Universidad de las Ciencias Informáticas

Facultad 9

TÍTULO: "PROPUESTA DE ARQUITECTURA PARA EL SISTEMA DE GESTIÓN AUTOMATIZADO DE RECURSOS HUMANOS GESTAPRO".

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Luis Díaz Vallejo

Tutor: Yoandrys González González

Cotutor: Arnaldo Gandol Álvarez

Consultante: Carlos Enrique Hernández Reyes

Asesor: Guillermo Báez Ramos

Ciudad de La Habana, 2009.

"Año del 50 Aniversario del Triunfo de la Revolución."

DEDICATORIA

Este trabajo va dedicado principalmente a mis padres por su apoyo incondicional desde los primeros años de mi vida, a todas aquellas personas que no pudieron estar junto a mí en esta etapa de mi vida porque les toco finalizar la de ellos antes de tiempo, a aquellos que viven y dieron todo porque fuera lo que soy, a ti mi abuela Maria Luisa por tu ejemplo y dedicación, mi abuelo Panchón por darme tu ejemplo y voluntad de lucha por lo que uno aspira, a ti mi hermana del alma por quererme y hacerme saber que somos uno. No quisiera terminar sin hacer referencia a la revolución Cubana, al Che y a nuestro Comandante a todos ustedes va dedicado este trabajo.

AGRADECIMIENTOS

Agradecer quisiera a mi familia, mis padres Orlando y Luisa, a mi hermana Lisset por fomentar el ánimo que me faltó en algunos momentos, a mi novia Ailyn por darme su apoyo y amor para poder terminar esta difícil tarea. Agradecer a todas mis amistades y amigos por su apoyo, a la Universidad de las Ciencias Informáticas por darme la posibilidad de formarme en ella y a nuestro Comandante en Jefe Fidel por tener la brillante idea de crear una carrera en una universidad de este tipo.

<En este acápite se incluye la opinión del tutor del trabajo de diploma>

El presente trabajo está encaminado a informatizar el proceso de gestión de los recursos humanos en los polos productivos de la Facultad 9 de La Universidad de las Ciencias Informáticas (UCI). Por ello se realiza la propuesta de llevar a cabo una arquitectura de software para poder lograr una buena automatización del proceso en gestión.

La UCI está encargada del estudio y la producción de bienes y servicios informáticos. Para ello cuenta con polos productivos destinados al estudio y desarrollo de software así como prestación de servicios informáticos. Por tanto es de suma importancia una correcta gestión de todo el personal inmerso en dicho proceso. Para ello hoy en día ya existe un mecanismo diseñado por la propia institución, este proceso se realiza de forma manual de ahí que surgiera la necesidad de implementar una aplicación para automatizar el proceso.

El presente trabajo de diploma contiene un estudio de los principales elementos que constituyen la arquitectura de software, durante su desarrollo se realiza un análisis de dichos elementos con el objetivo de lograr una organización estructural del sistema para la gestión de los recursos humanos en los polos productivos de la Facultad 9 de La Universidad de las Ciencias Informáticas, expresada en la relación entre sus componentes, conectores y restricciones.
De las decisiones arquitectónicas tomadas durante todo el período de desarrollo, definiendo las características fundamentales de las tecnologías y aspectos esenciales de diseño se provee a los miembros del equipo de una idea clara y precisa del resultado que se espera lograr una vez finalizado el proceso de desarrollo del mismo.

PALABRAS CLAVES

- Gestión de Recursos Humanos.
- Sistemas de Gestión de los Recursos Humanos.
- Arquitectura de Software

Introducción:

Las agresiones inescrupulosas del gobierno de Estados Unidos hacia nuestro país, ha provocado que hoy se cuente con un atraso considerable en cuanto a temas de desarrollo de software se refiere. El gobierno cubano viene tomando medidas para insertar al país dentro de un potente mercado como es el de la informática y las comunicaciones. Nuestro país se ha esforzado en informatizar la sociedad, esto ha traído consigo que se hayan tenido que crear disímiles centros donde se desarrollen sistemas informáticos. Ejemplo de estos centros son: Los Centros de Desarrollo de Software en las provincias de Villa Clara, Santiago de Cuba, entre otros.

Dentro de las medidas que se vienen tomando por el Gobierno Cubano para llegar a tener una población más informatizada; se encuentran la creación de Los Joven Club de informática y electrónica, los Politécnicos de Informática y la Universidad de las Ciencias Informáticas (UCI). Esta última creada en el seno de la batalla de ideas que libra el pueblo cubano. La UCI cuenta con estudiantes de todos los municipios del país los cuales dedican todos sus esfuerzos al estudio y desarrollo de la informática. Esta ya cuenta con más de 10 000 estudiantes distribuidos en 13 facultades. Dichas facultades con polos productivos en los cuales se realizan proyectos y servicios informáticos, en la que se encuentran inmiscuidos más del 80% de sus estudiantes. En la Facultad 9 se encuentran funcionando cinco polos productivos, Video y Sonido Digital, PICG, Simulación de Procesos Industriales, Calidad de Software y PetroSoft en los cuales se encuentran inmersos un 82% de los estudiantes de esta Facultad en tareas de desarrollo e investigación de productos y servicios informáticos.

El proceso de control, evaluación y asignación de tareas dentro de los polos productivos se lleva de manera manual. Los líderes de los polos y proyectos realizan este proceso utilizando Microsoft Exel. Donde los estudiantes que se le asignan al polo productivo llegan hasta estas personas y estos tienen que tomar los datos personales y adicionarlos al documento que se hace mención. Esto trae una problemática, que realizar una búsqueda de un estudiante o la actualización de un dato sea algo tediosa ya que tendría que buscar en todo el documento hasta encontrar la persona que se busca. Todo este

proceso se encuentra descentralizado ya que a menudo esta información se encuentra en más de un documento.

De hacerse necesario automatizar el proceso de asignación y control de los estudiantes y profesores que están inmersos en procesos de estudio y desarrollo de sistemas informáticos en la Facultad 9 surge la idea de desarrollar un sistema para la gestión de los recursos humanos de dichos polos productivos.

¿Cómo lograr que los desarrolladores del sistema para la gestión de los recursos humanos en los polos productivos de la Facultad 9 tengan una idea clara de lo que desarrollarán?

La arquitectura de software involucra los elementos más significativos del sistema y está influenciada por las plataformas, sistemas operativos, manejadores de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales. Es una radiografía del sistema que se está desarrollando, lo suficientemente completa como para que todos los implicados en el desarrollo tengan una idea clara de qué están construyendo, pero lo suficientemente simple como para que si se suprime algún elemento, una parte importante del sistema quede sin especificar.

De forma solapada los desarrolladores de software han ido desarrollando la arquitectura de software. Por mera lógica buscaron la forma de reutilizar funciones o heredarlas de sistemas anteriores. La utilización y búsqueda de solución a problemas sería la primera utilización de la arquitectura de software que se conoce, aunque no se escribieran de la forma en la que hoy es conocida.

Es evidente la necesidad de la representación arquitectónica en los sistemas de software, en primer lugar las representaciones arquitectónicas elevan el nivel de abstracción, facilitando la comprensión de sistemas complejos. En segundo lugar hace que aumente la posibilidad de reutilizar los distintos componentes o elementos de software y que se pueda reutilizar hasta la misma arquitectura.

Por todo lo antes expuesto y las necesidades de la Facultad 9 de la Universidad de las Ciencias Informáticas se plantea como:

Problema a resolver: Inexistencia de una arquitectura para el proceso de desarrollo del sistema de gestión de los recursos humanos en los polos productivos de la Facultad 9.

Objetivo General: Proponer una arquitectura para el sistema de gestión automatizado de recursos humanos en los polos productivos de la Facultad 9.

Objeto de estudio: El proceso de diseño de arquitectura de software.

Campo de acción: El proceso y descripción de arquitecturas de software para aplicaciones Web.

La **Idea a Defender** en el presente trabajo plantea que si se tiene un dominio de la tecnología y la metodología a utilizar a la hora de desarrollar la arquitectura y se hace un correcto análisis de los requisitos, se puede obtener una arquitectura que pueda ser reusable, flexible y que además cumpla con las funcionalidades requeridas por los clientes del software para gestión de los recursos humanos de los polos productivos que se pretende desarrollar en la Facultad 9.

Las **Tareas** que se llevarán a cabo para dar cumplimiento al objetivo propuesto se enumeran a continuación.

1. Análisis del estado en que se encuentra la gestión de los recursos humanos en los polos productivos de la Facultad 9.
2. Realización de un estudio de las diferentes arquitecturas existentes para establecer una comparación y seleccionar la adecuada a aplicar en el desarrollo del software que se pretende implementar.
3. Definición de la estrategia de desarrollo, con el objetivo de elegir y aplicar la más adecuada de acuerdo al tipo de sistema.
4. Identificación de los patrones arquitectónicos.
5. Fundamentación de dichos patrones y sus aplicaciones.
6. Definición de las herramientas a ser utilizadas en el proceso de desarrollo.
7. Descripción de la alternativa propuesta.
8. Evaluación de la arquitectura propuesta.

Dentro de los grupos de Métodos que se llevarán a cabo para la realización de las tareas antes planteada se proponen los siguientes:

Métodos Teóricos:

Análisis histórico lógico: Para determinar las tendencias en el desarrollo de modelos y enfoques arquitectónicos existentes en el mundo y en especial en Cuba.

Analítico-Sintético:

Para el procesamiento de la información y poder arribar a las conclusiones de la investigación y precisar las características del modelo arquitectónico. Para poder seleccionar y describir los requerimientos con que debe constar la aplicación que fueron expuestos por los jefes de polos de la facultad 9 y el vice decano de producción.

Métodos Empíricos:

Entrevistas individuales y colectivas:

Para poder comprender mejor la situación del problema, así como las opciones y sugerencias de los clientes.

Observación: Para la representación de las restricciones y propiedades del sistema.

Sistémica: Para el control de la evolución de la arquitectura propuesta.

Muestreo y estimación:

Población:

Se utilizará en la investigación los cinco líderes de los polos productivos de la Facultad 9 y cinco compañeros que cuenten con los conocimientos suficientes en el tema de arquitectura de software.

Muestra:

Se seleccionaron los cinco líderes de polos productivos de la facultad 9, seleccionando aquellos con mayor experiencia en el tema, así como cinco compañeros con conocimientos de arquitectura de software que puedan aportar sus ideas para una mejor evolución de la alternativa que se planteará.

Técnica de muestreo:

Criterio muestral:

La muestra seleccionada es intencional debido a que su fundamento consiste en escoger los integrantes de la muestra, por lo que el investigador selecciona explícitamente los elementos que son representativos o con posibilidades de brindar mayor información.

Posible Resultado:

Una arquitectura completa para ser utilizada en el desarrollo del software de gestión de los recursos humanos en los polos productivos de la Facultad 9.

Capítulo #1: Fundamentación Teórica

Introducción:

En el presente capítulo se dará una panorámica de las metodologías de desarrollo de software para seleccionar la más adecuada a aplicar en el proceso de desarrollo del software. Se abordará el tema de arquitectura de software, tipos de arquitecturas existentes así como la caracterización de dichas arquitecturas.

Para abordar el tema de arquitectura de software es necesario conocer el concepto de Arquitecto de software así como su importancia en el proceso de desarrollo. También la caracterización de los diferentes patrones arquitectónicos y su importancia dentro de la arquitectura. Se hará referencia a un grupo de herramientas que se utilizarán en el proceso de desarrollo del software de gestión que se quiere desarrollar en la Facultad 9.

1.1 Objeto de estudio:

1.1.1 Descripción del objeto de estudio:

Desde la creación de los primeros sistemas de cómputo el hombre ha venido pensando en el desarrollo de bienes y servicios informáticos con el fin de facilitarse el trabajo. En los inicios del desarrollo de la informática los desarrolladores se centraban en aspectos como la programación y el análisis, se hablaba además de una arquitectura muy precaria. Sin embargo uno de los componentes dentro del proceso de desarrollo que ayuda a mejorar y agilizar el desarrollo de medios informáticos es la arquitectura de software. La arquitectura brinda una visión a gran escala del sistema y la relación entre los fragmentos del mismo. De este término se viene discutiendo desde hace tiempo y ya existen disímiles estilos y patrones que han ido modificándose y reestructurándose a medida del incremento de las necesidades y expectativas de las empresas productoras de software. Define Roger S. Pressman “La arquitectura no es el software operacional, más bien es la representación que capacita al ingeniero del software para: analizar la efectividad del diseño para la consecuencia de los requisitos fijados, considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el diseño es relativamente fácil y reducir los riesgos asociados a la construcción del software”. [1]

Para un desarrollo correcto de la arquitectura de software es muy importante tener en cuenta el ambiente técnico, el cual va a estar dado por la experiencia que tenga el arquitecto en las tecnologías y en otros proyectos realizados. La arquitectura es un diseño de alto nivel del software, esta se relaciona con los casos de usos y la conexión de los elementos que forman el sistema.

Las metodologías de desarrollo de software centran su flujo de acciones en la arquitectura de software ya que el arquitecto de software es el encargado de seleccionar la metodología a seguir dentro del proyecto. Las metodologías de software guían todo el proceso de desarrollo y definen un grupo de pasos a seguir y todas centran su importancia en la arquitectura de software.

1.2 Metodologías de desarrollo de software:

Las metodologías de desarrollo de software sirven como plano para los implicados en el proceso de desarrollo. De no seleccionarse correctamente una metodología traería consigo el incumplimiento del plan de entrega del producto, insatisfacción con los clientes y un grupo de desarrolladores totalmente a la deriva. De ahí que la principal preocupación de las empresas desarrolladoras de software sea: ¿Cuál es el proceso o metodología más adecuada para llevar a cabo en la empresa?

A continuación se describen tres de las metodologías existentes para el desarrollo de software: Rational Unifield Process (RUP), Extremer Programming (XP), Microsoft Solution Framework (MSF).

1.2.1 Rational Unifield Process (RUP)

Esta metodología es aplicable a proyectos a largo plazo aunque no deja de ser adaptable para proyectos a menor escala. Presenta las siguientes características.

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios necesitan y desean. A partir de aquí los casos de uso guían el proceso de desarrollo.

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben de estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción.

Iterativo e incremental: Es el proceso de construir sistemas haciendo aproximaciones que se acercan progresivamente a la solución ideal. De esta forma se obliga a identificar los riesgos del proyecto en etapas tempranas y es además un enfoque de descubrimiento, invención e implementación continuos donde cada iteración termina en una forma predecible y repetible.

Esta metodología está dividida en cuatro fases de desarrollo: Inicio, Elaboración, Construcción y Transición y cuenta con nueve flujos de trabajo.

Modelamiento del negocio: Describe los procesos del negocio identificando quienes participan en él y las actividades que requieren automatización.

Requerimientos: Definen que es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y restricciones que se imponen.

Análisis y diseño: Describe como el sistema será realizado a partir de las funcionalidades previstas y las restricciones impuestas.

Implementación: Define como se organizan las clases y objetos en componentes cuales nodos se utilizarán así como la ubicación en ellos de los componentes.

Prueba: Busca defectos a lo largo del ciclo de vida.

Instalación: Produce versiones del producto y realiza actividades para entregar el software a los usuarios finales.

Administración del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

Administración de configuración y cambio: Describe cómo controlar los elementos producidos por todos los integrantes del equipo.

Ambiente: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto.

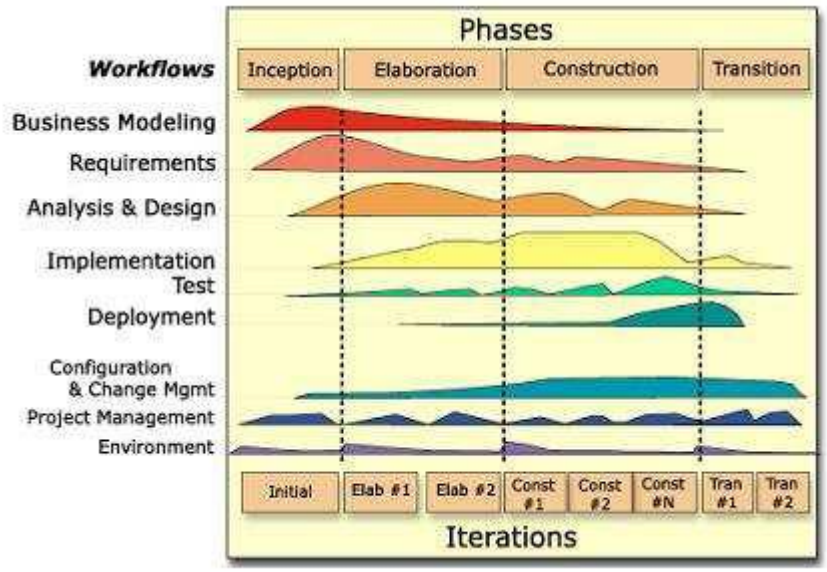

Figura 1 Las fases de desarrollo y los flujos de trabajos de RUP.

1.2.2 Extremer Programming (XP)

Esta metodología es muy exitosa cuando se aplica a proyectos a corto plazo y consiste en una programación muy rápida. La metodología XP no es aplicable a grandes proyectos, esta como particularidad incluye al cliente como parte del equipo de desarrollo. Está compuesta por cuatro fases: planificación, diseño, codificación y prueba.

