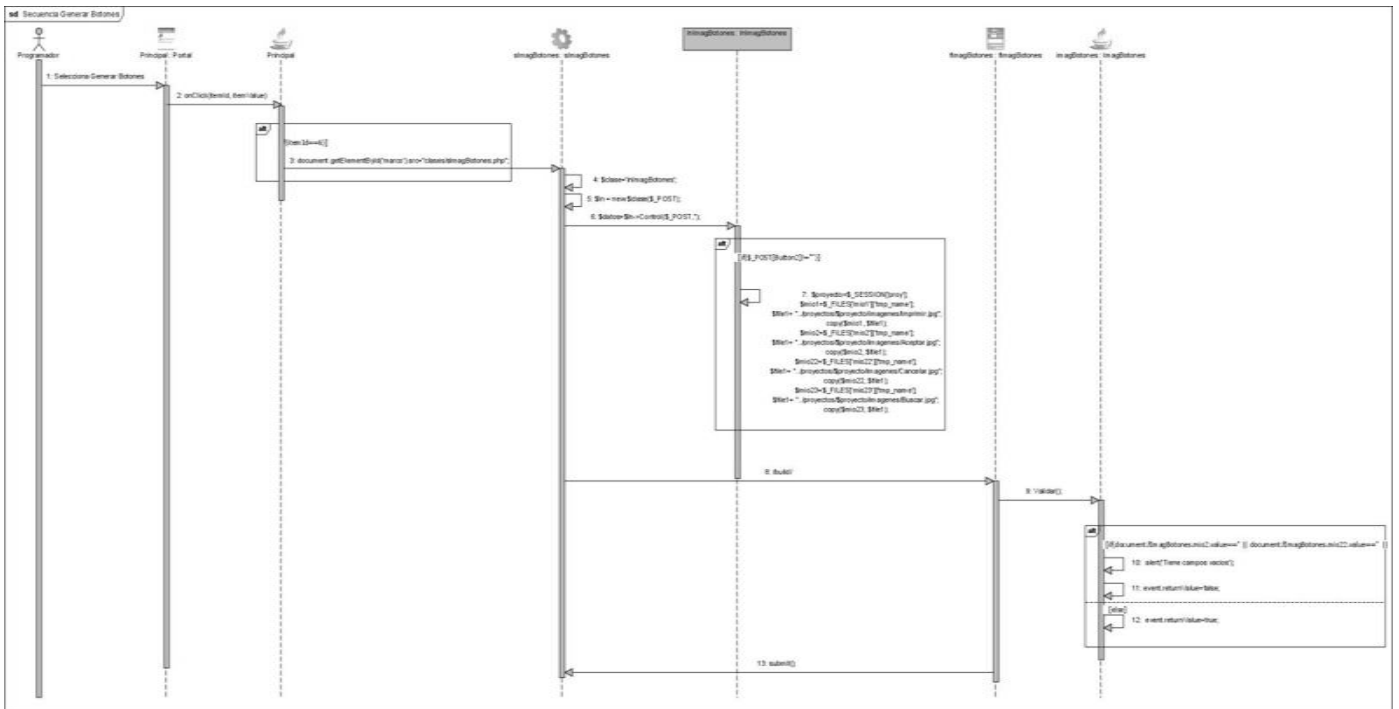
Generar Portal:



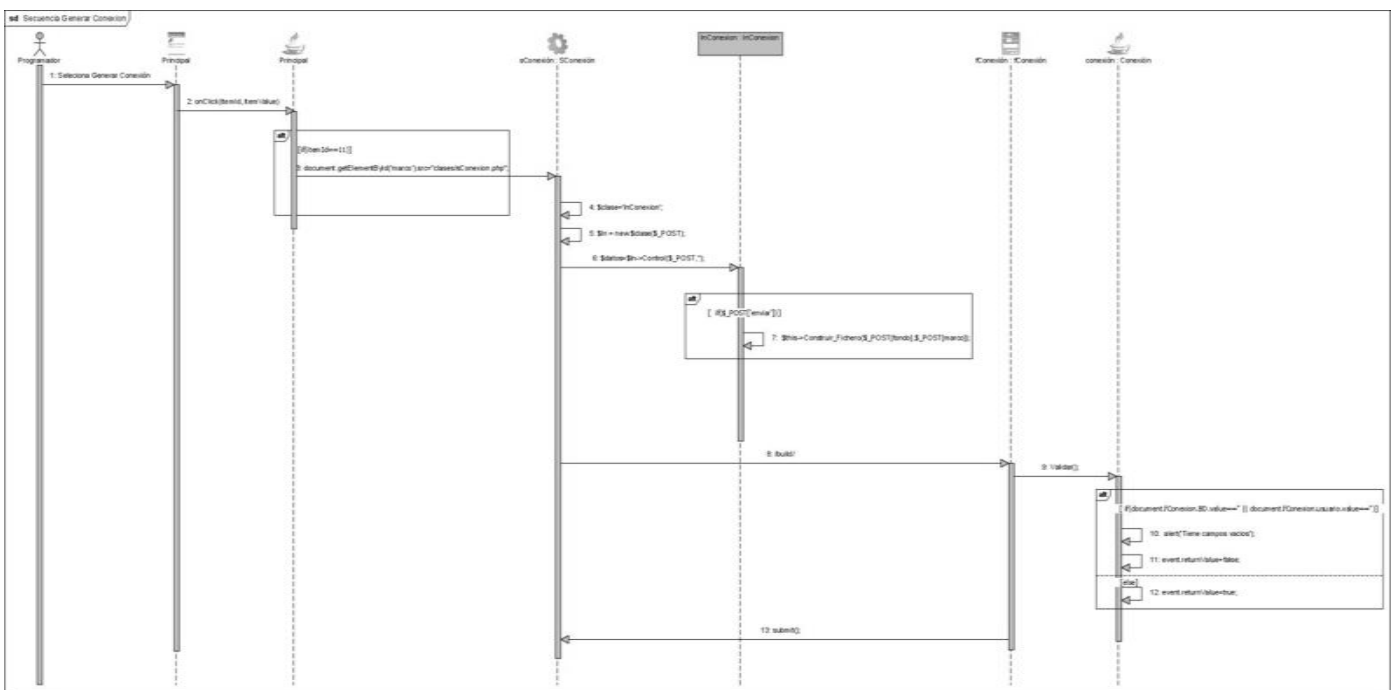
Crear Toobar



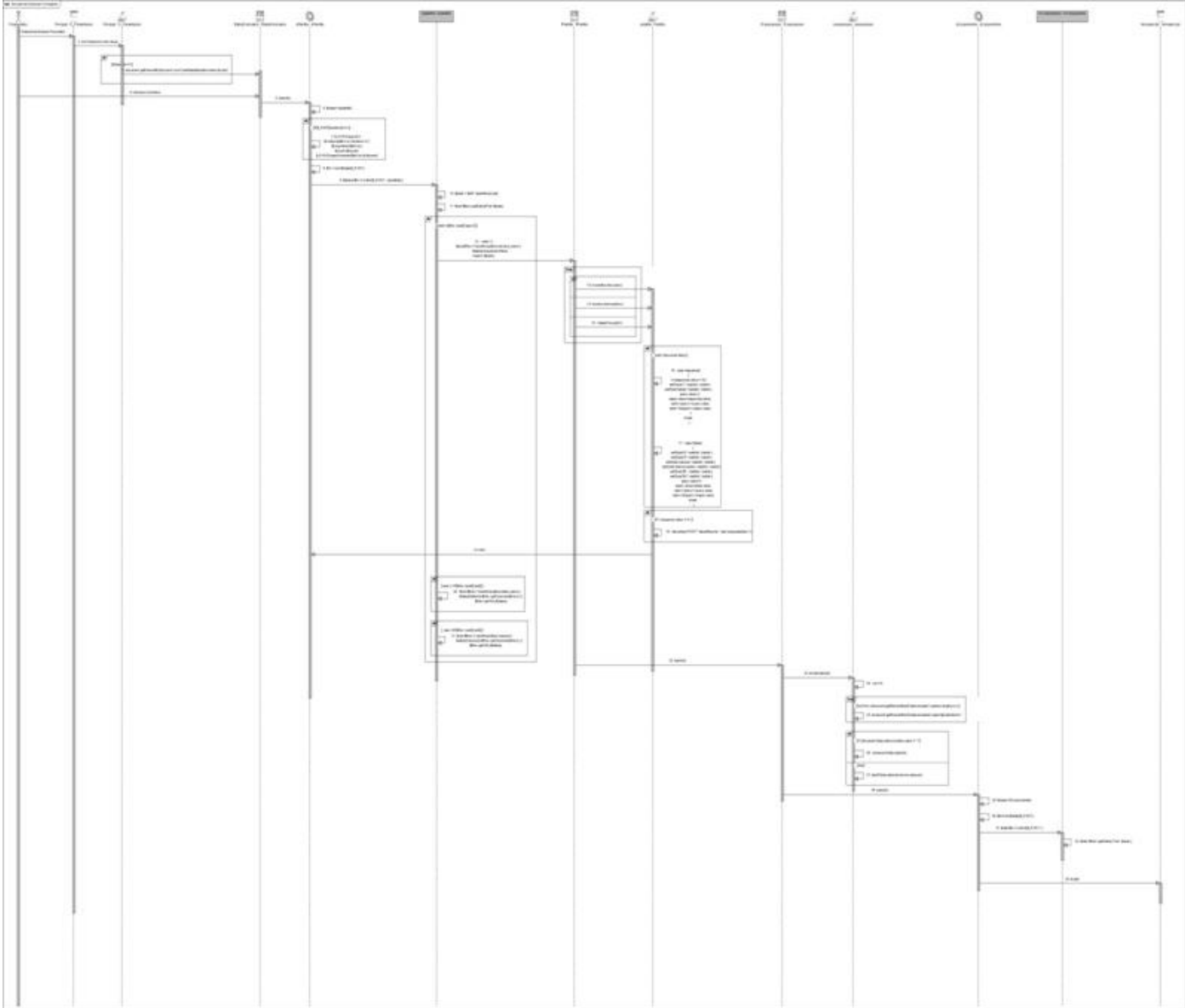
Crear Botones:



Generar Conexión:



Crear Formulario



3.5 Estándares de diseño:

La página principal de la aplicación, se concibe como un portal, con un menú, que no debe exceder de 3 niveles de profundidad, donde se agrupan las funcionalidades del sistema.

Las páginas deben tener una cabecera (banner) representativa, un área de trabajo y una barra menú con las opciones, además tener una hoja de estilo en común para lograr la uniformidad, es decir, se trabajará con la familia de fuentes Arial Helvética, Sans-Sheriff, el tamaño de la misma no debe diferir mucho de 11Px.

Los colores con los que se trabajarán serán tonalidades claras basadas en el verde, azul y amarillo, combinados con el color blanco o gris.

3.6 Tratamiento de Errores:

Los errores se tratan en una página especial que incluye el fichero de configuración general, y está preparada para recoger el número del error y presentar la pantalla con el error que le corresponde a ese código. En algunos casos incluye la forma de solucionar el error, como es el caso de la sesión, y la autenticación de los usuarios.

Algunos errores serán generados por funciones Java Script para evitar la ejecución de la página en vano. Este es el caso de los formularios de inserción/actualización, y las eliminaciones, se utilizan los errores en forma de mensajes de texto, como alerts de JavaScript, en la misma página donde se ejecutó la acción, de forma que el usuario pueda corregir más fácilmente y continuar, estos mensajes se generan donde se controla el error, solo que viajan entre las clases hasta llegar a la aportadora de contenido que es la que los mezcla finalmente con la salida del usuario. En operaciones muy largas o complicadas, se permite volver atrás, para revisar o modificar la información. Y se utilizan mensajes de confirmación, para acciones que son irreversibles como es el caso de las eliminaciones.

3.7 Descripción de las clases.

Nombre: InBase	
Tipo de clase: controladora.	
Atributo	Tipo
\$dSalidaCU	-
\$oFact	-
\$bEstado	-
\$msg	-
\$error	-
\$alCargar	-
\$iAccion;	-
Responsabilidades:	
Nombre:	getErroresAnt().
Descripción:	Permite dar salida a los mensajes de alerta o error registrados anteriormente en la clase.
Nombre:	getErrores().
Descripción:	Permite dar salida a los mensajes de alerta o error registrados en la clase.
Nombre:	getDatosRet(& \$datos, \$idp).
Descripción:	Introduce en el arreglo de datos a que aportara contenido el chequeo de errores, alertas y el id de la plantilla pasado al método geDatos. Este último será empleado por la aportadora de contenido para identificar que tag de Smarty debe rellenar dependiendo de la plantilla procesada.
Nombre:	vRetorno(& \$sess).
Descripción:	Verifica si se desea salir del Caso de Uso.
Nombre:	getXML(& \$datos).
Descripción:	Devuelve los datos en un XML.
Nombre:	movActualizacion(& \$sess, \$Elim = 0)
Descripción:	Resuelve el problema del movimiento por un navegador.
Nombre:	TransfArray (\$dat, \$indDenom).
Descripción:	Transforma los datos en un arreglo
Nombre:	getOpciones(\$datos, \$valorDef = -1, \$ajax = 0)
Descripción:	Convierte un arreglo en un select, en caso de que los datos hayan sido manipulados por AJAX, se le activa la variable \$ajax en 1.

Nombre: cEntidades	
Tipo de clase: controladora.	
Atributo	Tipo
-	-
Responsabilidades:	
Nombre:	getEsquemas(\$bd).
Descripción:	Devuelve el esquema de una Base de Datos que se le pasa de parámetro

Nombre:	getTablas(\$esquema).
Descripción:	Método que devuelve las tablas de una Base de Datos, pasándole de parámetro el esquema.
Nombre:	getCamposTipoLong(\$tabla).
Descripción:	Se encarga de devolverte los campos así y sus propiedades (el tipo de datos y la longitud) de una tabla que se le pasa de parámetro.

Nombre: InComponentes	
Tipo de clase: controladora.	
Atributo	Tipo
-	-
Responsabilidades:	
Nombre:	getDatos(\$idplant, & \$sess = array())
Descripción:	Se forman todos los componentes que llevara el formulario creados, así como la posición donde deben ir
Nombre:	Control(& \$sess = array(), \$dirP)
Descripción:	Se encarga de controlar la clase, llama al método getDatos.

Nombre: InEntidades.	
Tipo de clase: controladora.	
Atributo	Tipo
-	-
Responsabilidades:	
Nombre:	getDatos (\$idplant, & \$sess = array())
Descripción:	Según lo que se le pase como parámetro, dice que consulta debe ser ejecutada, si la que te devuelva el esquema, las tablas o los campos. Se encarga de la consulta sea ejecutada.
Nombre:	Control(& \$sess = array(), \$dirP)
Descripción:	Se encarga de controlar la clase, llama al método getDatos.

Nombre: InEstilos.	
Tipo de clase: controladora.	
Atributo	Tipo
-	-
Responsabilidades:	
Nombre:	Construir_Fichero(\$fondo,\$marco)
Descripción:	Construye el fichero del estilo según el color de fondo y el color del marco que se le pasen como parámetro.
Nombre:	Control(& \$sess = array(), \$dirP)
Descripción:	Se encarga de controlar la clase, llama la interfaz que se utilizará para introducir los datos.

Nombre: InImagBotones.	
Tipo de clase: controladora.	
Atributo	Tipo
-	-
Responsabilidades:	
Nombre:	Control(& \$sess = array(), \$dirP)
Descripción:	Se encarga de controlar la clase, llama la interfaz que se utilizará para introducir los datos.

Nombre: InPlantilla.	
Tipo de clase: controladora.	
Atributo	Tipo
-	-
Responsabilidades:	
Nombre:	getDatos(\$idplant, & \$sess = array())
Descripción:	Según lo que se le pase como parámetro, dice que consulta debe ser ejecutada, si la que te devuelva el esquema, las tablas o los campos. Se encarga de la consulta sea ejecutada.
Nombre:	Control(& \$sess = array(), \$dirP)
Descripción:	Se encarga de controlar la clase, llama la interfaz que se utilizará para introducir los datos. Así como definir los valores que serán cargados en los select de la plantilla.
Nombre:	getXML(& \$datos)
Descripción:	Se le pasa un arreglo y lo convierte a formato XML.