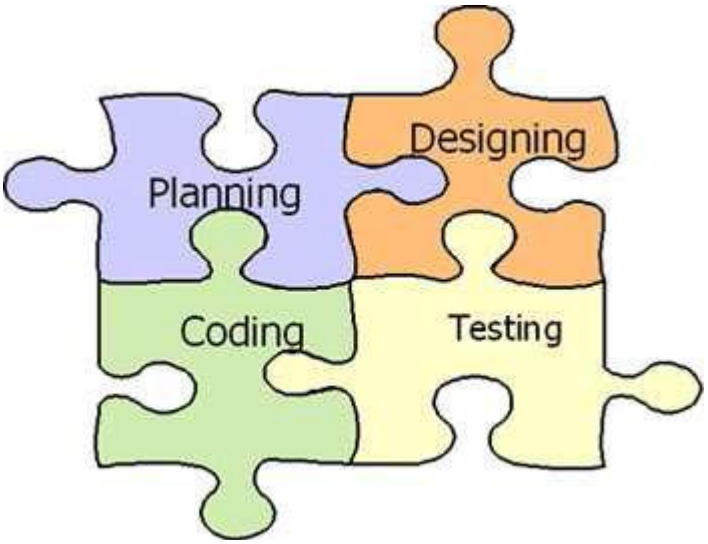

Figura 2 Flujo de trabajo de la metodología XP

XP Propone:

- Empezar en pequeño e ir adicionando funcionabilidad con retroalimentación continua.
- El manejo de cambios es una parte sustantiva del proceso.
- El costo de los cambios no depende de la fase o etapa en que se encuentre el proyecto.
- No introduce funcionabilidad hasta que estas no sean necesarias.
- El cliente o usuario forma parte del grupo de desarrollo.

XP Propone un grupo de derechos con los que cuentan los usuarios del producto al igual que los derechos de los desarrolladores del mismo.

Derechos del cliente:

- Decidir qué se implementa.
- Saber el estado real en que se encuentra el proyecto.
- Añadir, cambiar o quitar requerimientos del producto en cualquier momento del ciclo de desarrollo del mismo.
- Obtener lo máximo de cada semana de trabajo.
- Obtener un sistema funcional cada 3 o 4 meses de desarrollo.

Derechos de los desarrolladores:

- Decidir como se implementan los procesos que se llevan a cabo en el desarrollo del software.
- Pedir aclaraciones de los requerimientos al cliente de forma constante.
- La estimación del esfuerzo para implementar el sistema.

[11]

1.2.3 Microsoft Solution Framework (MSF)

Esta metodología es un poco más flexible, se centra en la gestión de los procesos y deja para un segundo plano la selección o elección de las tecnologías. Esta metodología propone un flujo de actividades de manera circular priorizadas según se muestra en la Figura 3.

MSF cuenta con tres actividades fundamentales priorizadas en el mismo orden en que aparecen a continuación: Planificación, Construcción y Administración.

Figura 3 Como se muestra el flujo de trabajo de MSF.

Características:

Adaptable: Es utilizado en cualquier parte, aunque su uso está limitado a un lugar específico.

Escalable: Organiza equipos pequeños entre tres y cuatro personas, así como también en proyectos que requieren algo más de cincuenta personas.

Flexible: Es utilizada en un ambiente de desarrollo de cualquier cliente.

Tecnología Agnóstica: Puede ser utilizada para el desarrollo de cualquier solución que soporte o esté basada en cualquier otra tecnología.

MSF está compuesta por varios modelos, que son los encargados de planificar las partes que están dentro del proceso de desarrollo de software.

Modelo de arquitectura de proyecto: Este modelo se encuentra diseñado para ir acortando el tiempo de desarrollo de los productos, este define los prototipos para la construcción de proyectos a través del lanzamiento de diferentes versiones del producto.

Modelo de equipo: Este ha sido desarrollado con la idea de mejorar el rendimiento del equipo, proporcionando una estructura modificable para la organización del equipo de desarrollo. Puede ser escalado dependiendo del tamaño o la cantidad de personas que se encuentren libres.

Modelo de procesos: Este modelo está diseñado para lograr un mejoramiento en el control del proyecto, minimizar los riesgos y aumentar la calidad a la hora de acortar el tiempo de desarrollo. Proporciona una estructura a seguir en el ciclo de vida, describiendo las fases, las actividades y la liberación de versiones.

Modelo de gestión de riesgos: Está diseñado para ayudar a identificar las prioridades, tomar las decisiones más correctas y el control de las emergencias que puedan ocurrir en todo el ciclo de desarrollo del producto.

Modelo de diseño del proceso: Está diseñado para distinguir claramente los objetivos y necesidades de los usuarios. Proporcionando un modelo eficiente y flexible centrado en el usuario dando un enfoque iterativo. [11]

1.3 Arquitecto de software:

Sobre este rol caen cierta cantidad de privilegios y decisiones importantes que debe de asumir esta persona de la mejor manera y en medida de los conocimientos con que cuente y desarrollarlos lo mejor posible.

El arquitecto de software es la persona que se encarga de articular la visión arquitectónica, para esto es necesario conceptualizar y experimentar con enfoques de arquitecturas alternativas, creando modelos, componentes y haciendo validaciones de la arquitectura propuesta contra los requerimientos. Esta persona debe de ser capaz de llevar acabo ciertas tareas que son:

Descomponer la aplicación en diferentes capas: La descomposición en capas es fundamental, debe de llevarse al nivel de acomodarla a más descomposiciones en capas inferiores para lograr luego que la aplicación pueda desplegarse con un mejor rendimiento.

Descomponer cada capa lógica en componentes: Se basa en asignar las responsabilidades de ejecución lo más claras posibles, ya sea en clases concretas, abstractas o interfaces que tengan contratos definidos, logrando que los programadores que se encargan de programar la lógica de la aplicación completen la implementación.

Selección de tecnologías: El arquitecto de software debe de ser capaz de manipular la complejidad que pueda ir tomando el producto al ser implementado, de acomodar o adaptar las tecnologías de forma tal que se puedan explotar al máximo las funcionabilidades de las mismas para lograr que el equipo de desarrollo no tenga que lidiar con los mecanismos más complejos.

Realizar una prueba de conceptos de la arquitectura: Es necesaria para poder decidir si a la arquitectura propuesta hay que someterla a ajustes, esto es un elemento en que el arquitecto de software debe de tener confianza y saber inspirarla al resto de los interesados.

Brindar algunos casos de uso de referencia: Se encarga de explicar cómo queda en la práctica la arquitectura propuesta y los componentes que van a implementarse. Esto va a servirle como retroalimentación al arquitecto para ir ganando en confianza con respecto a la asimilación de la arquitectura planteada en la solución del proyecto.

El arquitecto de software es la mano derecha del líder de proyecto para alcanzar el éxito del proyecto y la optimización de las tecnologías correctas que le aportarán al producto el valor real que espera el usuario. En fin, el arquitecto viene siendo el líder técnico del grupo de desarrollo. Él no es un programador pero sí es la aspiración de cualquier programador que quiera tomar decisiones significativas en la solución de un proyecto, este tiene que tener altos conocimientos de tecnologías de desarrollo y prácticas de diseño, es el encargado de la toma de decisiones para lograr la robustez, reusabilidad, portabilidad, flexibilidad de la aplicación, así como la mantenibilidad de la misma.

1.4 Arquitectura de software:

Cuando se representen temas de desarrollo de software se tiene que mencionar la arquitectura de software como elemento que se encarga de las decisiones más significativas sobre la organización de un sistema. Propone Rational Unifield Process en 1999 que:

“Una arquitectura es el sistema de decisiones significativas sobre la organización de un sistema de software, la selección de los elementos estructurales y de sus interfaces por los cuales el sistema es compuesto, junto con su comportamiento según lo especificado en las colaboraciones entre esos elementos, la composición de estos elementos estructurales y del comportamiento en subsistemas

progresivamente más grandes, y el estilo arquitectónico que dirigen esta organización, los elementos y sus interfaces, sus colaboraciones y su composición"[2].

Estas palabras exponen a la arquitectura como una etapa más de la ingeniería de software, que se centra en ser orientada a objeto y a UML. RUP la representa como algo más que una simple composición de herramientas o tecnologías a ser utilizad as en el proceso de desarrollo de software.

La arquitectura de software tributa a una visión abstracta de alto nivel. El objetivo fundamental es la de aportar elementos que ayuden en la toma de decisiones y que a la vez proporcione un lenguaje común que garantice una buena comunicación dentro del equipo de desarrollo. Está encaminada a dirigir el desarrollo de los sistemas de software y contribuir a que estos se realicen o construyan en los límites impuestos por el usuario, garantizando que se cumplan con los requerimientos o requisitos (funcionales y no funcionales) con que cuenta la aplicación que se está desarrollando.

1.4.1 Clases de estilos arquitectónicos:

En la actualidad y con la variedad existente en estilos arquitectónicos no es necesario definir uno nuevo para cada sistema que se vaya a desarrollar. Lo que se hace habitual es que se utilice uno de los que ya existen en dependencia de sus ventajas y desventajas para cada caso en específico.

1.4.1.1 Estilo de flujo de datos:

Esta clase de estilo arquitectónico es apropiada para los grupos de sistemas que implementan algunas transformaciones en sus datos dando pasos sucesivos. Algunos ejemplos de las mismas son la arquitectura tubería-filtros y las que presentan procesos secuenciales por lotes. [11]

- Tubería y filtros

1.4.1.2 Estilo centrado en datos:

Se utiliza preferentemente en el desarrollo de sistemas que se centran en la implementación de accesos y actualizaciones de datos en estructuras de almacenamiento. Ejemplos de esta familia son los repositorios, las bases de datos y las arquitecturas de pizarras. [11]

- Arquitectura de pizarra o repositorio.

1.4.1.3 Estilos llamadas y retornos:

Se centra en la modificabilidad y escalabilidad. Estos estilos son los más utilizados en sistemas de gran escala o grandes sistemas. Algunos ejemplos de las mismas son las arquitecturas de programa principal y subrutinas, los sistemas que se centran en llamadas a procedimientos remotos, los orientados a objetos y los jerárquicos en capas. [12]

- Modelo-Vista-Controlador (MVC).
- Arquitectura en Capas.
- Arquitectura Orientada a Objetos.
- Arquitectura Basada en Componentes.

1.4.1.4 Estilo Peer-to-Peer:

Esta arquitectura se centra en la modificabilidad por medio de la separación en distintas partes, por lo general en procesos que son independientes o entidades que se comunican a través de mensajes. [12]

- Arquitectura Basada en Eventos.
- Arquitectura Orientada a Servicios.
- Arquitectura Basada en Recursos.

1.5 Patrones arquitectónicos:

Se entiende que un patrón es una solución probada que da respuesta a un problema dado, que se presenta en un instante de tiempo determinado en un campo o área de acción determinada. Un patrón describe un problema que se repite una o varias veces en el ambiente de trabajo y le da la solución a ese problema, esto quiere decir que la solución que brinda el patrón puede utilizarse cuantas veces ocurra el problema. Un patrón es un codificador de conocimiento específico acumulado por la experiencia en un dominio dado.

En esencia un patrón no es más que: Una solución a un problema en un contexto determinado, es algo que codifica conocimiento específico acumulado por la experiencia en un dominio. Los elementos con que cuenta un patrón son:

Nombre: Define un vocabulario de diseño y facilita la abstracción.

Problema: Describe como aplicar el patrón, define el conjunto de fuerzas objetivas y restricciones y un grupo de prerrequisitos.

Solución: Elementos que constituyen el diseño y describen las formas para resolver fuerzas.

Consecuencias: Resultados que arrojara el patrón al ser aplicado.

1.5.1 Patrón Tubería y Filtros

El sistema tubería-filtros se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada. Los datos entran al sistema y fluyen a través de los componentes. En el estilo secuencial por lotes, los componentes son programas independientes; el supuesto es que cada paso se ejecuta hasta completarse antes que se inicie el paso siguiente. La variante por lotes es un caso degenerado del estilo, en el cual las tuberías se han vuelto residuales.

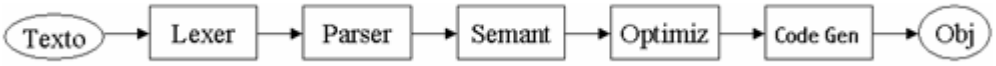

Figura 4 Compilador en tubería-filtro

El uso de este patrón se recomienda cuando:

1. Se puede especificar la secuencia de un número conocido de pasos.
2. No se requiere esperar la respuesta asincrónica de cada paso.
3. Se busca que todos los componentes situados corriente abajo sean capaces de inspeccionar y actuar sobre los datos que vienen de corriente arriba.

Al igual que se reconocen como ventajas de este estilo:

1. Es simple de entender e implementar. Es posible implementar procesos complejos con editores gráficos de líneas de tuberías o con comandos de línea.
2. Fuerza un procesamiento secuencial.
3. Es fácil de envolver en una transacción atómica.
4. Los filtros se pueden empaquetar, y hacer paralelos o distribuidos.

1.5.2 Patrón Pizarra o Repositorio:

Estos patrones se han usado en aplicaciones que requieren complejas interpretaciones de proceso de señales, o en sistemas que involucran acceso compartido a datos con agentes débilmente acoplados. También se han implementado estilos de este tipo en procesos en lotes de base de datos y ambientes de programación organizados como colecciones de herramientas en torno a un repositorio común. Muchos más sistemas de los que se cree están organizados como repositorios: bibliotecas de componentes reutilizables, grandes bases de datos y motores de búsqueda. Algunas arquitecturas de compiladores que suelen presentarse como representativas del estilo tubería-filtros, se podrían representar mejor como propias del estilo de pizarra, dado que muchos compiladores contemporáneos operan en base a información compartida tal como tablas de símbolos y árboles sintácticos abstractos (AST). Estos suelen hacer morphing a estilos de máquinas virtuales o intérpretes. [12]

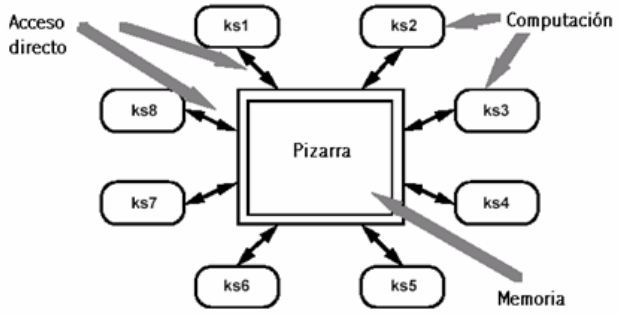

Figura 5 Pizarra

El estilo de pizarra tiene pleno sentido si tanto los agentes (o las fuentes de conocimiento) como la pizarra se entienden en términos virtuales y genéricos, como clases que son susceptibles de instanciarse en diversas variedades de objetos computacionales. De ser así, se podría incluir en este estilo un inmenso repertorio de aplicaciones de optimización y búsqueda en programación genética y evolutiva que de otro modo no encontraría un estilo en el cual encuadrarse.

1.5.3 Patrón Modelo Vista Controlador:

El patrón conocido como Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). Las vistas manejan la visualización de la información. Las clases controladoras interpretan las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado. [12]

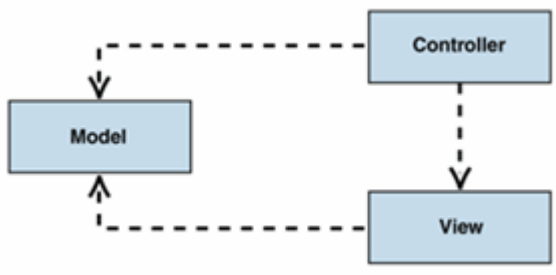

Figura 6 Modelo-Vista-Controlador

Modelo: Es una representación detallada de la información con la que el sistema debe operar. Es en el cual la lógica de datos garantiza y asegura la integridad de los mismos permitiendo derivar nuevos datos.

Como por ejemplo permitiendo la validación de los datos para funcionalidades específicas o simplemente las operaciones con los datos como: saber si es el cumpleaños de un usuario específico.

Vista: Es la encargada de presentar el modelo con que se interactuará con un formato que sea el adecuado para los usuarios. Generalmente es la interfaz de usuario de la aplicación.

Controlador: Es el encargado de mediante la invocación de eventos o acciones del usuario, realizar cambios en el modelo y en algunas ocasiones hasta en las vistas. El flujo de datos que sigue el Controlador generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de la aplicación de alguna manera.
2. El controlador captura la acción que solicitó el usuario y se encarga de gestionar el evento correspondiente a dicha acción.
3. Accede al modelo realizando frecuentemente cambios en dicho modelo o solamente actualizándolo de la forma más adecuada con respecto a la solicitud que realizó el usuario.
4. El controlador encarga a los objetos de la vista la labor de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.

1.5.4 Patrón en Capas:

Este patrón tiene como funcionabilidad principal asignar a diferentes capas funcionabilidades específicas y muy bien definidas. Este patrón se ve utilizado en muchos software de escritorio aunque no excluye la utilización de este en el desarrollo de algunas aplicaciones Web.

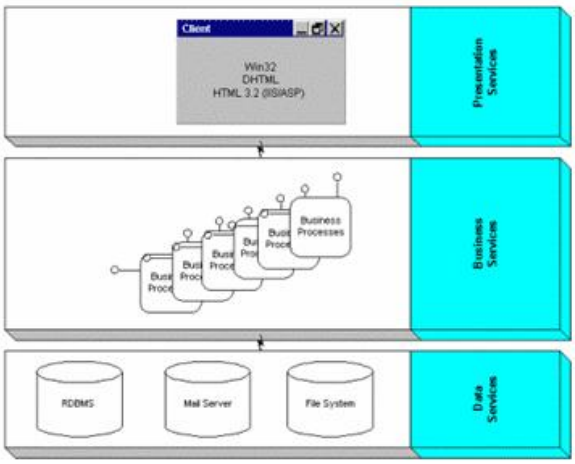

Figura 7 Arquitectura en 3 capas con "cliente flaco" en Windows DNA

La capa de presentación: A esta capa se le asigna la tarea de reunir todos los aspectos del producto que tienen que ver con las interfaces y la interacción con los usuarios esto incluye el manejo del aspecto de las ventanas, reportes, menús y gráficas entre otras.

La capa del dominio de la aplicación: Esta es la encargada de reunir todos los aspectos que van de una manera u otra a automatizar o apoyar los procesos de negocio que llevan a cabo los usuarios de la aplicación. Es la encargada de llevar las tareas de los procesos que se llevan a cabo en la aplicación así como el grupo de reglas y restricciones que son aplicadas. Esta capa también se conoce como lógica de la aplicación.

La capa de acceso a datos: Es la encargada de reunir toda la información del software, referente al manejo del grupo de datos persistentes. Es conocida también como la capa de Base de Datos.

En la actualidad existen varias formas de plantear este patrón, no quiere decir que sean versiones nuevas del mismo, sólo que en dependencia del tipo de problema han ido adaptándolo para lograr una mejor aplicación y seguido se dan a conocer algunas de estas variantes implementadas del patrón en capas.

Arquitectura Top-Down de capas: La arquitectura de esta solución plantea que todos los elementos de una capa determinada pueden enviar solicitudes a las capas inferiores a ella. Quiere decir que muestra una visión en cascada de solicitudes.

Arquitectura Bottom-Up de capas: Esta arquitectura se ve como el planteamiento inverso de la arquitectura Top-Dow antes mencionada ya que esta plantea que cada capa determinada puede enviarle mensajes en un momento dado a una capa superior a ella. Al igual que la anterior muestra una vista en cascada de solicitudes.

1.5.5 Patrón Cliente- Servidor:

La arquitectura reparte su capacidad de proceso en dos aplicaciones algo diferentes pero fuertemente vinculadas entre sí (las aplicaciones clientes y las aplicaciones servidoras).Las ventaja más significativa es la centralización de la gestión de la información y la separación de las responsabilidades. Esta arquitectura viene a sustituir a la arquitectura monolítica en la que no existe una distribución de la información tanto a niveles físicos como lógicos.

A continuación se muestran algunas ventajas de la arquitectura:

Centralización del control: La arquitectura centraliza el control de los accesos, recursos y la integridad de los datos en el servidor garantizando que una aplicación cliente que se encuentre con defecto o que no esté autorizada no pueda dañar los datos.

Escalabilidad: Esta arquitectura permite ir aumentando el número de clientes y servidores paulatinamente y no necesariamente a la misma vez, quiere decir que se pueden agregar aplicaciones clientes y servidoras por separado.

1.6 Caracterización de las herramientas

1.6.1 Herramienta Asistida por computadora para el modelado Visual Paradigm.

Visual Paradigm es una herramienta para un diseño profesional, que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El

Lenguaje de Modelado Unificado (UML) ayuda a una rápida construcción de aplicaciones de calidad, con un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

Listado de características:

1. Soporte de UML versión 2.1
2. Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documentos
3. Modelado colaborativo con CVS y Subversion (nueva característica) Interoperabilidad con modelos UML2 (meta modelos UML 2.x para plataforma Eclipse) a través de XML (nueva característica).
4. Ingeniería inversa - Código a modelo, código a diagrama.
5. Ingeniería inversa Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
6. Generación de código - Modelo a código, diagrama a código.
7. Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
8. Diagramas EJB - Visualización de sistemas EJB.
9. Generación de código y despliegue de EJB´s - Generación de beans para el desarrollo y despliegue de aplicaciones.
10. Diagramas de flujo de datos.
11. Soporte ORM - Generación de objetos Java desde la base de datos.
12. Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
13. Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
14. Generador de informes para generación de documentación Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
15. Importación y exportación de ficheros XMI.
16. Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de MS Visio.
17. Editor de figuras.
18. Otras herramientas y plugins de modelado UML:

- Plataforma Java (Windows/Linux/Mac OS X):
- SDE para Eclipse
- SDE para NetBeans
- SDE para Sun ONE
- SDE para Oracle Jdeveloper
- SDE para Jbuilder
- SDE para IntelliJ IDEA
- SDE para WebLogic Workshop. [3]

1.6.2 Lenguaje de modelado UML (Ingles UML).

UML en un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema el cual involucra una gran cantidad de software. UML pretende llegar a ser un estándar con el que se puedan modelar todos los componentes del proceso de desarrollo de software. Unifica las técnicas de modelados pasadas e incorpora las mejores prácticas actuales. [2]

De UML se puede señalar que es un lenguaje gráfico con sintaxis y semántica bien definidas. La sintaxis de la notación gráfica se especifica mediante su correspondencia con los elementos del modelo semántico subyacente cuya semántica se define de manera semi-formal por medio de un meta-modelo, textos descriptivos y restricciones. [2]

Tipos de modelos UML:

Estáticos:

- Diagrama de casos de uso.
- Diagrama de clases.
- Diagrama de objetos.
- Diagrama de componentes.
- Diagrama de despliegue.

Dinámicos:

- Diagrama de estados.
- Diagrama de actividad.
- Diagrama de interacción.

[4]

1.6.3 Lenguaje PHP para el desarrollo de la aplicación Web.

PHP es un lenguaje de programación interpretado, usado para el desarrollo de sitios Web, para la programación de aplicaciones del lado del servidor o creación de contenido dinámico para sitios Web. Este lenguaje está mundialmente difundido entre las comunidades de programadores, es gratuito y multiplataforma.

Características de PHP:

Gratuito: Al ser un lenguaje libre puede descargarse y utilizarse en cualquier aplicación de manera completamente libre.

Gran Popularidad: Existen en el mundo entero una gran comunidad de desarrolladores y programadores que implementan mejoras continuamente en su código y que estarían encantados de brindar su ayuda en caso de algún problema o sugerencia.

Enorme Eficiencia: Con escaso mantenimiento y un servicio gratuito, los sitios desarrollados con este lenguaje pueden soportar sin problema millones de visitas diarias.

Sencilla integración con múltiples bases de datos: Este lenguaje de programación es esencial para una página Web verdaderamente dinámica, con una integración con bases de datos. Puede conectarse fácilmente con cualquier Base de Datos que sea compatible con ODBC (Open DataBase Connectivity Standard).

1.6.4 IDE de desarrollo Zen Studio.

Zend Studio 5 se ha diseñado por una amplia gama de programadores y existen dos ediciones: Standard y Professional. Zend Studio 5, concebido con el fin de crear aplicaciones altamente fiables, proporciona una facilidad de uso inigualable, escalabilidad, fiabilidad, y la extensión que los programadores profesionales y de empresas requieren para desarrollar, distribuir, depurar y administrar aplicaciones PHP críticas de negocios.

El entorno de desarrollo utilizado para PHP más difundido entre los desarrolladores de páginas Web:

Código y define/especifica Jars adicionales o carpetas de Clase, que pueden utilizarse para el completado de códigos.

Integración del uso y completado de código personalizado de Zend Framework y vista de la lista de las funciones del framework desde la visualización de funciones PHP.

Visualización de los eventos de Zend Platform en una ventana de lista de eventos personalizada y dedicada. Click en cada evento para ver el detalle completo del evento en la ventana del navegador.

Aumentar la productividad con: Soporte PHP 5 completo, analizador de código, carpeta de código, completado de código, coloreado de sintaxis, administrador de proyecto, editor de código, depurador de gráficos y asistentes.

Documentación del código de forma más sencilla, aplicaciones, y proyectos con PHP Documentor, la herramienta de documentación estándar para PHP.

Simplificar el despliegue con la integración FTP y SFTP de forma tal que permita a los programadores en forma segura subir y descargar archivos de proyectos de modo transparente hacia y desde servidores remotos. [13]

1.6.5 Servidor Web Apache.

Es el servidor Web por excelencia para PHP, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Apache es una muestra, al igual que el sistema operativo Linux (un Unix desarrollado inicialmente para PC), de que el trabajo voluntario y cooperativo dentro de Internet es capaz de producir aplicaciones de calidad profesional difíciles de igualar.

A continuación se muestran algunas de las razones de por qué la popularidad de este software libre grandemente reconocido en muchos ámbitos empresariales y tecnológicos.

Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.

Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia al software de manera que si se quiere ver que es lo que se está instalando como servidor, se puede saber, sin ningún secreto, sin ninguna puerta trasera.

Apache es un servidor altamente configurable de diseño modular es muy sencillo ampliar las capacidades de este Servidor. Actualmente existen muchos módulos para Apache que son adaptables y están ahí para ser instalados cuando se les necesiten. Otra de sus importancias es que cualquiera que posea una experiencia en la programación de C o Perl puede escribir un módulo para realizar una función determinada.

Apache trabaja con gran cantidad de lenguajes de programación como Perl, PHP y otros lenguajes de script. Perl destaca en el mundo del script y Apache utiliza su parte de Perl tanto con soporte CGI como con soporte modo Perl. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.

1.6.6 Herramienta para el diseño de la base dato ER/Studio 7.5

ER/Studio versión 7.5 ofrece soporte XML para la generación de esquemas, que permite a los modeladores de datos y desarrolladores de aplicaciones colaborar en iniciativas de arquitectura orientada a servicios (SOA). ER/Studio 7.5 permite la conversión mejorada para incrementar la precisión durante la migración de modelos de datos desde herramientas tales como CA ERwin Data Modeler, Sybase

PowerDesigner y muchas otras. Las nuevas características dan a los modeladores de datos y arquitectos una mayor flexibilidad y control al extender el nivel de adecuación y validación en la aplicación y el ciclo de vida de la base de dato. Los arquitectos de datos ahora pueden colaborar más eficientemente con terceros manejando implementaciones SOAP y gobernabilidad de los datos al definir, desplegar y re-usar los estándares tradicionalmente utilizados para la gestión de la base de datos. La información puede ser mejor aprovechada para un mayor número de audiencia. El objetivo a largo plazo de este software es ayudar a construir modelos de datos bien documentados que refuercen los estándares, permitan la re-utilización y mejoren la eficiencia y calidad de los proyectos.

Nuevas características:

Asistente fácil de usar para la construcción de esquemas XML directamente desde un modelo de datos físico o lógico, permitiendo a los arquitectos de datos incorporar los mismos estándares en el desarrollo SOAP que ellos utilizaron previamente para la construcción de las bases de datos.

Utilería para el nombramiento de estándares y un editor para el mapeo de diferentes tipos de datos que automáticamente transforma el modelo de metadatos.

Utilería para re-arquitectura y conversión del modelo de datos que mejora significativamente la calidad de los metadatos cuando se importan desde plataformas tales como herramientas de modelado, plataformas de BI, ETL, formatos estándar de intercambio de archivos y herramientas de modelado.[5]

1.6.7 Postgres SQL.

Postgre SQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre. [6]

Lo siguiente es una breve lista de algunas de las características, a partir de PostgreSQL 7.1.x.

DBMS Objeto-Relacional

PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.

Altamente_Extensible

PostgreSQL soporta operadores funcionales, métodos de acceso y tipos de datos definidos por el usuario.

Soporte_SQL_Comprensivo

PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

Integridad Referencial

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

API Flexible

La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el sistema de modelado de Bases de Datos relacionales para PostgreSQL (RDBMS PostgreSQL). Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

Lenguajes Procedurales

PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

Cliente/Servidor

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL. [6]

1.6.8 Tortoise SVN

Esta herramienta es libre se encuentra registrada bajo la licencia publica general GNU (GPL). Es un cliente para el sistema de control de versiones Subversion. Maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un repositorio central. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos, y quién hizo el cambio.

La herramienta a la cual se hace mención se integra con mucha facilidad en el Shell de Windows. Permitiendo seguir trabajando con las herramientas que ya se conocen sin tener que emigrar o cambiar a otra diferente cuando se necesiten utilizar las funcionabilidades del control de versiones.

Características de la herramienta:

Integración con el shell de Windows:

TortoiseSVN se integra perfectamente en el Shell de Windows (por ejemplo, el explorador). Esto significa que puede seguir trabajando con las herramientas que ya conoce. Y ni siquiera está obligado a usar el Explorador de Windows. Los menús contextuales de TortoiseSVN también funcionan en otros administradores de archivos, y en el diálogo Fichero/Abrir que es común a la mayoría de aplicaciones estándar de Windows. Sin embargo, hay que tener en cuenta que TortoiseSVN está desarrollado con la mirada puesta en hacerle extensión al Explorador de Windows. Por este motivo, puede que en otras aplicaciones la integración no sea tan completa y que, por ejemplo, los iconos sobre impresionado en las carpetas no se muestren.

Íconos sobre impresionados:

El estado de cada carpeta y fichero versionado se indica por pequeños íconos sobre impresionados. De esta forma, puede ver fácilmente el estado en el que se encuentra su copia de trabajo.

Fácil acceso a los comandos de Subversion:

Todos los comandos de Subversion están disponibles desde el menú contextual del explorador. TortoiseSVN añade su propio submenú allí.

Dado que TortoiseSVN es un cliente de Subversion, también se desea mostrar algunas de las características del propio Subversion:

Versionado de carpetas:

CVS sólo controla la historia de ficheros individuales, pero Subversion implementa un sistema “virtual” de ficheros versionados que sigue la pista de los cambios en todos los árboles de directorios en el tiempo. Los ficheros y los directorios están versionados. Como resultado, hay comandos reales en el lado del cliente como mover y copiar que operan en ficheros y directorios.

Confirmaciones atómicas:

Una confirmación o bien entra en el repositorio completamente, o no entra en absoluto. Esto permite a los desarrolladores construir y confirmar cambios como unidades lógicas. [14]

1.6.9 Subversion.

Esta herramienta forma parte del grupo de programas de software libre que se encargan de la gestión y manejo de ficheros y directorios a través del tiempo. Este es como un servidor de ficheros ordinarios sólo que guarda o recuerda todos los cambios hechos en los mismos, permitiendo con esto la recuperación antigua de datos, o simplemente sentarte a analizar el historial de cambio de dichos ficheros.

Subversion posee las siguientes características:

- Versionado de directorios:

Subversion implementa un sistema de versionado virtual, que almacena los cambios realizados tanto en ficheros como en directorios.

- Verdadero control de versiones:

Subversion soporta operaciones adicionales sobre los ficheros y directorios almacenados guardando el historial de cambios, como por ejemplo añadir, borrar, copiar o renombrar (mover), tanto para ficheros como para directorios.

- Commits Atómicos:

Subversion garantiza que todas las modificaciones sobre un repositorio se realizan completamente, o por el contrario no se realiza ninguna. Esto garantiza la eliminación de problemas de concurrencia, cuando por algún motivo solamente se envía una parte de los cambios al repositorio. Esta propiedad se debe a que Subversion almacena sus datos en una base de datos, que cumple las propiedades ACID (atomicidad, coherencia, aislamiento y permanencia).

- Control de versiones sobre meta datos:
- Consistencia en los datos:

Subversion maneja las diferencias entre los datos almacenados mediante un algoritmo de diferencias, que funciona de forma idéntica tanto para ficheros de texto, como para ficheros binarios (que almacenan código máquina). Ambos tipos de ficheros son almacenados idénticamente en el repositorio, y no es necesario especificar si los datos son binarios o no. [7]

1.6.10 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 Java Script, etc., de una forma muy sencilla y visual.

Además incluye un software de cliente FTP completo, permitiendo entre otras cosas trabajar con mapas visuales de los sitios Web, actualizando el sitio Web en el servidor sin salir del programa.

Características de esta herramienta:

- Integración de RSS: con Dreamweaver 8 se puede integrar entradas RSS provenientes de otras páginas con sólo introducir la fuente y arrastrar y colocar los campos. De esta forma se puede introducir datos en formato XML fácil y cómodamente.
- 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 se puede 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, WebTV, PDAs...).
- Accesibilidad: Dreamweaver 8 incorpora las normas de accesibilidad de prioridad dos, marcadas por la WCAG/W3C.
- Transferencia de archivos: Ahora con Dreamweaver 8 se puede seguir trabajando con tus archivos mientras el programa se comunica con tu servidor e incluye los archivos creados o modificados recientemente. Su sincronización ha mejorado notablemente siendo posible una mejor gestión de cambios, además de permitir en uso de bloqueo/desbloqueo de archivos para que estos no se sobrescriban.
- 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, se puede encontrarla en la parte lateral izquierda del modo de Código, esta barra hace mucho más accesible el código al permitir la navegación por etiquetas y su contracción. Una de las nuevas novedades es la posibilidad de añadir comentarios con un sólo click.

Conclusiones:

En el presente capítulo fueron abordados todo un grupo de conceptos que son de suma importancia para que los lectores puedan entender el contenido de este trabajo. Se explica en que consiste el trabajo y hacia que rama del conocimiento está dirigida, en que consisten tres de las metodologías de desarrollo de software. De este capítulo depende que se pueda entender el contenido del resto de la investigación.

Capítulo #2: Descripción de la alternativa propuesta.

Introducción:

En este capítulo se abordarán dos documentos de suma importancia para el desarrollo de una arquitectura de software robusta, flexible y reusable. Para la parte del trabajo que se desarrollará más adelante se dejará la tarea de definir la línea base de la arquitectura así como las vistas arquitectónicas, dos tareas que es necesario de su cuidado y seguimiento ya que en ellas se muestra como quedará la alternativa arquitectónica que se plantea para el desarrollo del software de gestión de los recursos humanos.

2.1 Línea Base de la Arquitectura.

2.1.1 Propósito.

La línea base de la arquitectura tiene como propósito proporcionar la información necesaria para estructurar el sistema desde el más alto nivel de abstracción. En ella se describe la estructura del sistema en cuanto a los elementos, las configuraciones y sus restricciones.

Los usuarios de la Línea Base de la Arquitectura son:

El equipo de arquitectos del proyecto, que le sirve de guía para la toma de decisiones arquitectónicas y son los encargados del mantenimiento y refinamiento de esta línea base.

Los miembros del equipo de desarrollo encargados de desarrollar la aplicación, la utilizan como la guía para la implementación del sistema.

Los clientes tienen en ella una garantía de la calidad y el conocimiento sobre en que tecnología está desarrollada su solución.

2.1.2 Alcance.

La Línea Base de la Arquitectura describe la estructura de la aplicación en su más alto nivel de abstracción.

Describe detalladamente el organigrama de la arquitectura encarnada en los estilos arquitectónicos que se utilizarán; los principales frameworks de desarrollo y como se adaptarán a la solución; se propone la utilización de un conjunto de patrones que resuelven problemas que se podrían presentar a lo largo del desarrollo del sistema.

Se proponen las principales tecnologías y herramientas que soportan los estilos y patrones especificados y cumplen con las restricciones del sistema.