Nombre: InPortal.	
Tipo de clase: controladora.	
Atributo	Tipo
-	-
Responsabilidades:	
Nombre:	Control(& \$sess = array(), \$dirP)
Descripción:	Se encarga de controlar la clase, llama la interfaz que se utilizará para introducir los datos.

Nombre: InToolBar.	
Tipo de clase: controladora.	
Atributo	Tipo
-	-
Responsabilidades:	
Nombre:	Control(& \$sess = array(), \$dirP)
Descripción:	Se encarga de controlar la clase, llama la interfaz que se utilizará para introducir los datos.

Nombre: InProyecto.	
Tipo de clase: controladora.	
Atributo	Tipo
-	-
Responsabilidades:	
Nombre:	Construir_Proyecto(\$proy,\$menu)
Descripción:	Construye el proyecto según el nombre del proyecto y el menú que se le pasen como parámetro.
Nombre:	Control(& \$sess = array(), \$dirP)
Descripción:	Se encarga de controlar la clase, llama la interfaz que se utilizará para introducir los datos.

Nombre: tComponentes.	
Tipo de clase: controladora.	
Atributo	Tipo
\$imagenes	-
\$estilos	-
\$js	-
\$xml	-
\$posx	-
\$posy	-
\$ancho	-
\$largo	-
\$id	-
\$datos	array
Responsabilidades:	
Nombre:	grid()
Descripción:	Forma el código del gris y retorna una cadena con todo el código que este lleva
Nombre:	tree()
Descripción:	Forma el código del árbol y retorna una cadena con todo el código que este lleva
Nombre:	tabs()
Descripción:	Forma el código del tab y retorna una cadena con todo el código que este lleva
Nombre:	select()
Descripción:	Forma el código del select y retorna una cadena con todo el código que este lleva
Nombre:	text()
Descripción:	Forma el código del input de tipo text y retorna una cadena con todo el código que este lleva.
Nombre:	radio()
Descripción:	Forma el código del input de tipo radio y retorna una cadena con todo el código que este lleva.
Nombre:	hidden()
Descripción:	Forma el código del input de tipo hidden y retorna una cadena con todo el código que este lleva.

Nombre:	checkbox()
Descripción:	Forma el código del input de tipo checkbox y retorna una cadena con todo el código que este lleva.
Nombre:	getComponente(\$tipoComp,\$id,\$posx,\$posy,\$ancho,\$largo,\$xml)
Descripción:	Se encarga de devolver el componente que se solicita, así como de asignar las propiedades de los componentes.

Nombre: tEntidades.	
Tipo de clase: controladora.	
Atributo	Tipo
-	-
Responsabilidades:	
Nombre:	getConsulta(\$sql)
Descripción:	Se encarga de realizar cualquier consulta que se desee hacer a la Base de Datos con la que s esta trabajando.

Nombre: tPatron.	
Tipo de clase: controladora.	
Atributo	Tipo
\$ancho	-
\$largo	-
\$titulo	-
\$imagenes	-
\$estilos	-
\$js	-
\$identificadores	array
\$componentes	array
Responsabilidades:	
Nombre:	__set
Descripción:	Método para asignar valores a los campos privados de la clase. Campo: nombre del atributo privado. Valor: valor que se le va asignar a dicho campo
Nombre:	__get
Descripción:	Método para devolver los valores de los campos privados de la clase. Campo: nombre del atributo privado.
Nombre:	Encabezado()
Descripción:	Forma el código del encabezado de la página que se desea generar.
Nombre:	AreaTitulo()
Descripción:	Forma el código del área del titulo de la página que se desea generar.
Nombre:	AreaOpciones()
Descripción:	Forma el código del área de opciones, es decir, el área donde están los botones Nuevo, Modificar, Eliminar y Salir de la página que se desea generar.
Nombre:	AreaBotones()
Descripción:	Forma el código del área de los botones Aceptar y Cancelar de la página que se desea generar.

Conclusiones:

En este capítulo se mostraron varios artefactos para llevar a cabo el proceso de construcción del sistema. Igualmente se identificaron otras funcionalidades del sistema que lo hacen más factible. Se utilizaron diagramas de clases Web para explicar la lógica del negocio del sistema, se crearon mecanismos de diseño que simplifican el modelado. En este momento, ya se tiene confeccionada completamente la propuesta que trae este trabajo.

CAPITULO IV: IMPLEMENTACIÓN Y PRUEBA

Introducción:

En este capítulo se aborda la parte de implementación donde se muestra las dependencias entre las partes de código del sistema (diagrama de componentes) o la estructura del sistema en ejecución (diagrama de despliegue): los diagramas de componentes se utilizan para modelar la vista de implementación estática de un sistema, mientras que los diagramas de despliegue se utilizan para modelar la vista de despliegue estática. Así como la etapa de prueba para garantizar la calidad del producto desarrollado.

4.1 Diagrama de Despliegue.

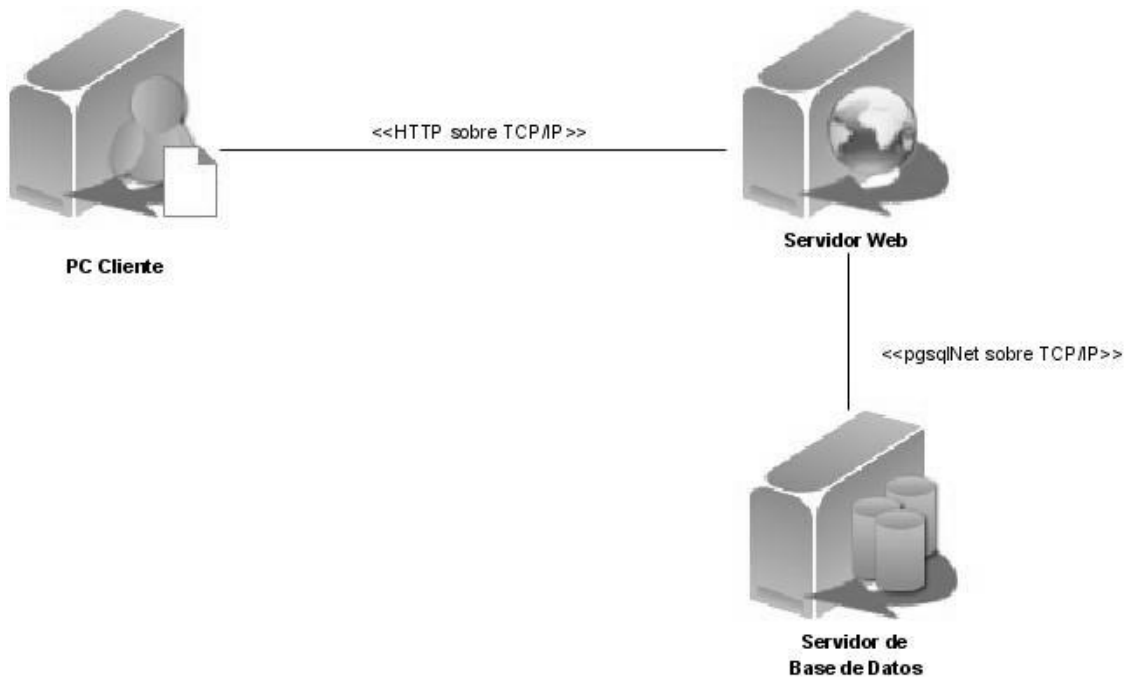
Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos). Estarán formados por instancias de los componentes software que representan manifestaciones del código en tiempo de ejecución (los componentes que sólo sean utilizados en tiempo de compilación deben mostrarse en el diagrama de componentes).

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos, procesos (caso particular de un objeto). En general un nodo será una unidad de computación de algún tipo, desde un sensor a un mainframe. Las instancias de componentes software pueden estar unidas por relaciones de dependencia, posiblemente a interfaces (ya que un componente puede tener más de una interfaz).

Un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento. Pueden representarse instancias o tipos de nodos que se representa como un cubo 3D en los diagramas de implementación.