2.1.3 Metodología de Desarrollo Utilizada.

El sistema para el cual se centra el desarrollo de la investigación será llevado a cabo haciendo uso de la metodología de desarrollo de software Racional Unified Process (RUP) caracterizada en el capítulo 1. Para esta metodología se pone de manifiesto tres elementos muy importantes para el desarrollo de la arquitectura y es que dicha metodología está:

- Dirigida por casos de uso.
- Centrada en la arquitectura.
- Iterativa e Incremental.

2.1.4 Estructura organizativa de la arquitectura.

En el capítulo anterior se vio que la arquitectura de software es la organización fundamental de un sistema. Es la encargada de sus componentes, la relación entre ellos así como su ambiente y los principios que orientan su diseño y evolución.

El mayor nivel de abstracción para estructurar un sistema son los estilos arquitectónicos, la elección del mismo está dado por el tipo de sistema que se vaya a desarrollar.

Por lo antes expuesto y los estilos arquitectónicos caracterizados en el capítulo 1 se decide que el estilo de clases arquitectónicas a aplicar en el desarrollo de la aplicación será el estilo de Llamada y Retornos,

dentro de este estilo arquitectónico se definen un grupo de patrones que dan respuesta a ciertos problemas, el patrón que se utilizará en el desarrollo de esta aplicación es el conocido como Modelo-Vista-Controlador (MVC).

2.1.5 Patrón Arquitectónico Seleccionado Modelo-Vista-Controlador (MVC).

El MVC ha sido propio de las aplicaciones en Smalltalk por lo menos desde 1992, antes que se generalizaran las arquitecturas en capas múltiples. Un propósito común en numerosos sistemas es el de tomar datos de un almacenamiento y mostrarlos al usuario. Luego que el usuario introduce modificaciones, las mismas se reflejan en el almacenamiento. Dado que el flujo de información ocurre entre el almacenamiento y la interfaz, una tentación común, un impulso espontáneo (hoy se llamaría un anti-patrón) es unir ambas piezas para reducir la cantidad de código y optimizar el rendimiento. Un problema es que las aplicaciones tienden a incorporar lógica de negocios que van más allá de la transmisión de datos.

Este patrón presenta un grupo de ventajas y de desventajas que quedarían enumeradas como siguen:

Ventajas:

1. Se pueden mostrar distintas variantes de interfaces simultáneamente, porque la vista no depende del modelo.
2. La interface tiende a cambiar más rápido que las reglas de negocios. Agregar nuevos tipos de vista no afecta al modelo.

Desventajas:

1. Puede aumentar un poco la complejidad de la solución. Como está guiado por eventos, puede ser algo más difícil de depurar.
2. Si hay demasiados cambios en el modelo, la vista puede caer bajo un diluvio de actualizaciones, a menos que se prevea programáticamente.

2.2 Metas y restricciones arquitectónicas.

2.2.1 Requerimientos de Hardware

- Estaciones de trabajo o PC Clientes
    1. Periféricos: Mouse, Teclado
    2. Tarjeta de red
    3. 128 MB de Ram o Superior
- Servidor de Aplicaciones
    1. Periféricos: Mouse, teclado.
    2. Tarjeta de Red
    3. 512 MB de RAM o más
    4. 10 GB de espacio en disco
- Servidor de Base Datos
    1. Periféricos: Mouse, teclado.
    2. Tarjeta de Red.
    3. 512 MB de Ram o superior.
    4. 40 GB o más GB de disco

2.2.2 Requisitos de Software

1. El sistema debe ser multiplataforma, especialmente para Windows XP y Linux.
2. El sistema debe visualizarse en Internet Explorer 6.0 y Mozilla 3.0 o superior.
3. El sistema debe tener como gestor de Base de datos Postgres SQL 8.2.

2.2.3 Restricciones de diseño o implementación.

1. El sistema será implementado en el lenguaje PHP
2. El sistema deberá utilizar como IDE de desarrollo el Zend Development Environment.
3. El sistema usará como herramienta de modelación el Visual Paradigm for UML 6.0 Enterprice Edition.

2.2.4 Requerimientos de apariencia o interfaz externa

1. El sistema debe tener una interfaz intuitiva, organizada y de fácil entendimiento para el usuario.
2. El sistema debe presentar una interfaz agradable, que favorezca el estado de ánimo del cliente y que combine correctamente los colores, tipo de letra y tamaño y que los íconos estén en correspondencia con lo que representan.

2.2.4 Requerimientos de Seguridad

1. El sistema deberá ser capaz de permitir que cada usuario se autentique.
2. El sistema deberá garantizar que las funcionalidades se muestren de acuerdo al nivel de usuario que este activo.
3. El sistema deberá estar protegido contra acciones que no estén autorizadas o que puedan afectar la integridad de los datos.
4. El sistema deberá verificar acciones irreversibles (eliminaciones).
5. El sistema deberá verificar que los datos no viajen de forma transparente por la red, deberán ser encriptados mediante los métodos de encriptación SHA1 (método irreversible) y la criptografía de llave pública esta última utilizada en el protocolo Secure Sockets Layer (SSL).

2.2.5 Requerimientos de rendimiento

1. El sistema deberá tener un tiempo de respuesta rápida a cualquier solicitud, para ello se utilizará al máximo los recursos de las máquinas donde esté instalado el sistema.

2.2.6 Requerimientos de Usabilidad

1. El sistema deberá ser utilizado por usuarios que tengan conocimiento acerca del funcionamiento y procesamiento de la información con que se trabaja en el sistema.

2.2.7 Requerimientos de soporte

1. El sistema deberá ser revisado y actualizado cada 6 meses.

2.3 Selección de las tecnologías de implementación.

En esta sección del trabajo se describe el lenguaje utilizado para el desarrollo del software así como todas las herramientas que van a utilizarse para la implementación de la aplicación. Estas herramientas y lenguajes ya se encuentran caracterizadas en el Capítulo 1 por ello solo serán abordadas las condiciones del sistema que se está desarrollando y las características de dichas herramientas que sirven para dar solución o desarrollar a esta aplicación en un momento dado.

Partiendo que el sistema será una aplicación Web y que para el desarrollo de la misma se deben evitar las restricciones que traen aparejado algunos lenguajes de desarrollo de páginas Web propietario se utilizará como lenguaje de programación el PHP que ya no es un lenguaje relativamente nuevo, diseñado desde sus inicios con el fin único de diseñar aplicaciones Web. Las tareas más habituales en el desarrollo de estas aplicaciones, pueden hacerse con PHP de forma fácil, rápida y efectiva. PHP resulta fácil y ameno de aprender para recién llegados al mundo de la programación. Es fácil dar los primeros pasos y ver los resultados rápidamente. PHP es un lenguaje multiplataforma, y no propietario. Un script PHP normal puede ejecutarse sin cambiar ni una sola línea de código en cualquier servidor que interprete PHP. En su desarrollo hay un proceso de colaboración que hace que se tengan inmediatamente disponibles, de forma gratuita, una enorme cantidad de recursos: el lenguaje en sí, el servidor para ejecutarlo, manuales y tutoriales y sobre todo scripts, que se pueden descargar y usar rápidamente. Esta abundancia de código libremente disponible ayuda aun más en el proceso de desarrollo. Según una encuesta de Netcraft publicado en abril de 2004, PHP está siendo utilizado por más del 45% de los sitios en Internet de los 37,6 millones de sitios Web en todo el mundo [10], PHP se está ejecutando en más de 9 millones de sitios, y sigue creciendo a un ritmo explosivo. En los dos últimos años PHP tiene una tasa promedio de 6,5% de crecimiento mensual.

Para el desarrollo de esta aplicación no se tomará ningún framework, se utiliza para la implementación el IDE de desarrollo Zend Development Environment. Ya que este IDE es un programa de la casa Zend, impulsores del lenguaje de servidor PHP, orientada a desarrollar aplicaciones Web.

El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. Está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows, Linux y MacOS, aunque el desarrollo de las versiones de este último sistema se retrase un poco más.

Zend Studio consta de dos partes en las que se dividen las funcionalidades, de parte del cliente y las del servidor. Las dos partes se instalan por separado, la parte cliente contiene la interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer del servidor, que instala Apache y el módulo PHP, en caso de que estén instalados, los configura para trabajar juntos en depuración.

Permite además editar varios archivos y moverse fácilmente entre ellos, marcar a qué elementos corresponden los inicios y cierres de las etiquetas, paréntesis o llaves, moverse al principio o al final de una función, identificación automática del código, etc.

Este programa ayuda a aumentar la productividad. Todas las opciones que dispone están pensadas con acierto por personas que conocen como nadie la tecnología. Zend Studio incorpora suficientes ayudas como para que merezca la pena.

Por cuestiones de tiempo y de mejorar la calidad y facilidad de programación del diseño de las páginas del producto se propone la utilización de una herramienta que aunque no forma parte del grupo de herramientas de la licencia libre, sin dudas es una de las mejores para realizar esta tarea; se habla del Dreamweaver 8. Herramienta de diseño Web de las más interesantes para gestionar de forma completa la creación y diseño de toda una página Web, sigue siendo una herramienta al alcance de muy pocos debido a su elevado precio, esto hace, que el usuario final busque otras opciones más asequibles, como FrontPage, HotMetal ó Namo Web Editor. También es cierto que Dreamweaver 8 se encuentra orientado a un uso profesional en el desarrollo de Webs, y con esta idea, se vuelve nuevamente a que los puntos de referencia de los usuarios más modestos no sean otros que los programas ya mencionados anteriormente.

Con todo el material que incorpora Dreamweaver 8 puede gestionar cualquier tipo de páginas y, por supuesto, usar su propia tecnología Flash ó Shockwave que dotan a la página de unos aspectos visuales inmejorables dentro de Internet.

El mayor desacierto con Dreamweaver, era su carácter de herramienta html, ahora el aspecto de la ventana de diseño ha mejorado, permitiendo realizar más funciones. De forma simultánea, se tiene acceso tanto al código como al diseño, pudiéndose ver sus vistas dentro de la misma ventana y retocar de forma manual el código html de un objeto determinado. Gracias a que viene incluida en el programa, Dreamweaver 8 es uno de los paquetes de referencia del mercado, sin él, la aplicación no sería la misma.

Para poder trabajar con PHP es necesario una tecnología de servidor que sea capaz de interpretarlo, para ello se selecciona la tecnología de servidor Apache que además del grupo de características que se tratan en el capítulo 1 es necesario responder el por qué del uso de este servidor.

Muchos de los servidores Webs que se pueden encontrar son comerciales, por eso mismo, más adecuados para usuarios profesionales que pueden permitirse inversiones en este sentido; además, un software profesional no es a priori mejor que uno libre: es más, a veces sucede justo lo contrario. Otra razón para preferir Apache es la gran difusión dentro de los servidores Web, impecable tarjeta de visita que garantiza rendimiento y estabilidad. Y además Apache es un proyecto, y como proyecto, está abierto a las críticas, parches y corrección de errores sugeridos directamente por sus usuarios, que pueden entrar a formar parte del equipo de desarrollo; así mismo, la posibilidad de contar con un software con tantas fuentes es algo digno de tener en cuenta.

El equipo Apache se acerca a los usuarios en cuanto a mantenimiento y documentación, manteniendo la documentación al día en diferentes sitios Web, y dotando a cada instalación con un manual de aproximadamente 1.5Mb, que quizá no sirva para revelar todos los secretos del servidor de red, pero ayudará en su configuración e instalación.

Además de la excelente compatibilidad que ha logrado a lo largo de todos estos años con PHP, lenguaje en el cual se desarrollará la aplicación.

Una aplicación de gestión de recursos humanos como la que se pretende desarrollar debe contener una gran cantidad de datos que fluyen por ella, así como una cantidad de información guardada; para resolver esto es necesario contar con un Servidor de Base Datos. Para este servidor se utiliza el Postgres SQL, que es ideal para bases de datos donde se requiera recibir gran cantidad de peticiones, por lo que es factible su uso en grandes bases de datos. Además de presentar soporte de vistas, subconsultas, tener integridad referencial y buena escalabilidad, Postgres SQL es uno de los pocos gestores de base datos multiplataforma existentes, esta herramienta tiene una alta aceptación en los desarrolladores de bases de datos en el mundo y de su compatibilidad con diferentes sistemas operativos como Linux, Solaris, HP-UX, AIX, IRIX, FreeBSD, OpenBSD, NetBSD, MacOs, SCO OpenServer, SCO Unixware, BeOS, BSDI, Compaq Tru64, QNX, WinNT/2000. Además de su estabilidad, seguridad, presenta procedimientos almacenados, soportes disparadores. Además de compatibilidad con un sin números de sistemas operativos es altamente compatible con ODBC, JDBC, C/C++, SQL embebido (en C), Tcl/Tk, Perl, Python, PHP lenguaje de programación en el que desarrollará el sitio Web.

En este momento el sistema ya está preparado para competir con los sistemas gestores de bases de datos comerciales en algunos ámbitos. Algo que gusta de la comunidad de desarrollo es que: son honestos y reconocen las limitaciones del sistema. También que cualquier base de datos que se esté usando activamente es un elemento dinámico y vivo en el que se están cambiando los datos constantemente y donde el tamaño de los datos almacenados suele ir creciendo con el tiempo. Postgres SQL tiene una capacidad ilimitada de almacenamiento de datos.

2.4 Descripción de las 4 vistas de la arquitectura.

Racional Unified Process metodología seleccionada para el desarrollo de este software de gestión define la arquitectura siguiendo 4+1 vistas.

2.4.1 Vistas de caso de uso.

A partir de la vista de Casos de Uso, se puede definir los escenarios o los casos de uso que serán de interés para cada iteración del ciclo de desarrollo. Esta describe los escenarios o casos de uso que tienen significación y que encapsulan la funcionalidad central del sistema. Los casos de uso del sistema arquitectónicamente más significativos de la presente iteración se muestran a continuación:

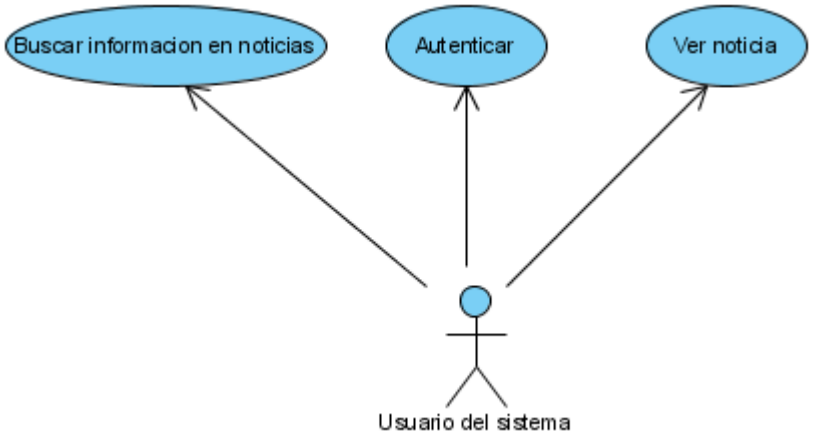

Figura 8 DCUS Usuario del Sistema

De este DCUS para los usuarios no autenticados que cuenta con tres Casos de Usos se decide que: El Casos de Uso arquitectónicamente significativos es el CUS Autenticar.

- **Caso de Uso Autenticar**

Los usuarios introducen sus credenciales del dominio para que el sistema las verifique y estos puedan hacer todas las funcionalidades según su rol. El mismo introduce su nombre de usuario y contraseña. El sistema procesa los datos entrados, permitiendo o no el acceso al sistema en dependencia del rol del usuario. El rol de Usuario Avanzado tiene acceso a total a la información y todos los módulos de administración de la aplicación en el caso del rol de Usuarios del Sistema solamente van a tener acceso a las actividades que se le hayan definido por los administradores del sistema o los Usuarios Avanzados.

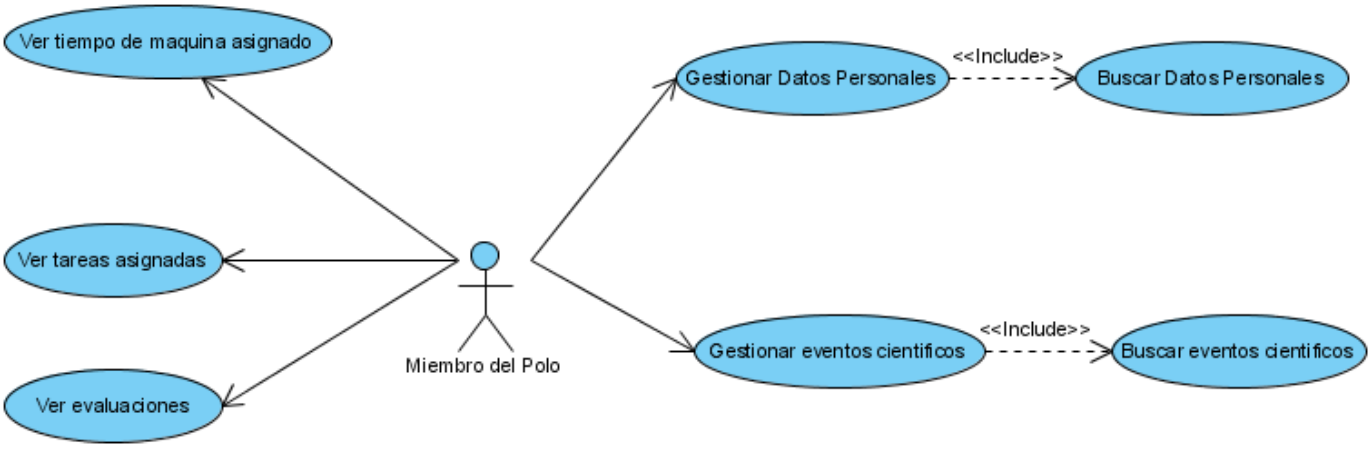

Figura 9 DCUS para los Miembros del Polo

CUS Arquitectónicamente significativos del DCUS Usuarios del Sistema

- **CUS Gestionar Datos Personales**

El caso de uso se inicia cuando el Usuario del sistema se dispone a gestionar sus datos personales escogiendo esta opción en el menú. Luego este escoge qué desea hacer si ver o modificar sus datos personales. Este caso de uso concluye cuando el Usuario del sistema realiza la solicitud. Para poder hace algunas de estas actividades es usuario debe de estar ya previamente autenticado en el sistema.

- **CUS Ver Evaluaciones**

Para el inicio de este CU el usuario debe haberse autenticado previamente en el sistema. Entonces ya puede acceder a la opción del menú evaluaciones, donde revisa las mismas. En este caso solamente este usuario va a poder ver su evaluación no modificarlas ni hacerle ningún cambio a menos que cuente o se le asignes esos privilegios dentro del Sitio Web.

- **CUS Ver Tiempo de Máquina Asignado**

El actor debe estar previamente autenticado como Usuario del sistema. El caso de uso se inicia cuando el usuario del sistema desea ver su tiempo de máquina y va a la opción del menú distribución del tiempo de máquina. El sistema le mostrará su tiempo de máquina si este usuario tiene alguno asignado, sino solamente le mostrará todos los tiempos de máquina para que pueda ver en que máquina podrá acceder en ese horario.

- **CUS Ver Tareas Asignadas**

El actor debe estar previamente autenticado como Usuario del sistema. El usuario del sistema revisa las tareas asignadas. La aplicación debe de ser capaz de mostrar al usuario el grupo de tareas con que cuenta el usuario así como el tiempo de inicio y fin de la tarea y el estado de cumplimiento. En caso de que no tenga tareas el sistema le mostrará el mensaje de que en ese momento el usuario no tiene tareas a realizar.

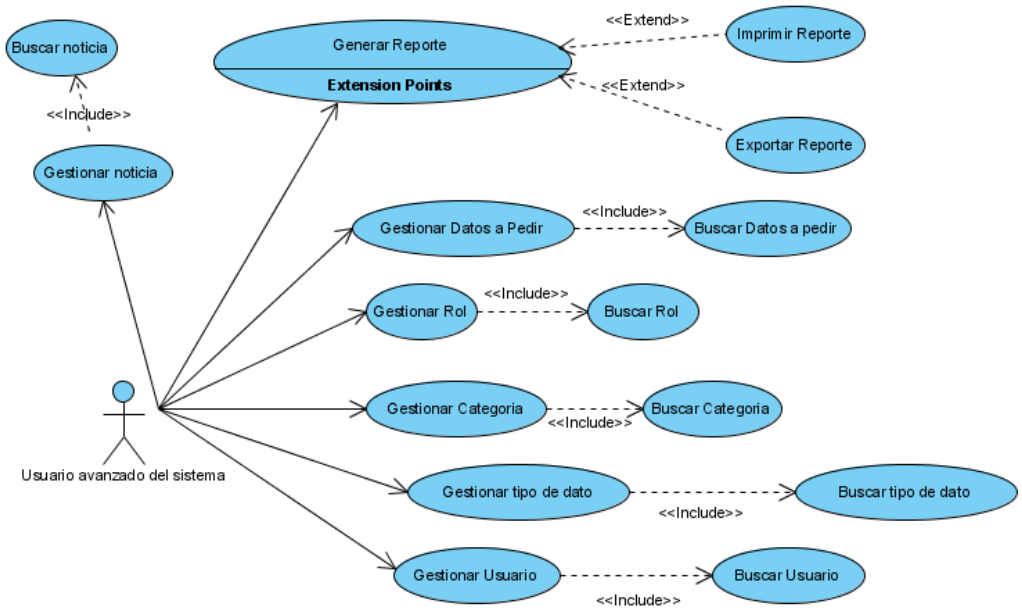

Figura 10 DCUS del Usuario Avanzado del Sistema

CUS Arquitectónicamente significativos del DCUS Usuario Avanzado del Sistema

- **CUS Gestionar Dato a Pedir**

El actor debe estar previamente autenticado como Usuario avanzado del sistema. El caso de uso se inicia cuando el Usuario avanzado del sistema se dispone a gestionar dato a pedir escogiendo esta opción en el menú. Luego escoge qué desea hacer si ver, eliminar, modificar o registrar un Dato a Pedir. Este caso de uso concluye cuando el Usuario avanzado del sistema realiza la solicitud.

- **CUS Gestionar Rol**

El actor debe estar previamente autenticado como Usuario avanzado del sistema. El caso de uso se inicia cuando el Usuario avanzado del sistema se dispone a gestionar los roles escogiendo esta opción en el menú. Luego escoge qué desea hacer, si ver, eliminar, modificar o registrar un rol. Este caso de uso concluye cuando el Usuario avanzado del sistema realiza la solicitud. El rol que se gestiona es el rol que juega un usuario determinado dentro del proyecto al que pertenece.

- **CUS Gestionar Usuario**

El actor debe estar previamente autenticado como Usuario avanzado del sistema. El caso de uso se inicia cuando el Usuario avanzado del sistema se dispone a gestionar usuario escogiendo esta opción en el menú. Luego escoge qué desea hacer si ver, eliminar, modificar o registrar un usuario. Este caso de uso concluye cuando el Usuario avanzado del sistema realiza la solicitud. Este caso de uso es el encargado de la gestión de los estudiantes o profesores en los polos y proyectos productivos de la facultad 9.

- **CUS Generar Reporte**

El actor debe estar previamente autenticado como Usuario avanzado del sistema. El caso de uso se inicia cuando el Usuario avanzado del sistema se dispone a generar reporte escogiendo esta opción en el menú. Luego escoge qué reporte desea generar, así como los datos que quiere que se vean en el mismo y además, puede si lo desea, imprimir o exportar el reporte. El caso de uso concluye cuando el Usuario avanzado del sistema realiza la solicitud.

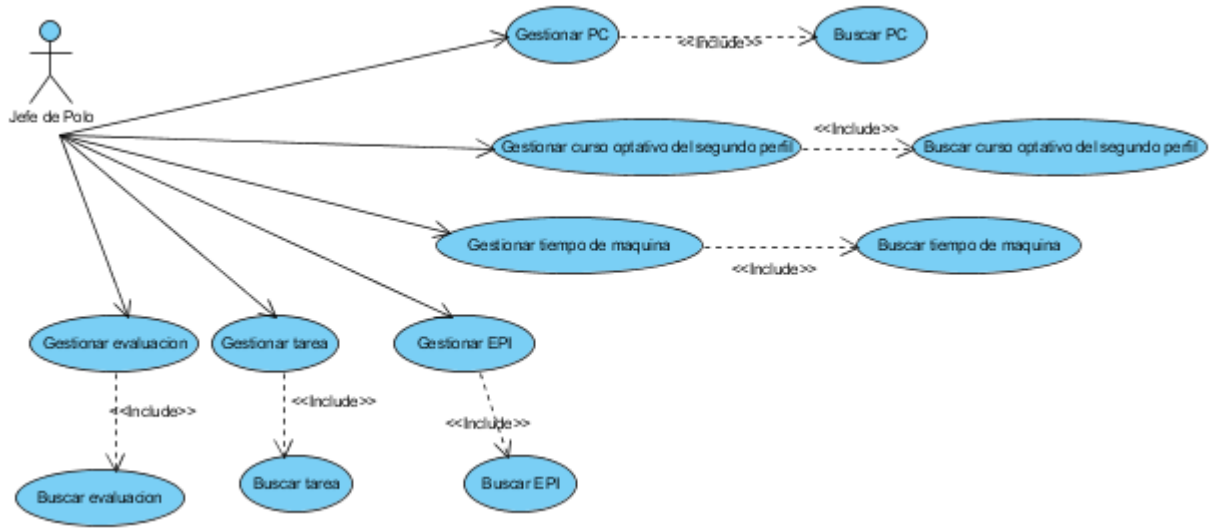

Figura 11 DCUS del Jefe de Polo

CUS Arquitectónicamente significativos del DCUS Usuario Avanzado del Sistema

- **CUS Gestionar Tiempo de Máquina**

El actor debe estar previamente autenticado como usuario del sistema y tener los privilegios que le son asignados al Jefe de Polo. El caso de uso se inicia cuando el Jefe Polo se dispone a gestionar el tiempo de máquina con que cuenta el polo productivo escogiendo esta opción en el menú. Luego este escoge que desea hacer si ver, eliminar, modificar o registrar un tiempo de máquina. Este caso de uso concluye cuando el Jefe Polo realiza la solicitud.

- **CUS Gestionar Tarea**

El actor debe estar previamente autenticado como Usuario del sistema y tener los privilegios que le son asignados al Jefe de Polo. El caso de uso se inicia cuando el Jefe Polo se dispone a gestionar las Tareas con las que cuenta el polo productivo escogiendo esta opción en el menú. Luego este escoge que desea hacer si ver, eliminar, modificar o registrar una Tarea. Este caso de uso concluye cuando el Jefe Polo realiza la solicitud.

- **CUS Gestionar Evaluación**

El actor debe estar previamente autenticado como Usuario del sistema y tener los privilegios que le son asignados al Jefe de Polo. El caso de uso se inicia cuando el Jefe Polo se dispone a gestionar las Evaluaciones con las que cuentan los integrantes del polo productivo escogiendo esta opción en el menú. Luego este escoge que desea hacer si ver, eliminar, modificar o registrar una Evaluación. Este caso de uso concluye cuando el Jefe Polo realiza la solicitud.

Figura 12 DCUS del Vicedecano de Producción

CUS Arquitectónicamente significativos del DCUS del Vicedecano de Producción.

- **CUS Gestionar Polo**

El actor debe estar previamente autenticado como Usuario del sistema y tener los privilegios que le son asignados al Vicedecano de Producción. El caso de uso se inicia cuando el Vicedecano de Producción se dispone a gestionar los polos productivos escogiendo esta opción en el menú. Luego este escoge que desea hacer si ver, eliminar, modificar o registrar un polo productivo. Este caso de uso concluye cuando el Vicedecano de Producción realiza la solicitud.

2.4.2 Vista lógica.

La vista lógica tiene gran importancia en el documento de descripción de la arquitectura pues en ella se hace referencia a la distribución lógica de la aplicación en partes, así como la dependencia que existen entre las mismas, por otro lado se encuentra la distribución en subsistemas y el análisis de las interfaces que proporcionan para la comunicación entre ellos.

Distribución en Partes del sistema

La aplicación cuenta con cuatro paquetes fundamentales como quedan descritos a continuación:

Control

Este paquete cuenta con todo un subsistema que agrupan las clases encargadas de la selección de los eventos para cada Caso de Uso, darle interpretación a estos eventos y darle a las otras dos partes que conforman el modelo los mensajes de la acción a realizar previamente procesada por la clase correspondiente dentro de este paquete. Las clases que agrupa dentro de su subsistema este paquete son las encargadas de informales a las interfaces de que debe mostrar y en que momento, a demás de informarle a la parte de modelo las acciones a realizar.

View

Este paquete cuenta la agrupación de dos subsistemas que van a agrupar las clases que se utilizan para mostrar la información que desea ver el usuario y los recursos que necesitan para la actualización de las vistas. Estos dos subsistemas que se agrupan en este paquete son las encargadas de la interacción del usuario con la aplicación.

Model

En este paquete se encuentran un grupo de de subsistemas con las funcionalidades de la representación detallada de la información con la que el sistema debe operar. Así como la tarea de garantizar y asegurar la integridad de los mismos, permitiendo derivar nuevos datos. Como por ejemplo permitiendo la validación de los datos para funcionalidades específicas o simplemente las operaciones con los mismos.

Global

Este paquete cuenta con dos subsistemas destinados a brindar las funcionalidades comunes o de apoyo para los paquetes Control y Model.

A continuación se muestra la distribución de los paquetes significativos dentro de cada una de las partes, así como la relación existente entre las partes que conforma la arquitectura.

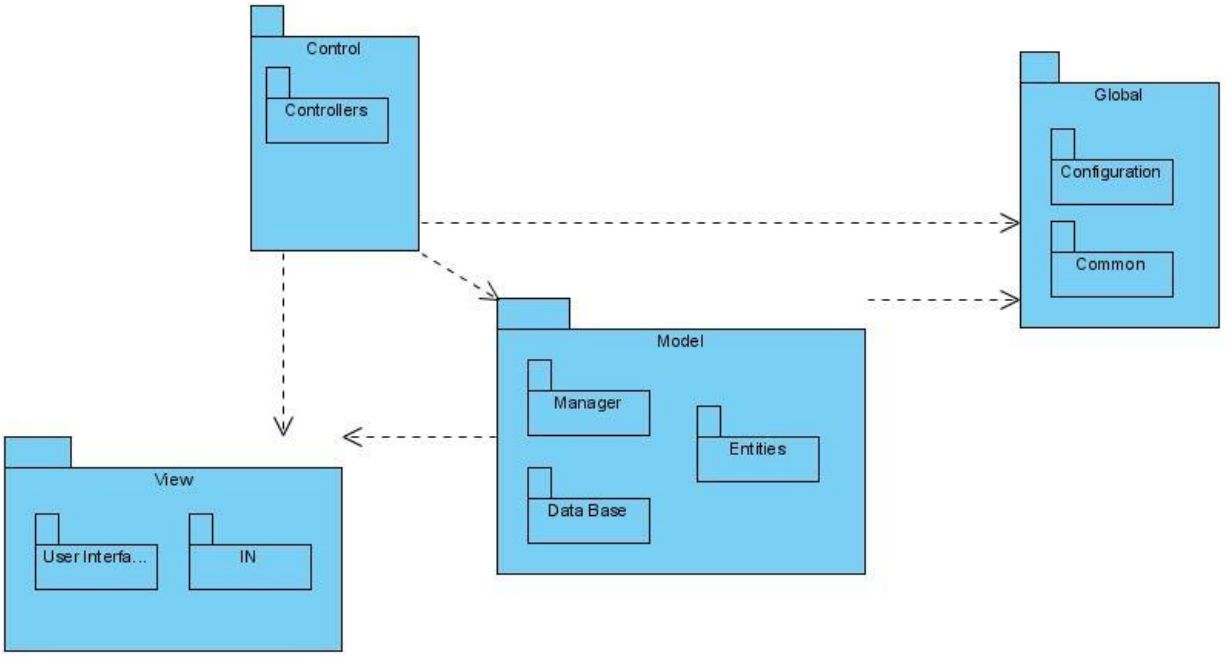

Figura 13 Diagrama de Paquetes para los CU Arquitectónicamente Significativos

Subsistemas y Dependencias

Existe una relación lógica entre los diferentes subsistemas que conforman la aplicación. Como se muestra en la Figura 13 la aplicación cuenta con ocho subsistemas. La ventaja de utilizar esta separación lógica luego de haberse identificado y modelado correctamente es que el tiempo de desarrollo del sistema se acotaría considerablemente. Para obtener una arquitectura ejecutable se realiza la consideración de por cada subsistema las clases que le dan la funcionabilidad a cada caso de uso arquitectónicamente significativos. Se mostrarán también los diferentes diagramas de clases correspondiente a cada caso de uso descrito anteriormente para ver como es la relación y dependencia de dichas clases y su vinculación en cada subsistema.

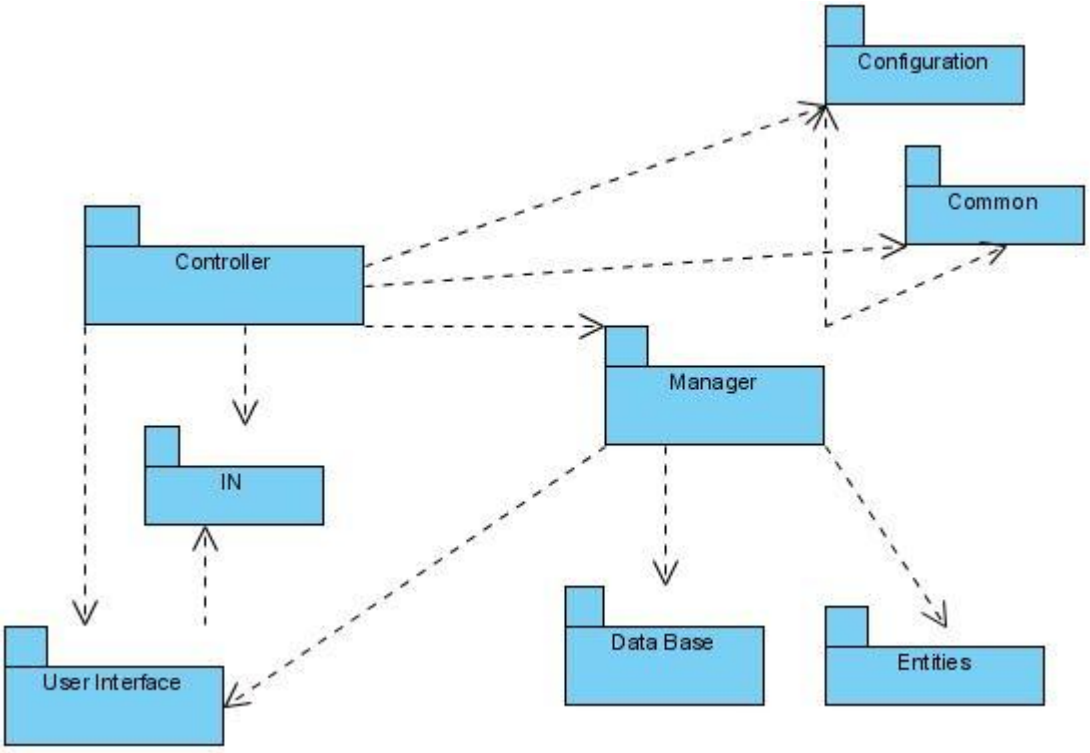

Figura 14 Relaciones de los Subsistemas de Diseño

Controller

Contiene un grupo de clases que garantizan para cada un de los Casos de Uso Críticos el tratamiento y selección de los eventos, interpretan las acciones del usuario, accediendo a las operaciones de negocio de la aplicación y modificando a partir de sus resultados el estado del modelo y la navegación entre vistas. Incluye los métodos que encapsulan el acceso asíncrono a los servicios Web y el control de navegación entre pantallas.