Las instancias de componentes de software muestran unidades de software en tiempo de ejecución y generalmente ayudan a identificar sus dependencias y su localización en nodos. Pueden mostrar también qué interfaces implementan y qué objetos contienen. Su representación es un rectángulo atravesado por una elipse y dos rectángulos más pequeños. [10]

Diagrama de Despliegue de la Aplicación



4.2 Diagrama de Componentes:

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes de Ada, bibliotecas cargadas dinámicamente, etc. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.

El diagrama de componente hace parte de la vista física de un sistema, la cual modela la estructura de implementación de la aplicación por sí misma, su organización en componentes y su despliegue en nodos de ejecución. Esta vista proporciona la oportunidad de establecer correspondencias entre las clases y los componentes de implementación y nodos. La vista de implementación se representa con los diagramas de componentes.

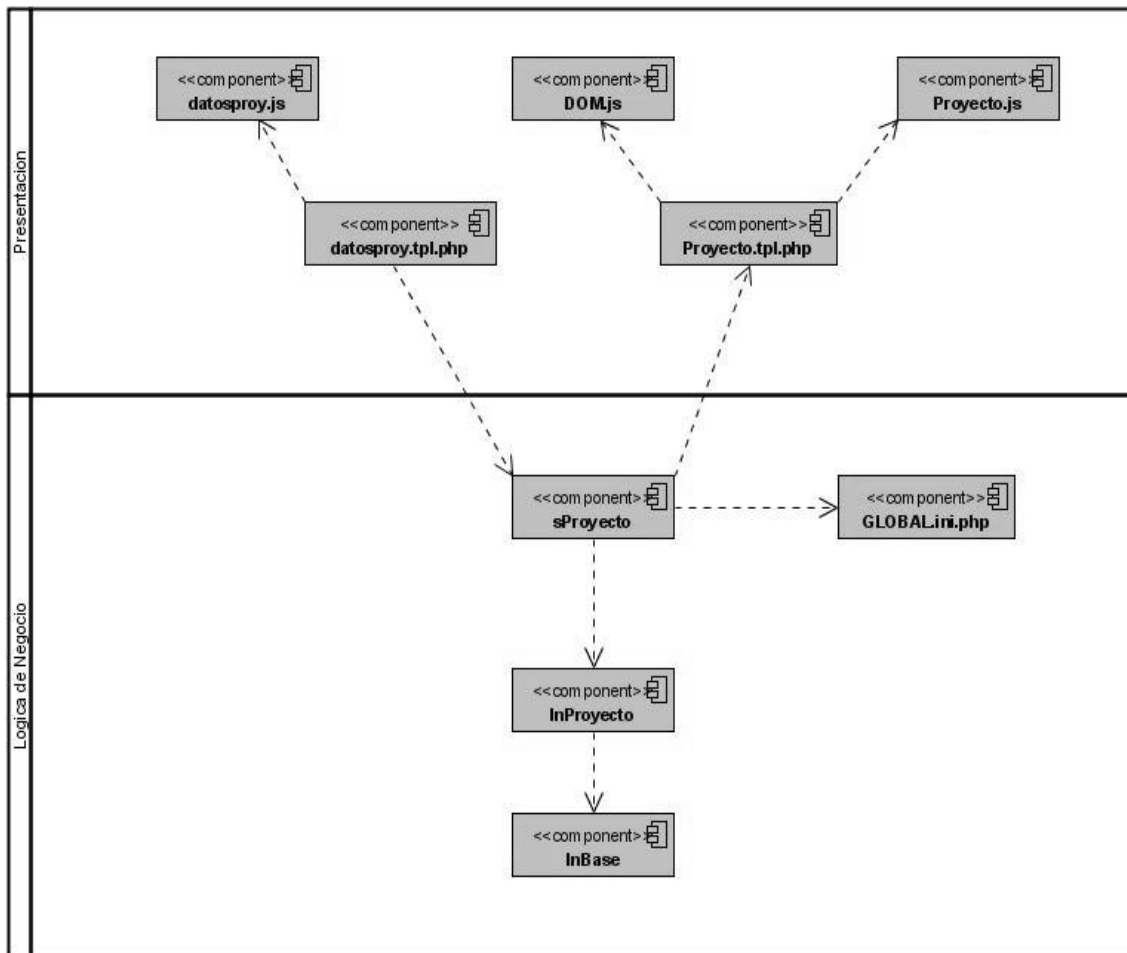
¿Qué es Componente?

Es una parte física reemplazable de un sistema que empaqueta su implementación y es conforme a un conjunto de interfaces a las que proporciona su realización.

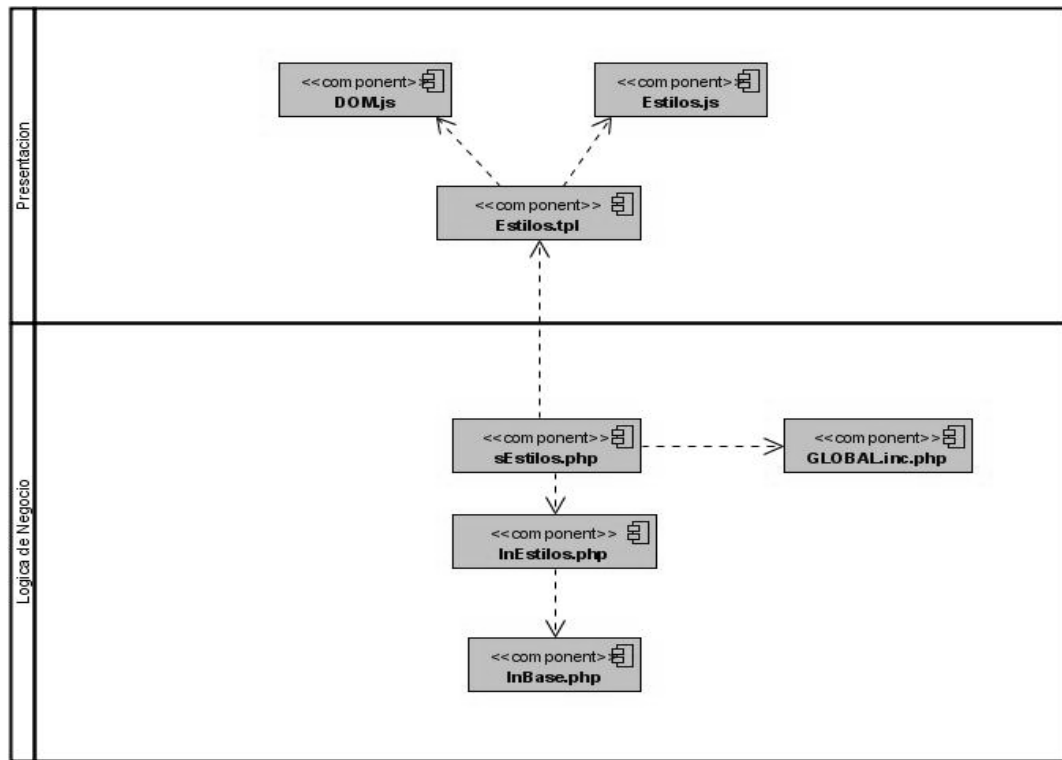
Algunos componentes tienen identidad y pueden poseer entidades físicas, que incluyen objetos en tiempo de ejecución, documentos, bases de datos, etc. Los componentes existentes en el dominio de la implementación son unidades físicas en los computadores que se pueden conectar con otros componentes, sustituir, trasladar, archivar, etc.