User Interface

Contiene la agrupación de las clases encargadas de la visualización de los datos así como brindar las interfaces de los formularios con las cuales interactuarán los clientes de la aplicación.

IN

En el subsistema se encuentran las clases con los datos que necesitan las User Interfaces para la visualización. Las clases que se encuentran agrupadas en este subsistema son creadas directamente por las clases del subsistema Controller. De este grupo depende totalmente el visualizado correcto por parte de las User Interface de la información que se debe mostrar para una petición específica del usuario.

Configuration

El subsistema cuenta con una sola clase que brinda a las clases dentro del los subsistemas Controller y Manager las constantes o grupo de variables que son persistentes en todo el siclo de vida y que sirven para todo el proceso de configuración de la aplicación. Ejemplo: Las constantes con la información del nombre del La Base Dato y con los URL de las páginas del Sitio Web.

Common

El subsistema Common cuenta con un grupo de clases comunes para las clases de los dos otros grupos que dependen de él como se ve en la Figura 13. Ejemplo de estas clases son: La encargada del tratamiento de Excepciones y la encargada de los mensajes globales.

Manager

Agrupa las clases que se encargan de toda la lógica del negocio y validaciones de la aplicación para cada uno de los Casos de Uso críticos.

Data Base

Agrupa todas las clases correspondiente con el trabajo de La Base Dato (BD). En este subsistema estarán las clases que se encargarán de la conexión con la BD, preparación de las consultas e inserción

de los parámetros para realizar las consultas. Agrupa también otro grupo de clases que sirven de ayuda a las clases antes mencionadas a realizar sus funciones.

Entities

El subsistema agrupa dentro del todas las entidades correspondientes a los casos de uso arquitectónicamente significativos con que cuenta la aplicación.

2.4.2.1 Diagrama de Clases del Diseño para los CU Arquitectónicamente Significativo

En este subepígrafe se mostrarán los diagramas de clases del diseño para cada uno de los casos de usos arquitectónicamente significativos definidos anteriormente en la vista de casos de uso.

- **CUS Autenticar**

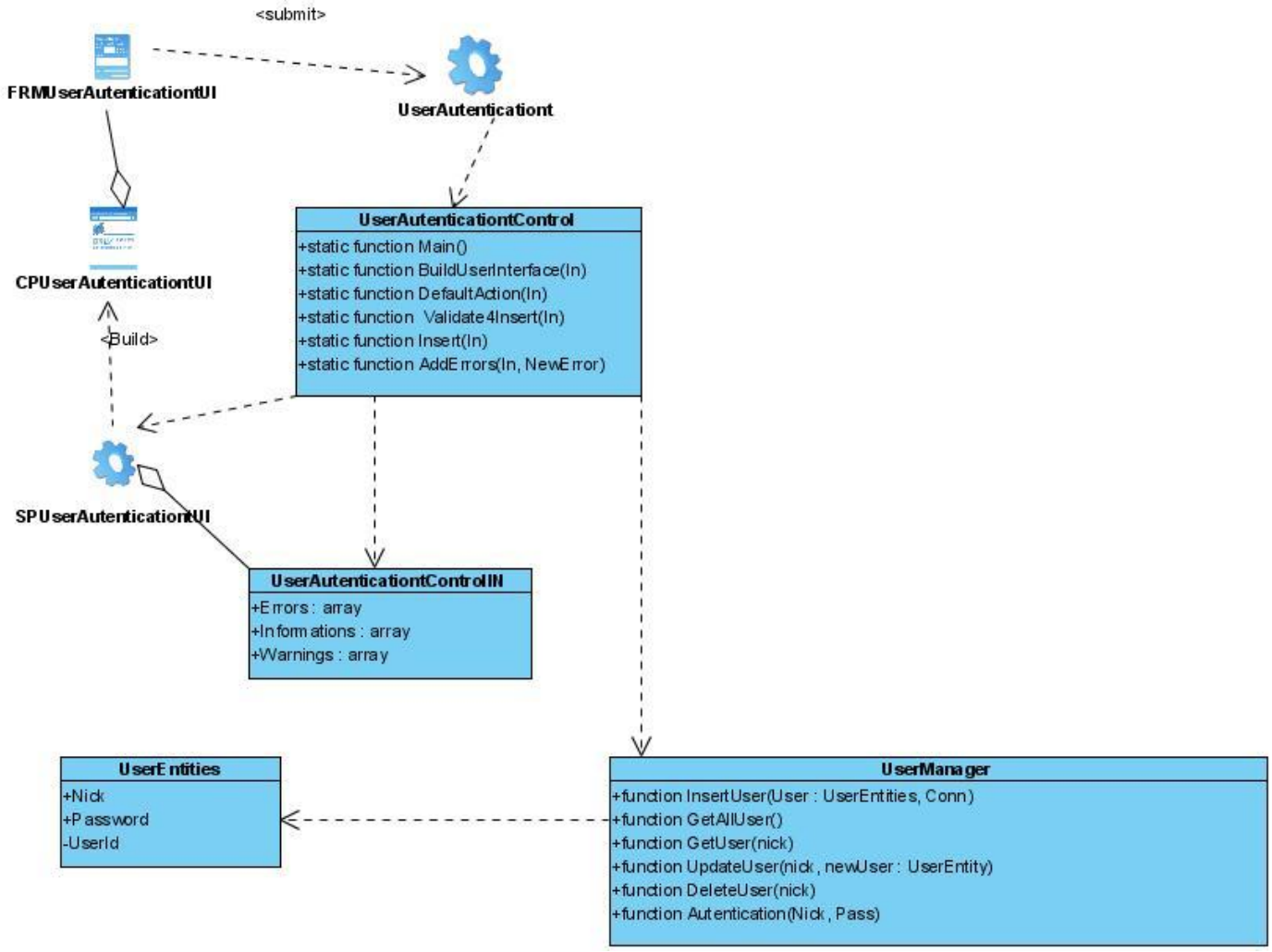

Figura 15 Diagrama de Clase del Diseño CUS Autenticar.

- **CUS Gestionar Datos Personales**

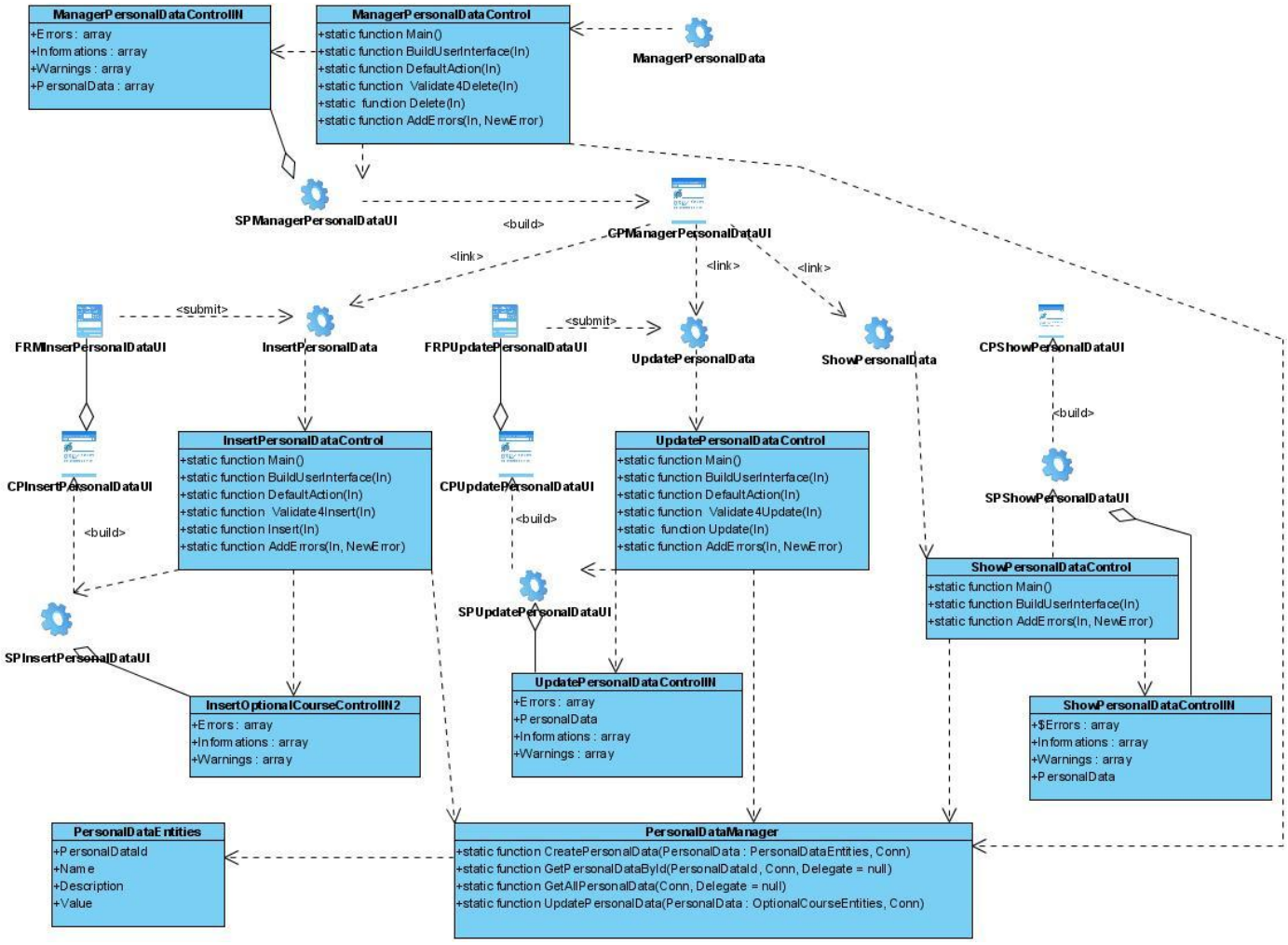

Figura 16 Diagrama de Clase del Diseño CUS Gestionar Datos Personales.

- **CUS Ver Evaluaciones**

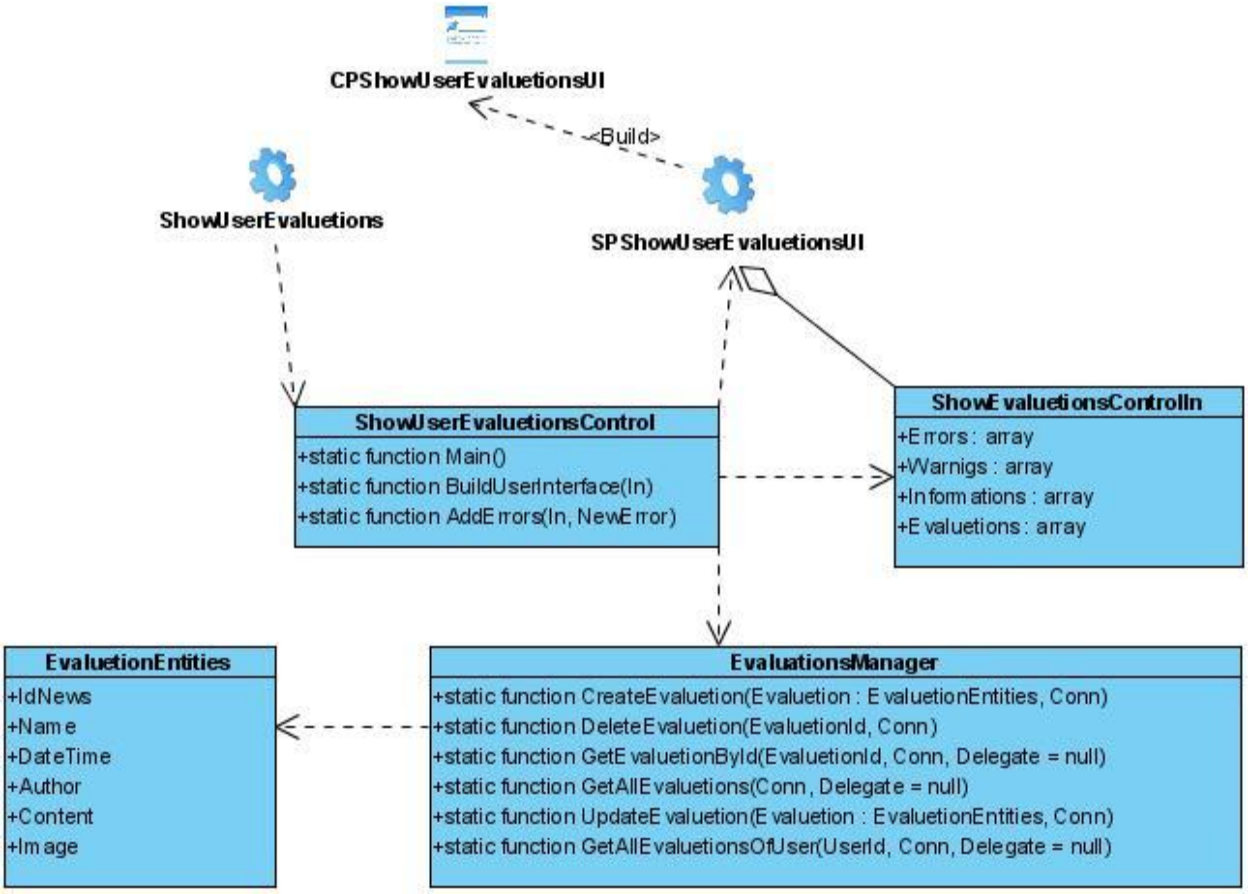

Figura 17 Diagrama de Clase del Diseño CUS Ver Evaluación

- **CUS Ver Tiempo de Máquina Asignado**

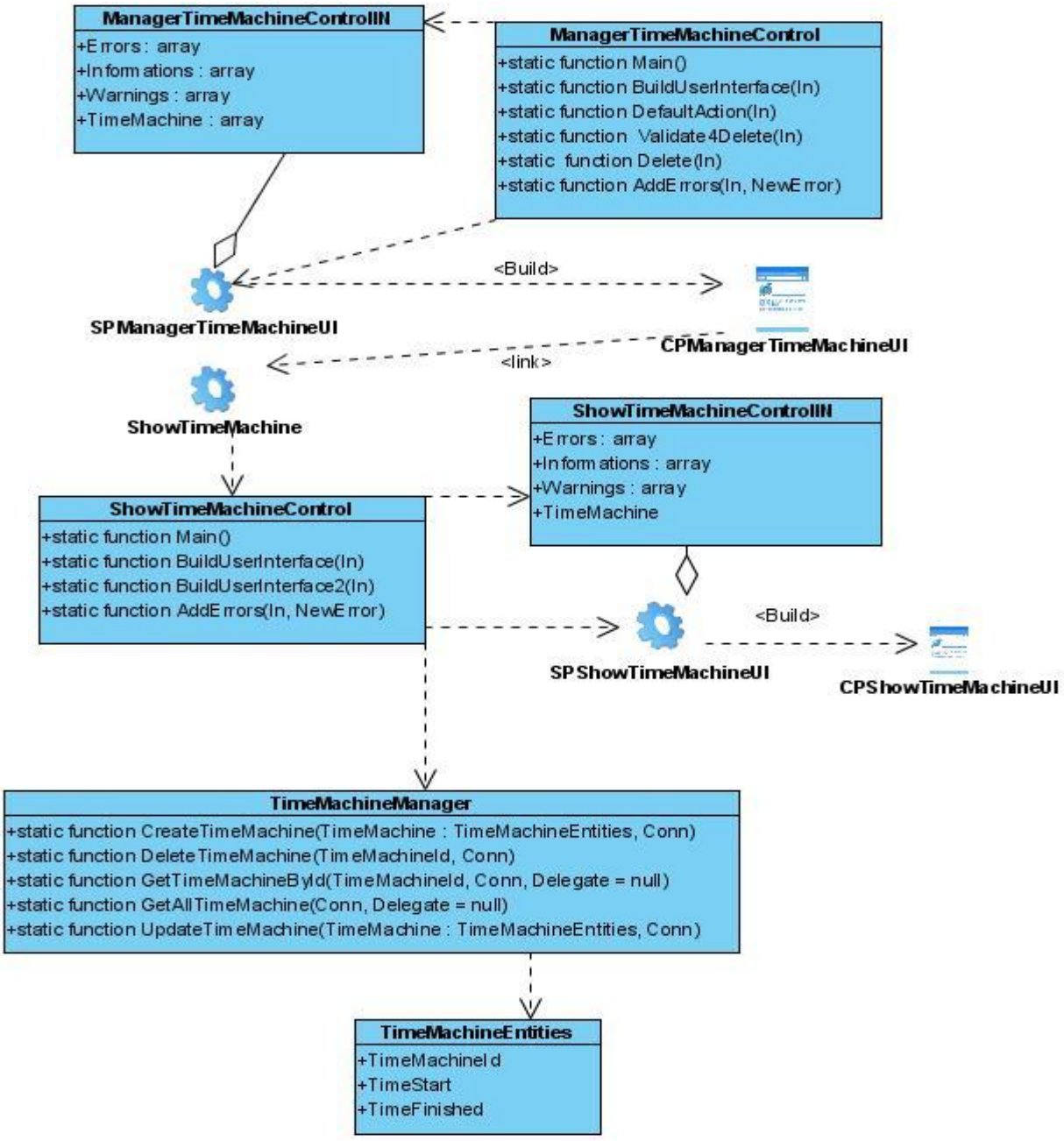

Figura 18 Diagrama de Clase del Diseño CUS Ver Tiempo de Máquina Asignado.

- **CUS Ver Tareas Asignadas**

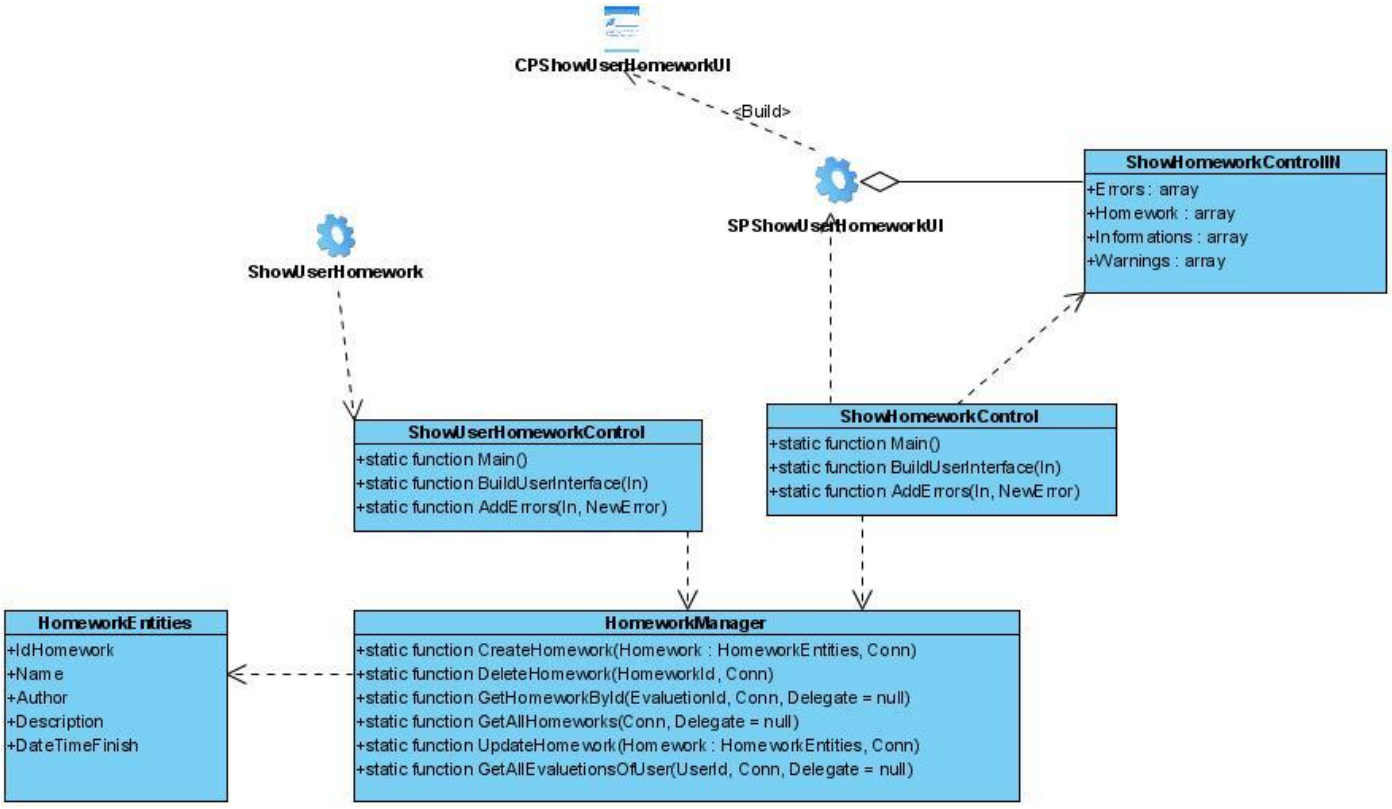

Figura 19 Diagrama de Clase del Diseño CUS Gestionar Tareas Asignadas

- **CUS Gestionar Dato a Pedir**

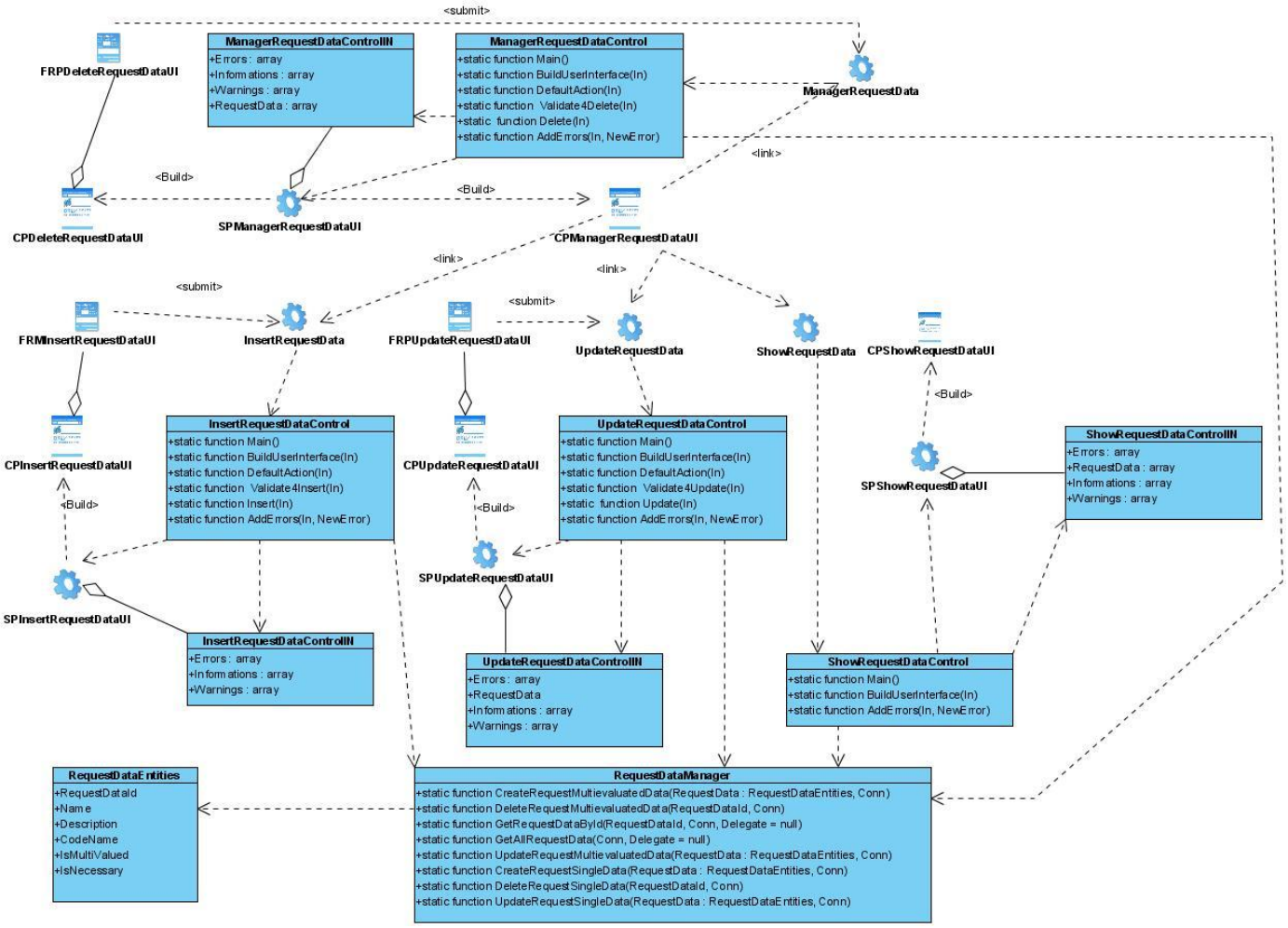

Figura 20 Diagrama de Clase del Diseño CUS Gestionar Dato a Pedir.

- **CUS Gestionar Rol**

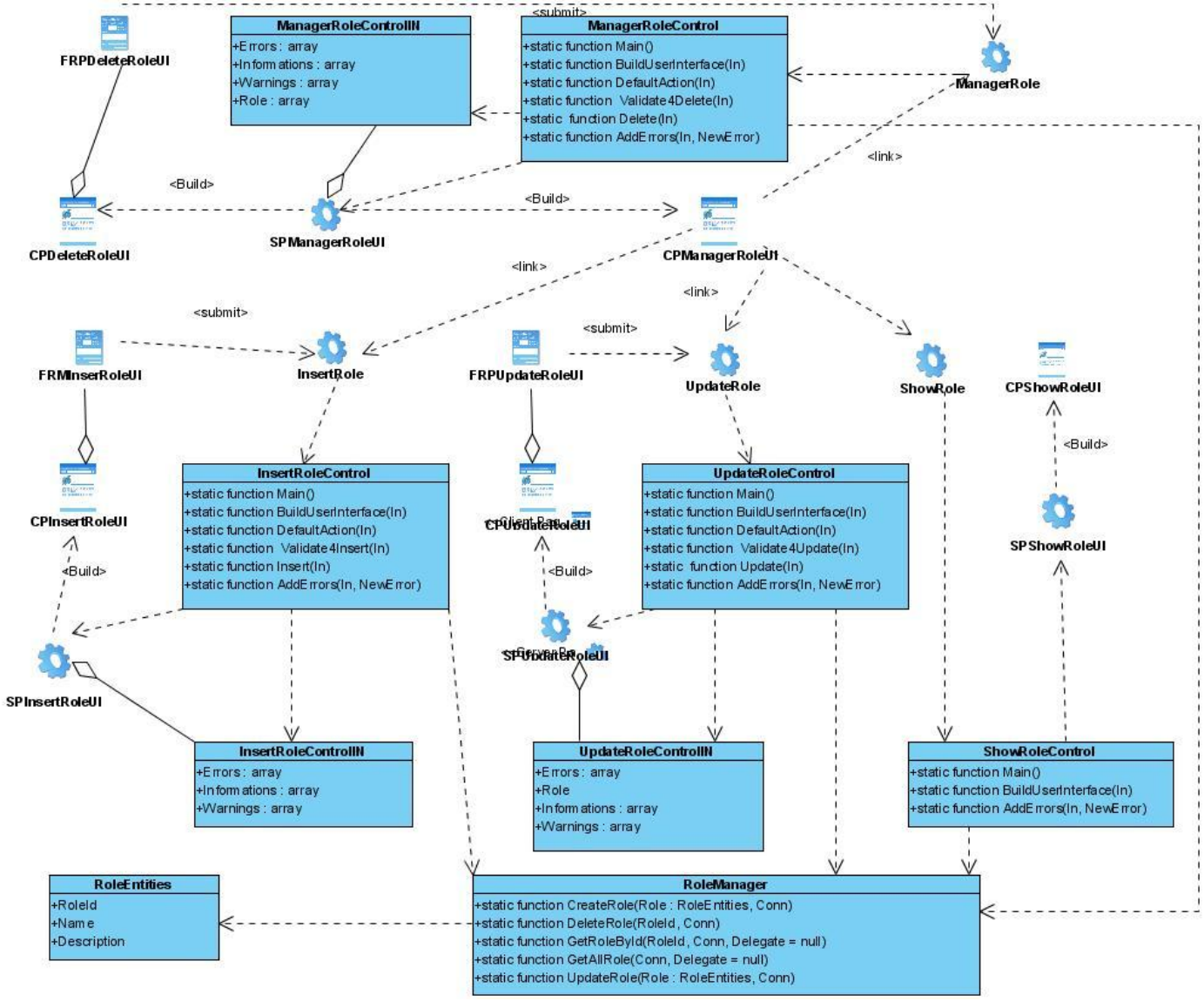

Figura 21 Diagrama de Clase del Diseño CUS Gestionar Rol.

- **CUS Gestionar Usuario**

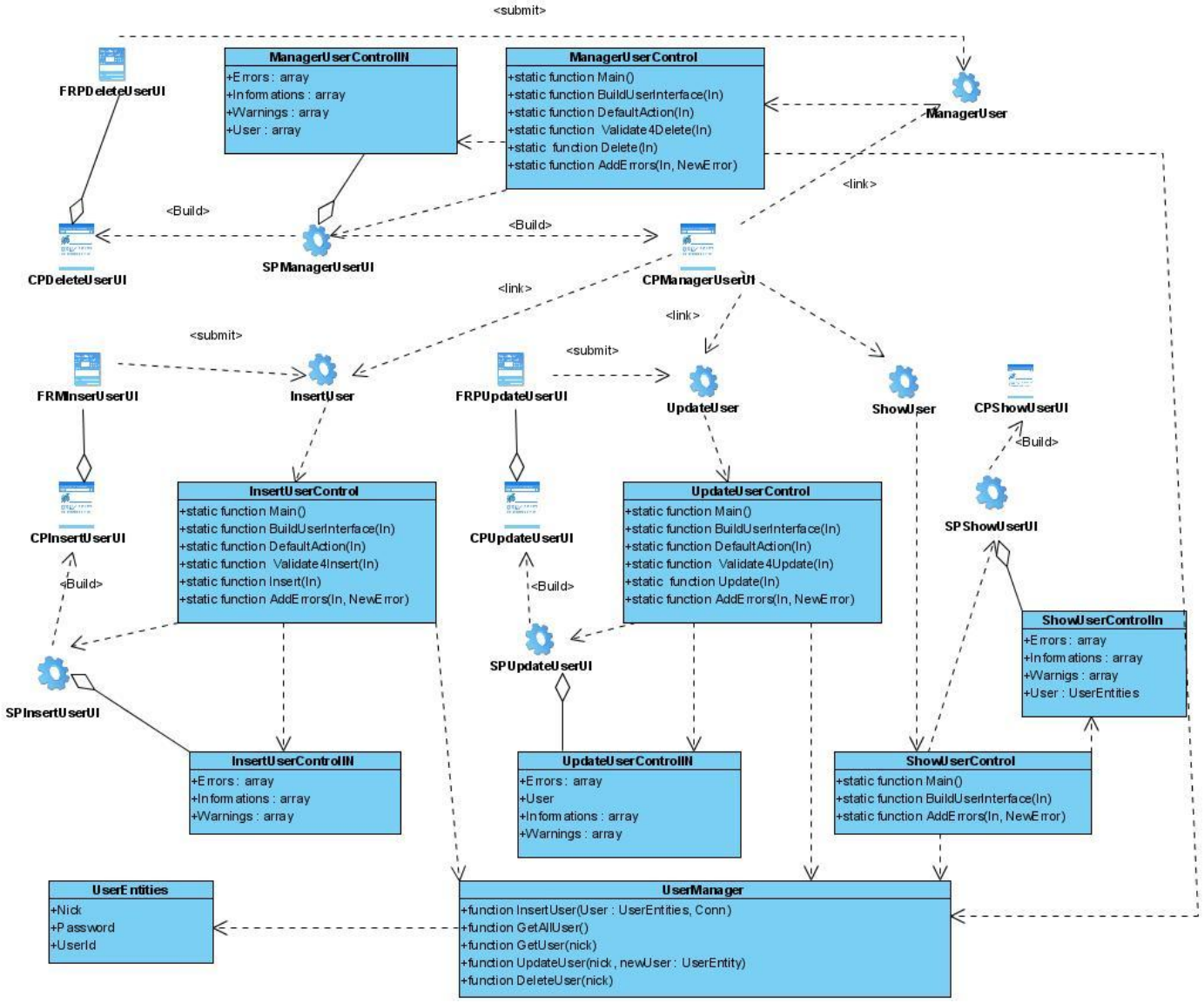

Figura 22 Diagrama de Clase del Diseño CUS Gestionar Usuario.

- **CUS Generar Reporte**

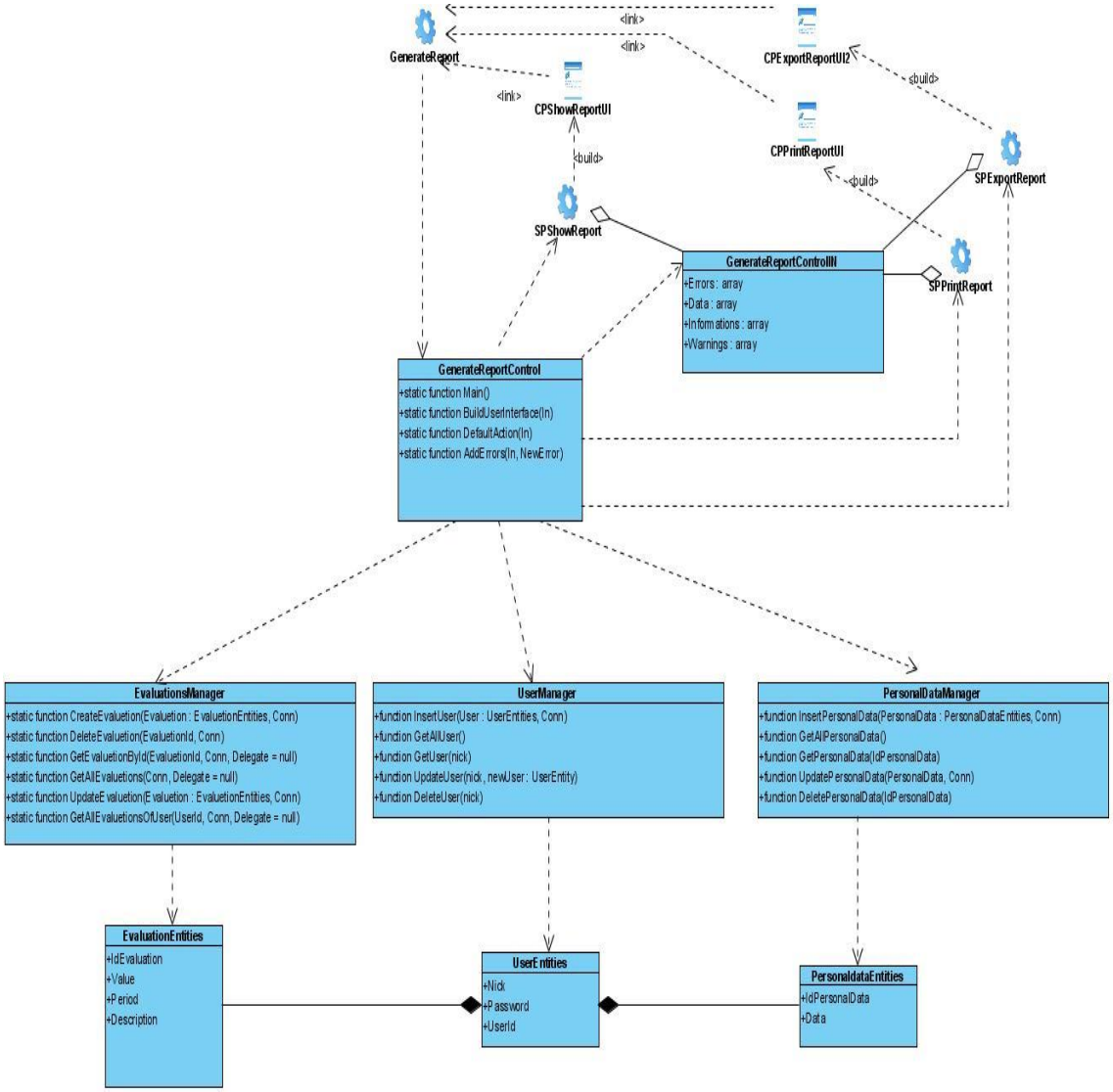

Figura 23 Diagrama de Clases del Diseño CUS Generar Reporte.