Los componentes tienen dos características: Empaquetan el código que implementa la funcionalidad de un sistema, y algunas de sus propias instancias de objetos que constituyen el estado del sistema. Los llamados últimos componentes de la identidad, porque sus instancias poseen identidad y estado. [11]

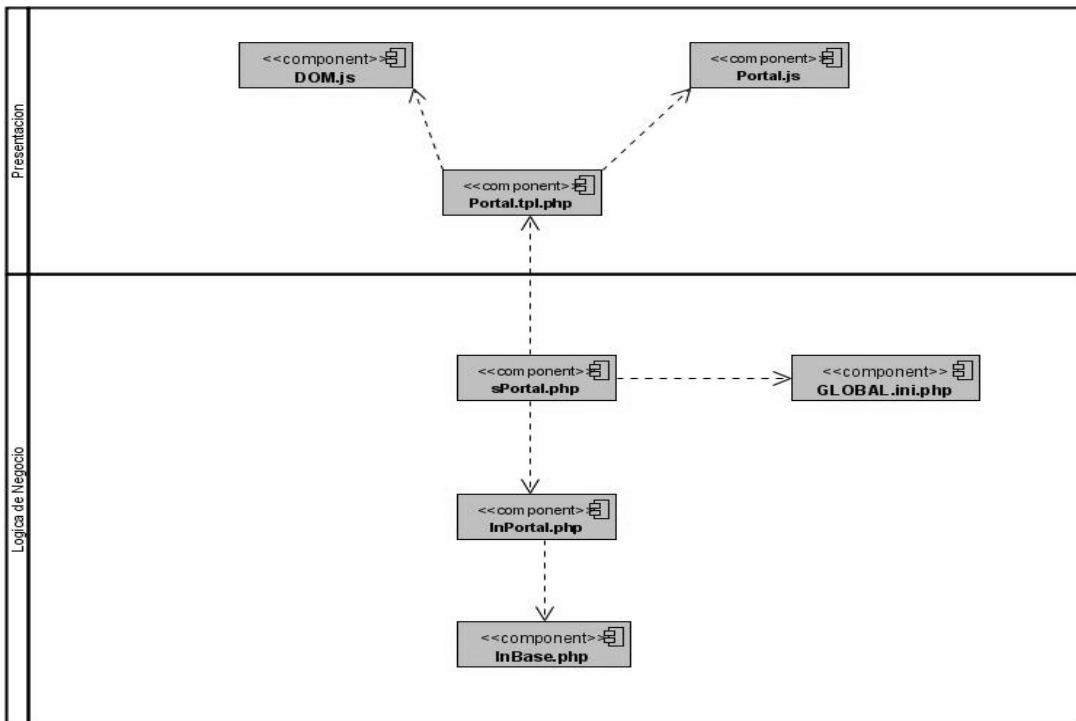
Generar Proyecto:



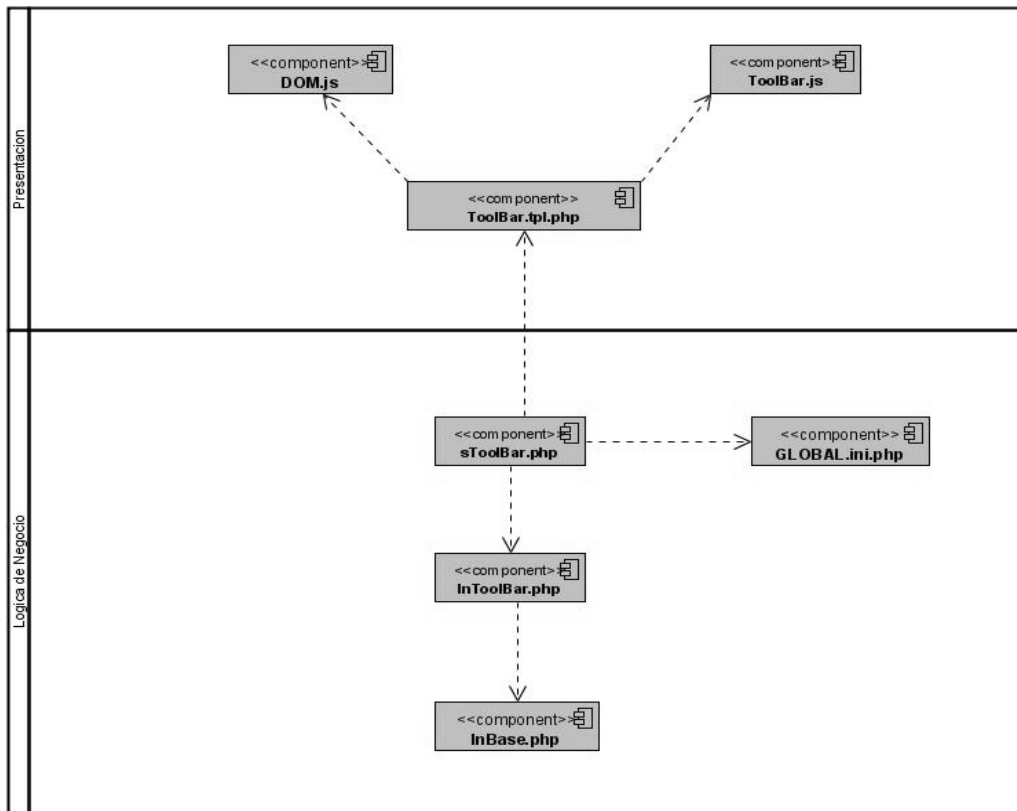
Generar Estilos:



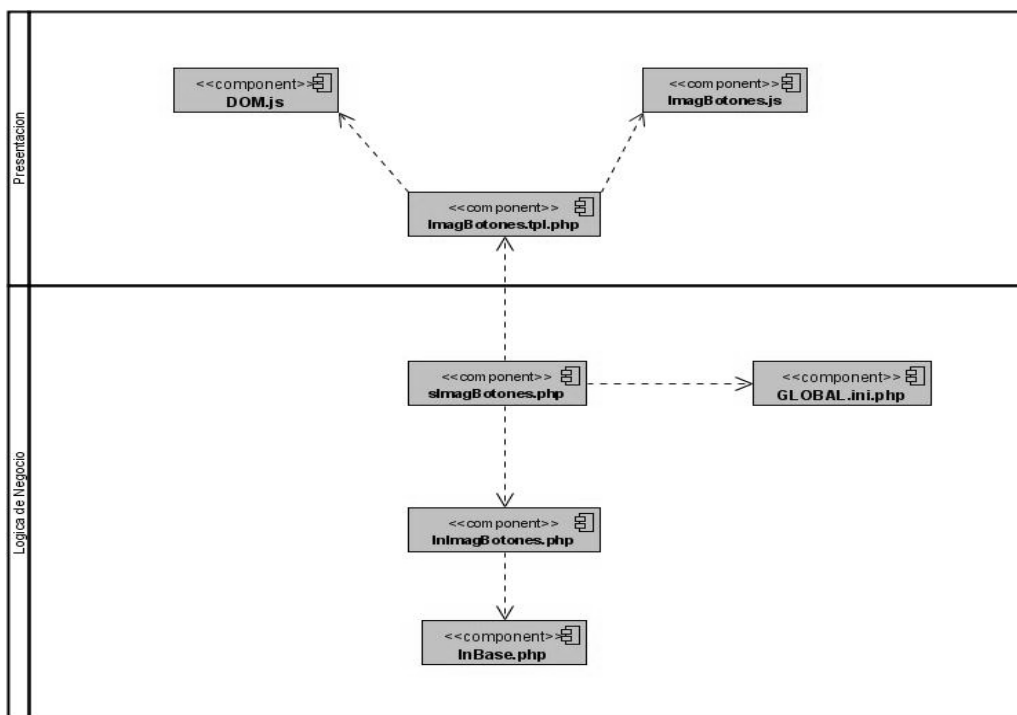
Generar Portal:



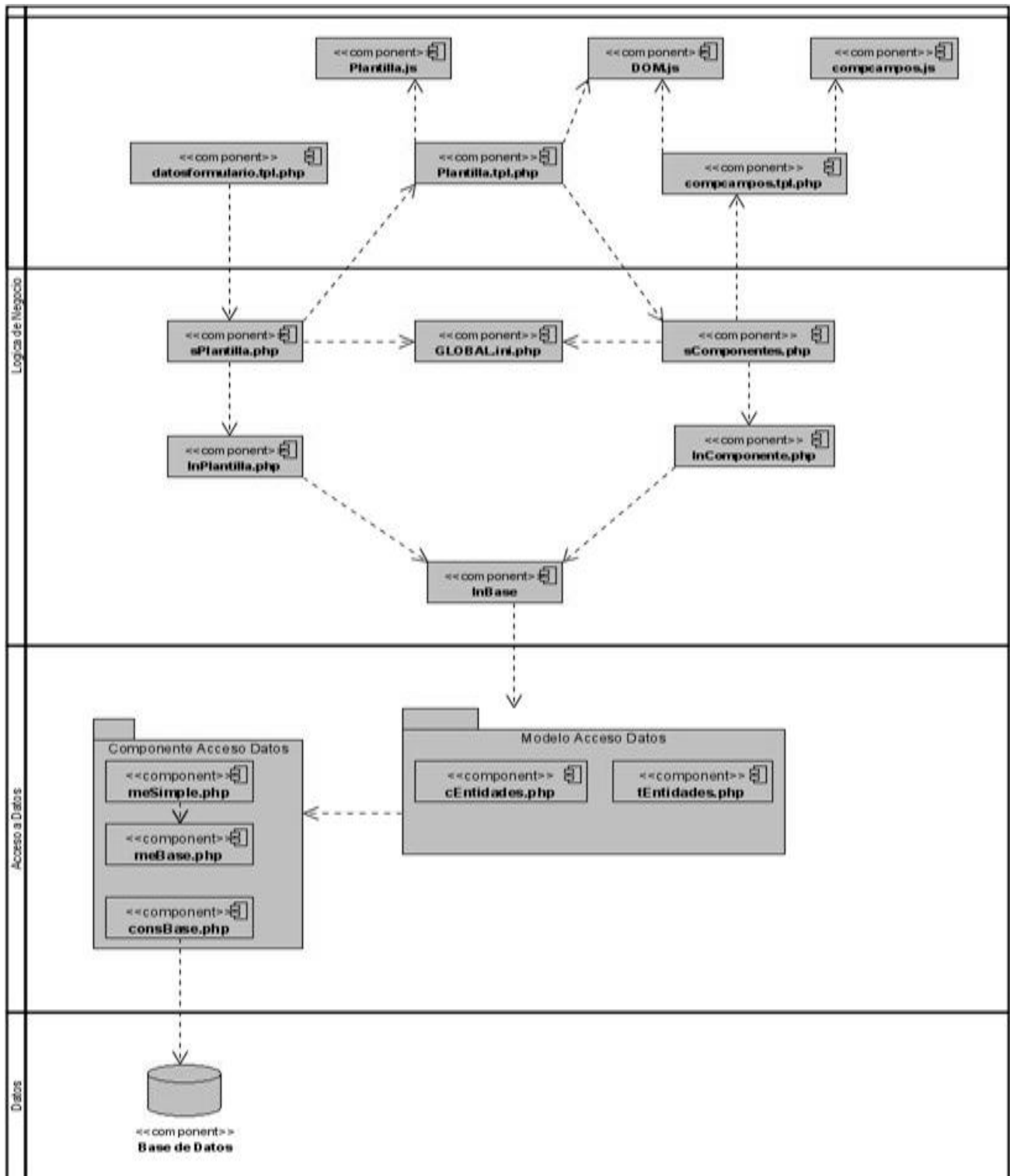
Crear Toobar



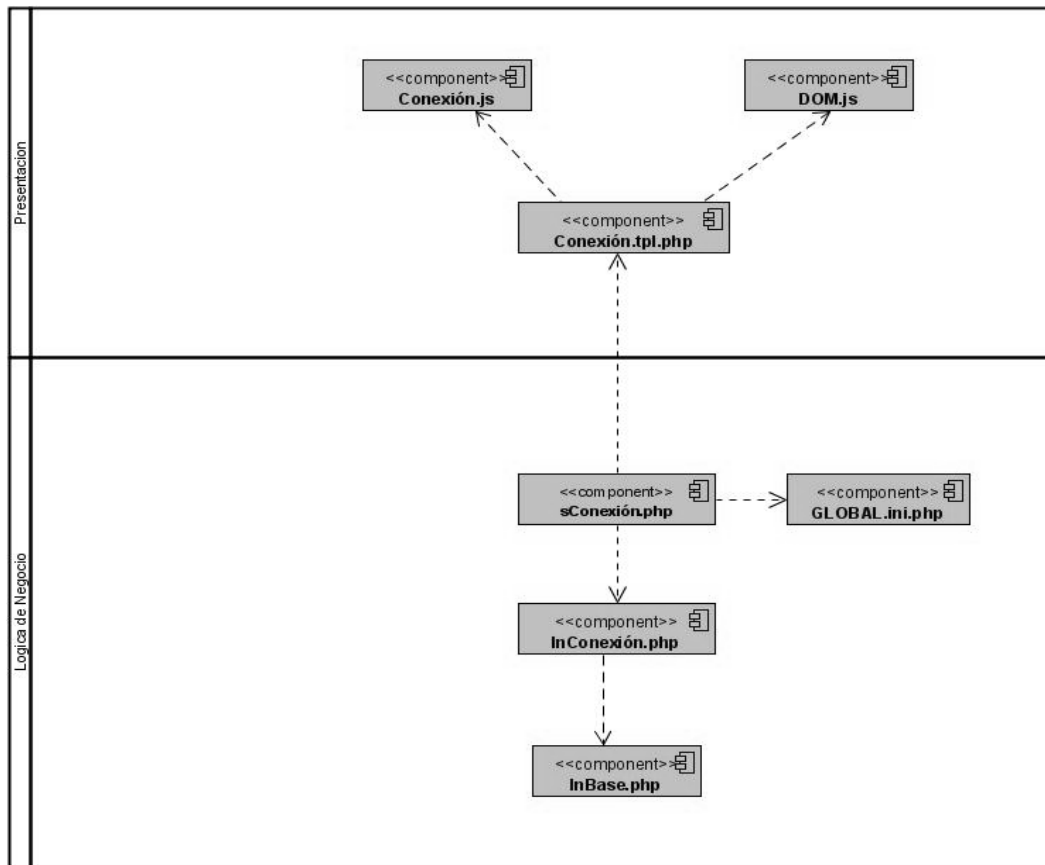
Crear Botones:



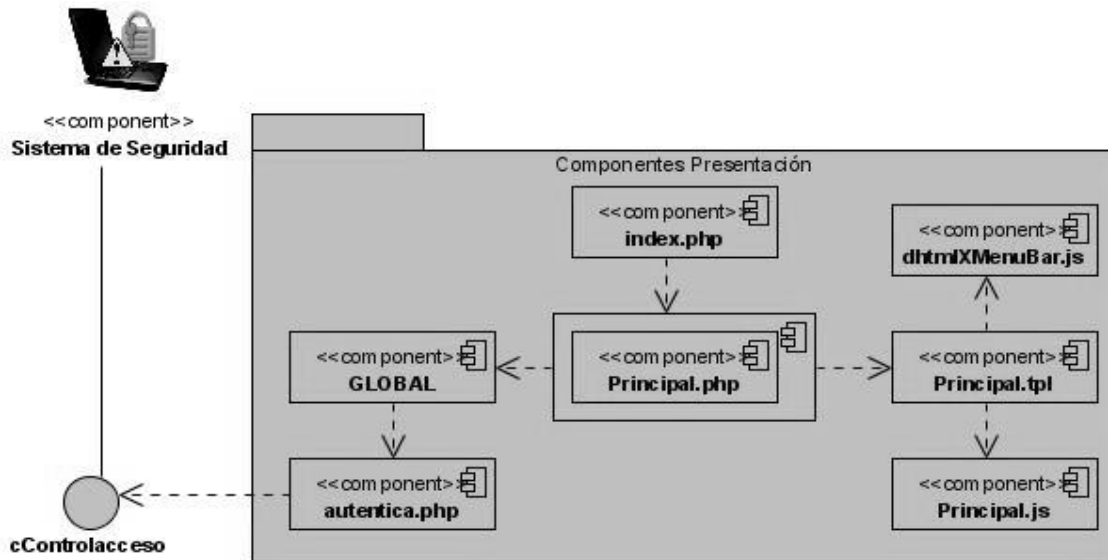
Generar Formulario:



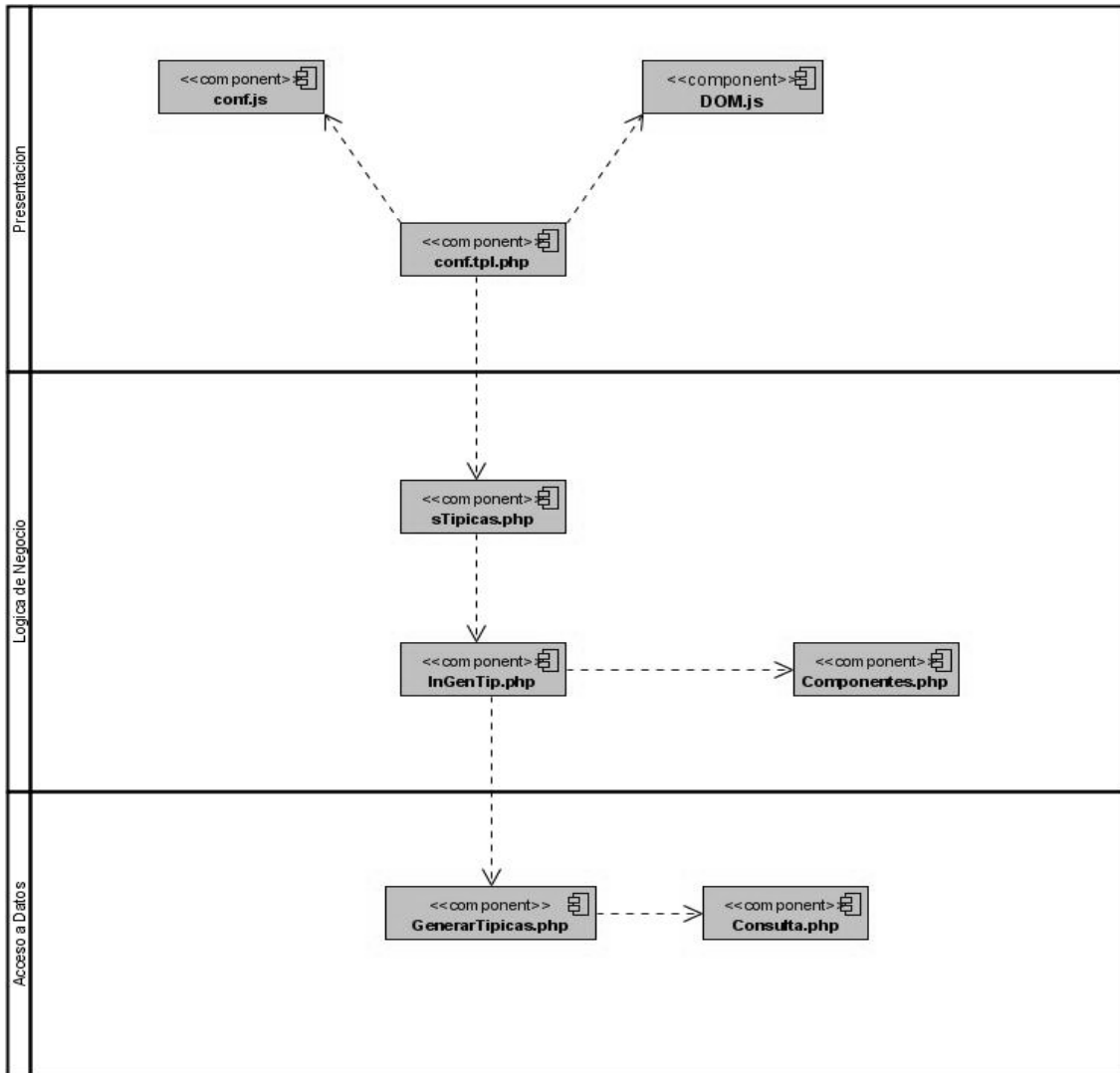
Generar Conexión:



Presentación y Seguridad:



Generar Típicas:



4.3 Prueba:

La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado.

Cualquier proceso de ingeniería puede ser probado de una de dos formas:

- Se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa.
- Se pueden desarrollar pruebas que aseguren que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

La primera aproximación se denomina prueba de la caja negra y la segunda prueba de la caja blanca.

Prueba de caja negra:

Las pruebas se llevan a cabo sobre la interfaz del software, y es completamente indiferente el comportamiento interno y la estructura del programa.

Los casos de prueba de la caja negra pretende demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Se derivan conjuntos de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales del programa.

La prueba de la caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Los casos de prueba deben satisfacer los siguientes criterios:

- Reducir, en un coeficiente que es mayor que uno, el número de casos de prueba adicionales.
- Que digan algo sobre la presencia o ausencia de clases de errores.[12]

Modelo de prueba:

CU Generar Proyecto.		
Entrada	Resultados	Condiciones
	El sistema muestra una interfaz con todos los requisitos que tiene que tener el servidor en el cual se va a montar el proyecto.	
Proyecto: "Prueba" Menú: "C:\Documents and Settings\leevan\Escritorio\Chatarra!!!\ _menu.xml"	El sistema muestra un mensaje de confirmación para crear el proyecto. El sistema muestra una interfaz de que el proyecto fue creado con éxito	

CU Generar Portal.		
Entrada	Resultados	Condiciones
Titulo de la Aplicación: "Prueba" Menu: "C:\Documents and Settings\leevan\Escritorio\Tesis William!!! \Imagenes\software.jpg"	El sistema muestra un mensaje de confirmación para crear el portal.	

CU Generar Estilos.		
Entrada	Resultados	Condiciones
Color de Fondo: "FFFFFF" Color del Marco: "000000"	El sistema muestra un mensaje de confirmación para crear el estilo.	

CU Crear Conexión.		
Entrada	Resultados	Condiciones
Color de Fondo: "FFFFFF" Color del Marco: "000000"	El sistema muestra un mensaje de confirmación para crear el estilo.	

CU Generar Formulario.		
Entrada	Resultados	Condiciones
Nombre: "Prueba" Titulo: "Prueba"	El sistema muestra un mensaje de confirmación para continuar introduciendo datos.	
Esquemas de la Aplicación: "material" Tablas: "nom_prod" Campos: "actual, precio"	El sistema muestra un mensaje de confirmación para continuar introduciendo datos.	
Asocio a cada campo un componente: "Actual->Texto" "Precio->Texto" Selecciono que se va a usar un gris.	El sistema muestra un mensaje de confirmación para generar el formulario.	

Conclusiones:

Al finalizar este capitulo, se tiene la distribución física de la aplicación, junto a una serie de pruebas para comprobar la calidad del sistema. Al finalizar estas fases se tiene una noción completa del software.

CONCLUSIONES:

Luego de un estudio realizado, unido al diseño e implementación de una primera versión del sistema, se obtuvo resultados bastante buenos, pues se logró dar una solución elegante y eficiente a la problemática existente en los proyectos a la hora de lograr una mayor eficiencia en la programación. Derivado de esto y después de finalizado todo un proceso de desarrollo de software se puede llegar a las conclusiones siguientes: Valorados los impactos causados por el proceso de desarrollo e implantación del Generador de Código, se puede afirmar que el producto es sostenible, mejorándose grandemente la calidad de la producción de software, logrando el establecimiento de estándares de código.

Se puede plantear en este momento, que concluido este trabajo de diploma y desarrollado los temas que en él se exponen y que describen el proceso de análisis, modelación, elaboración y producción de un sistema capaz de generar código; se ha cumplido la hipótesis planteada en el mismo, obteniendo resultados prometedores y relevantes en un campo de investigación donde el país aun se encuentra en una fase inicial, o casi nula.

El desarrollo e implantación de este sistema en diferentes proyectos no solo a nivel de ERPFAR, sino también a nivel nacional podría convertir la producción de software en algo más prometedor, pues la fase de construcción de los proyectos disminuiría considerablemente, trayendo consigo llevar a Cuba a un estadio superior en la producción de software.

RECOMENDACIONES

Esta aplicación es solo un comienzo a un tema, el cual puede convertirse en un IDE de desarrollo por su gran aplicación y utilidad a la hora de programar. Se le deben mejorar algunos módulos como el de creación de interfaz, el cual puede abarcar diferentes modelos de interfaz, ampliándose la cantidad y calidad de componentes. Además de hacerse una investigación más minuciosa acerca de cómo generar las clases de Lógica de Negocio, para ahorrarle aun más el trabajo al programador.

En el mundo existen algunos grupos de desarrollo que se están dedicando al tema, y cada vez se optimiza mas la forma de generar código, por lo que con respecto a esto se debe seguir investigando y observando variantes que surgen y salen al mercado, para poder mejorar esta aplicación llegándola hasta integrar a otras, haciendo mas fácil su manejo y fiabilidad.

Otra variante que se puede seguir investigando es acerca de orientar esta aplicación a SOA, como aplicación orientada a servicios para integrarla a cualquier aplicación y hasta utilizar servicios que la misma pueda brindar, logrando una mayor extensibilidad, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato, así como la extensión de este generador a cualquier lenguaje de programación y no solo centrarlo a un único lenguaje de desarrollo.

La implementación de un Modulo de Generación XMI, ayudará mucho en el trabajo de los ingenieros de software, siendo un pilar fundamental en cualquier proyecto, pero para implementar este modulo se debe investigar en la estructura de los XMI para poder generarlos. Su estudio podría resolver grandes problemas en el Análisis y Diseño de los proyectos. Con su desarrollo e implantación el software no solo ahorraría tiempo a los programadores sino a los diseñadores también, ampliando su campo de acción en el proyecto y siendo de gran utilidad en cualquier ámbito de desarrollo.

El diseño e implementación de la modificación de la arquitectura y la creación de un repositorio de componentes de trabajo le daría un acabado casi total a la aplicación, haciéndola extensible, duradera y eficaz, pues se estaría hablando entonces de un software con el cual, a pesar de los cambios realizados y las nuevas propuestas seguiría tan vigente como hasta el momento.

BIBLIOGRAFIA**Citadas:**

- [1] COMPILADORES Y GENERADORES DE CÓDIGO.
<http://www.mailxmail.com/curso/informatica/generadores/capitulo2.htm> (25/04/2007)
- [2] CODECHARGE STUDIO. http://www.yessoftware.com/products/product_detail.php?product_id=1
(10/05/2007)
- [3] MAKE ME FEEL GOOD. <http://www.abcdatos.com/webmasters/programa/z4687.html> (10/05/2007)
- [4] Clarion. <http://www.gopac.com.mx/herramientas/clarion/descripcion.htm> (10/05/2007).
- [5] Rational Unified Process.
http://www.itera.com.mx//index.php?option=com_content&task=view&id=18&Itemid=42 (28/05/2007).
- [6] Modelado de Sistemas con UML.
<http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/>(1/06/2007)
- [7] Conceptos básicos de Dreamweaver 8. http://www.aulaclie.es/dreamweaver8/t_1_1.htm (15/05/2007).
- [8] Diseño. <http://lsi.ugr.es/~arroyo/inndoc/cicloVidaSoft.html>(1/06/2007)
- [9] Modelado de Sistemas con UML Capítulo 4.
<http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x194.html>. (1/06/2007)
- [10] Ana Fernandez Vilas. Diagrama de Despliegue.
<http://www-gris.det.uvigo.es/~avilas/UML/node50.html>(20/04/2007)
- [11] Diagramas de Componentes. <http://www.creangel.com/uml/componente.php>. (20/04/2007)
- [12] Prueba. http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php (1/06/2007)

Consultadas:

- Pressman, R. *Software Engineering. A Practitioner's Approach*. Fourth Edition. McGraw – Hill. USA, 1999.
- Booch, G., Rumbaugh, J., Jacobson, I. *El Lenguaje Unificado de Modelado*. Addison-Wesley. 1999.
- Larman, C. *UML Y PATRONES, Introducción al análisis y diseño orientado a objetos*. La Habana. Cuba 2004.
- Booch, G., Rumbaugh, J., Jacobson, I. *El Proceso Unificado de Desarrollo de Software*. La Habana. Cuba 2004.

GLOSARIO

Arquitectura: Consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. La arquitectura software establece los fundamentos para que analistas, diseñadores y programadores trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema de información.

Patrón: Es una solución a un problema no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas en distintas circunstancias).

PHP: Es un lenguaje de programación usado frecuentemente para la creación de contenido para sitios web con los cuales se puede programar las páginas HTML y los códigos de fuente. PHP es un acrónimo recursivo que significa "**PHP Hypertext Pre-processor**". Se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios web.

HTML: Acrónimo inglés de **HyperText Markup Language** (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Internet Explorer, Opera, Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos y también de los más fáciles de aprender.

CSS: Referencia al lenguaje de hojas de estilo en cascada (Cascading Style Sheets). Es utilizado para definir la presentación de un documento HTML o XML.

RUP: El Rational Unified Process (RUP) es una metodología formal, a veces también llamada proceso. El RUP describe a gran detalle todas las actividades, roles, responsabilidades, productos de trabajo y herramientas para definir quién hace qué y en qué momento en un proyecto de desarrollo de software

TopSpeed: Traducido del Inglés al Español, se refiere a Velocidad superior.

UML: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Herramientas CASE: Las Herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Autenticación: (Griego: *αυθεντικός* = verdadero o genuino, de ' los authentes' = el autor) es el acto de establecimiento o confirmación de algo (o alguien) como auténtico, es decir que reclama hecho por o sobre la cosa son verdadero. La autenticación de un objeto puede significar (pensar) la confirmación de su procedencia, mientras que la autenticación de una persona a menudo consiste en verificar su identidad. La autenticación depende de uno o varios factores de autenticación.

Auditoria: Consiste principalmente en estudiar los mecanismos de control que están implantados en una empresa, organización o software, determinando si los mismos son adecuados y cumplen unos determinados objetivos o estrategias, estableciendo los cambios que se deberían realizar para la consecución de los mismos.

XML: Sigla en inglés de eXtensible Markup Language («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

XMI: XML Metadata Interchange (XML de Intercambio de Metadatos) es una especificación para el Intercambio de Diagramas. La especificación para el intercambio de diagramas fue escrita para proveer una manera de compartir modelos UML entre diferentes herramientas de modelado.

PDO: Siglas de PHP Data Objects (objetos de Datos PHP) es una extensión de acceso a bases de datos para PHP5. Esta extensión define una interfaz ligera y consistente para acceder a bases de datos en PHP. Cada driver de bases de datos que implementa la interfaz PDO puede exponer características específicas de la base de datos como funciones de extensión regulares.

API: Del inglés Application Programming Interface (Interfaz de Programación de Aplicaciones) es del conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

Arquitectura MVC: Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el controlador es el Sistema de Gestión de Base de Datos y el modelo es el modelo de datos.

SOAP: Siglas de *Simple Object Access Protocol* es un protocolo estándar creado por Microsoft, IBM que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

Heurísticas: Se denomina heurística a la capacidad de un sistema para realizar de forma inmediata innovaciones positivas para sus fines.

PDOStatement: Término usado para referirse a las declaraciones y funciones de PDO.

JavaScript: Es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas web.

Smarty: Es un motor de plantillas para PHP, cuyo objetivo es separar el contenido de la presentación en una página web, se encuentra bajo la licencia LGPL por lo que puede ser usado libremente.

Paquetes de Ada: es un lenguaje de programación estructurado y fuertemente tipado de forma estática que fue diseñado por Jean Ichbiah de CII Honeywell Bull por encargo del Departamento de Defensa de los Estados Unidos. Es un lenguaje multipropósito, orientado a objetos y concurrente, pudiendo llegar desde la facilidad de Pascal hasta la flexibilidad de C++.

ASP: Active Server Pages (ASP) es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente.

JSP: JavaServer Pages (JSP), en el campo de la informática, es una tecnología para crear aplicaciones web.

MS-SQL: Microsoft SQL Server, gestor de base de Datos creado y comercializado por la Compañía Microsoft.

ODBC: Son las siglas de **Open DataBase Connectivity**, que es un estándar de acceso a Bases de Datos desarrollado por Microsoft Corporation, el objetivo de ODBC es hacer posible el acceder a cualquier dato de cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos.

SQL: Lenguaje de Consulta Estructurado (**Structured Query Language**) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos.

MySQL: Es un sistema de gestión de base de datos relacional, multihilo y multiusuario como software libre en un esquema de licenciamiento dual.

Coldfusion: Servidor de páginas webs de la casa de Macromedia (Actualmente Adobe) que genera de manera rápida contenido dinámico por medio de tags especiales embebidos en código HTML.