- **CUS Gestionar Tiempo de Máquina**

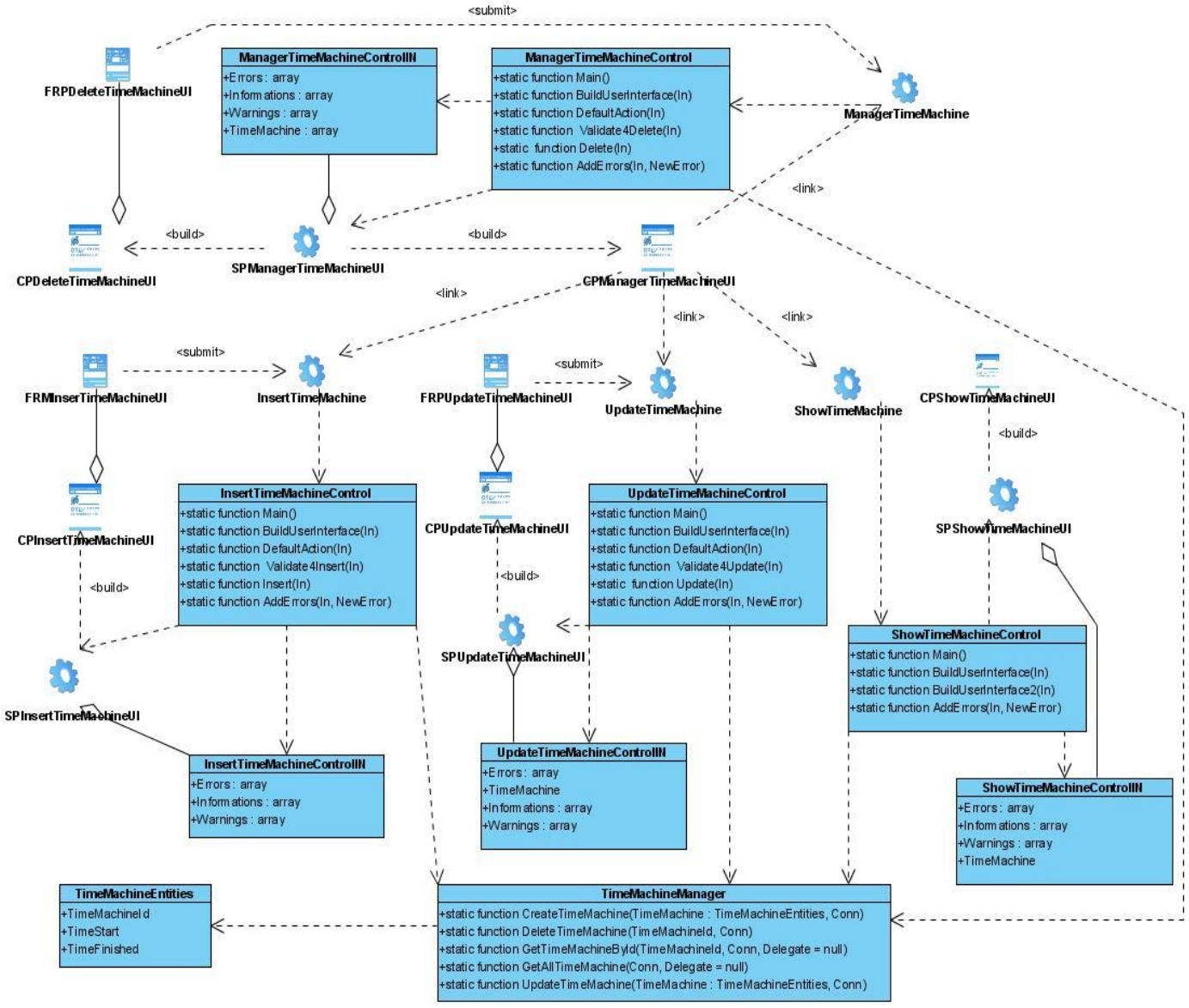

Figura 24 Diagrama de Clase del Diseño CUS Gestionar Tiempo de Máquina.

- **CUS Gestionar Tarea**

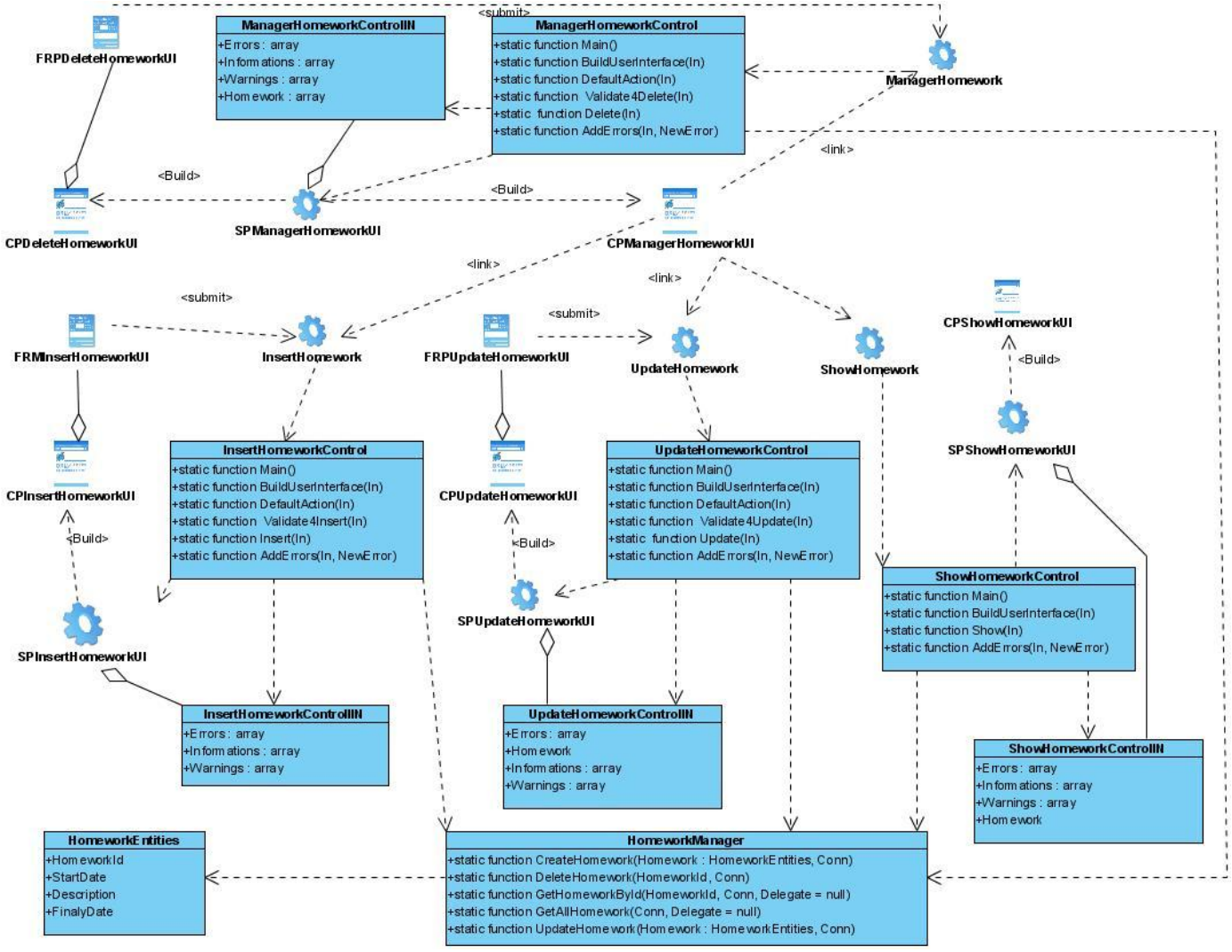

Figura 25 Diagrama de Clase del Diseño CUS Gestionar Tarea.

- **CUS Gestionar Evaluación**

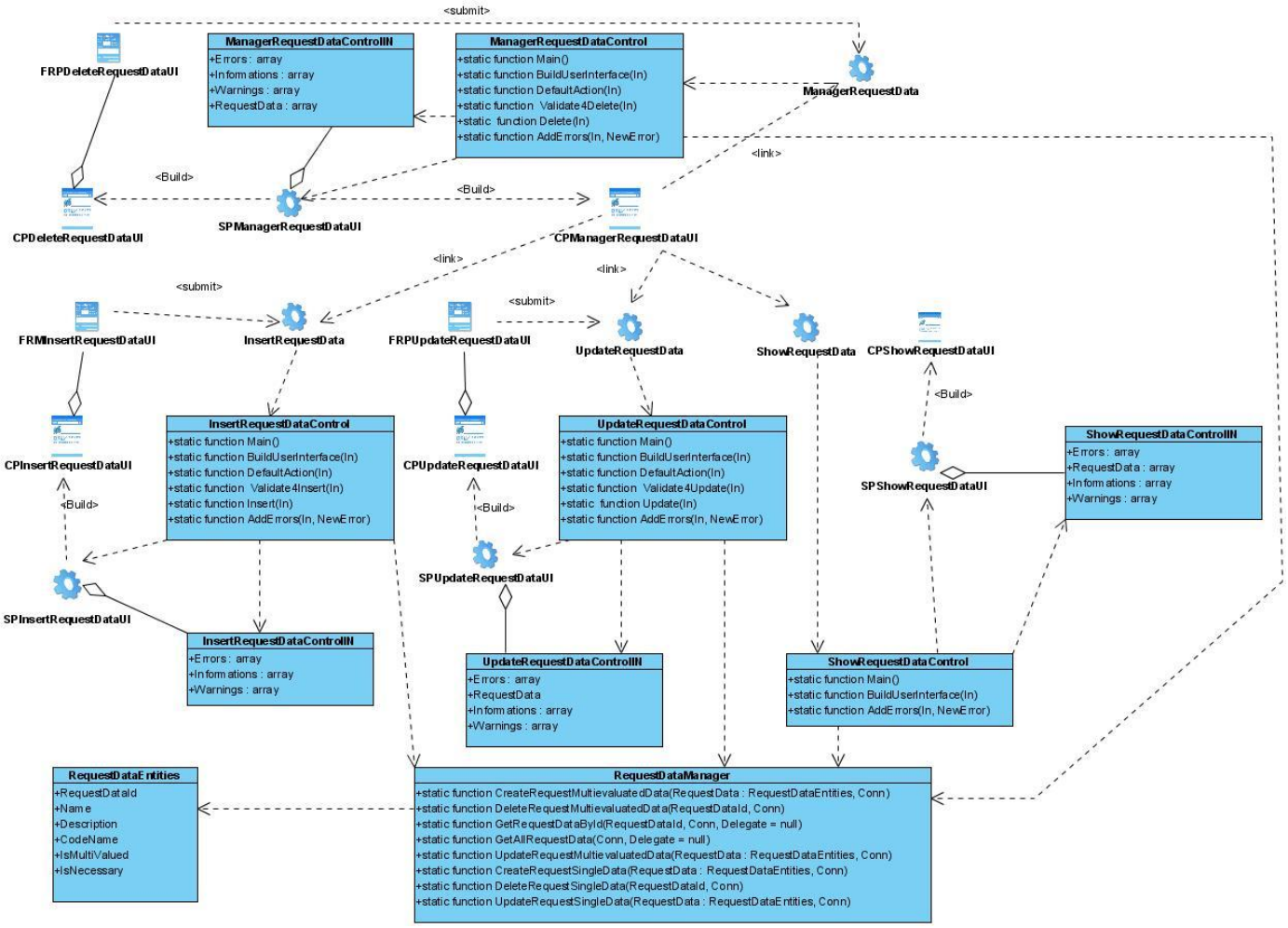

Figura 26 Diagrama de Clase del Diseño CUS Gestionar Evaluación

- **CUS Gestionar Polo**

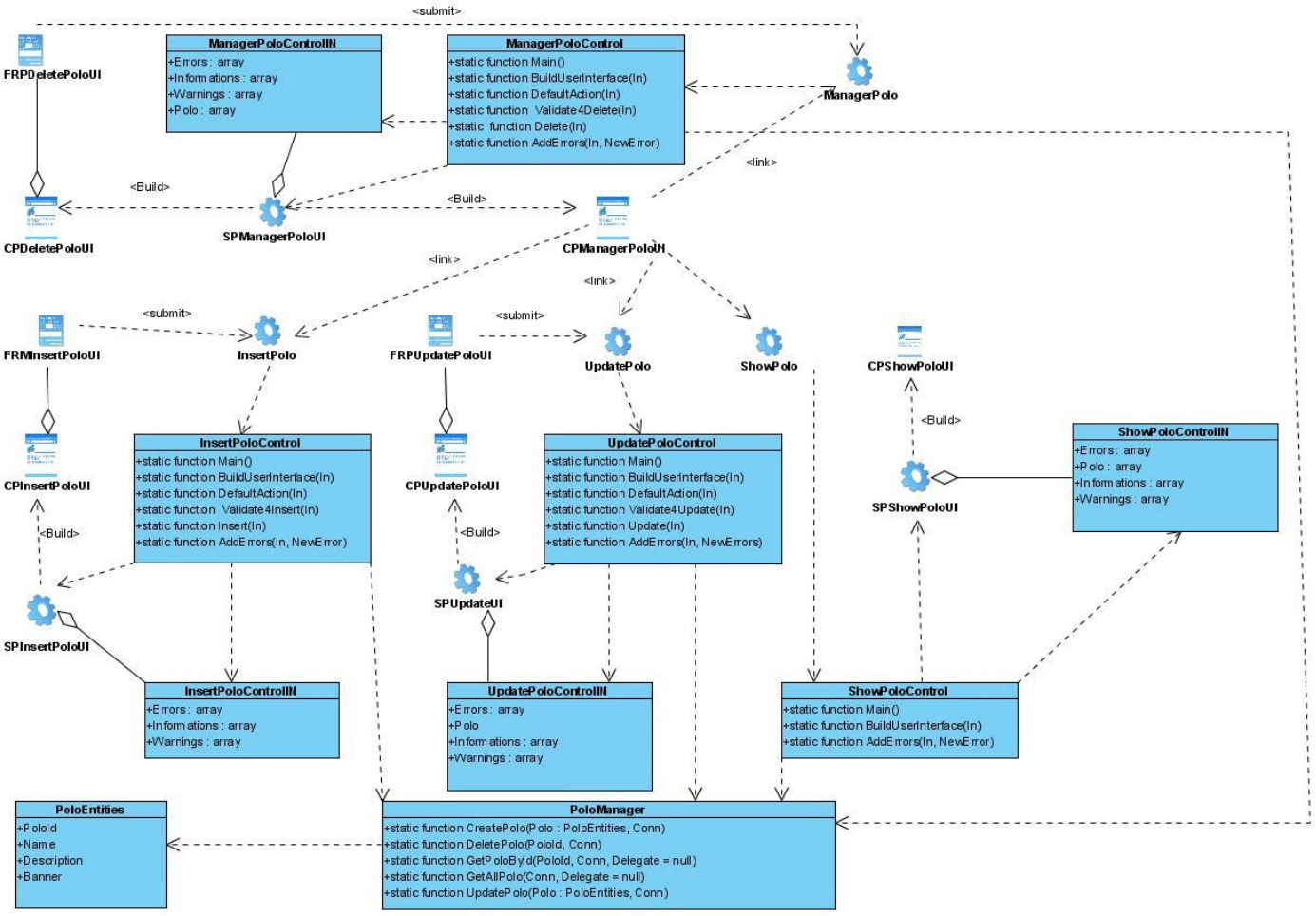

Figura 27 Diagrama de Clase del Diseño CUS Gestionar Polo.

2.4.3 Vista implementación.

En la implementación se empezará con el resultado del diseño y se implementará el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. El flujo de trabajo de diseño se propone crear un plano del modelo de implementación, por lo que sus últimas actividades están vinculadas a la creación del modelo de despliegue. El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue [11].

Los diagramas de despliegue y componentes conforman lo que se conoce como un modelo de implementación al describir los componentes a construir y su organización y dependencia entre nodos físicos en los que funcionará la aplicación [11].

El flujo de implementación está fuertemente determinado por el lenguaje de programación. El mismo tiene bosquejos con el Modelo de Diseño. Se mantendrá la misma distribución en subsistema propuesta lo único que las clases de cada unos de los casos de uso encapsuladas en los subsistemas estarán enfocadas a componentes, esta distribución por componentes se puede realizar de varias maneras, entre ellas se encuentran la distribución de componentes por clases de cada caso de usos de los subsistemas pero ubicadas en cada una de las partes que conforman el Modelo Lógico, la descomposición en componentes solo se realizará teniendo en cuenta los grupos de clases agrupados en cada subsistemas para cada caso de Uso , así la dependencia será estricta entre los componentes y quedará mejor definida.

Diagramas de Componentes de los CU Arquitectónicamente Significativos

La aplicación que se desarrollará está dividida en cuatro partes fundamentales model, view, control y global. Además esta aplicación agrupa entre sus partes ocho subsistema definidos y explicados en la vista lógica de la arquitectura. A continuación se hará referencia a como quedan distribuidas las clases del sistema en términos de componentes por cada una de las cuatro partes en que se distribuye la aplicación para cada uno de los casos de uso arquitectónicamente significativos.

- **Caso de Uso Autenticar**

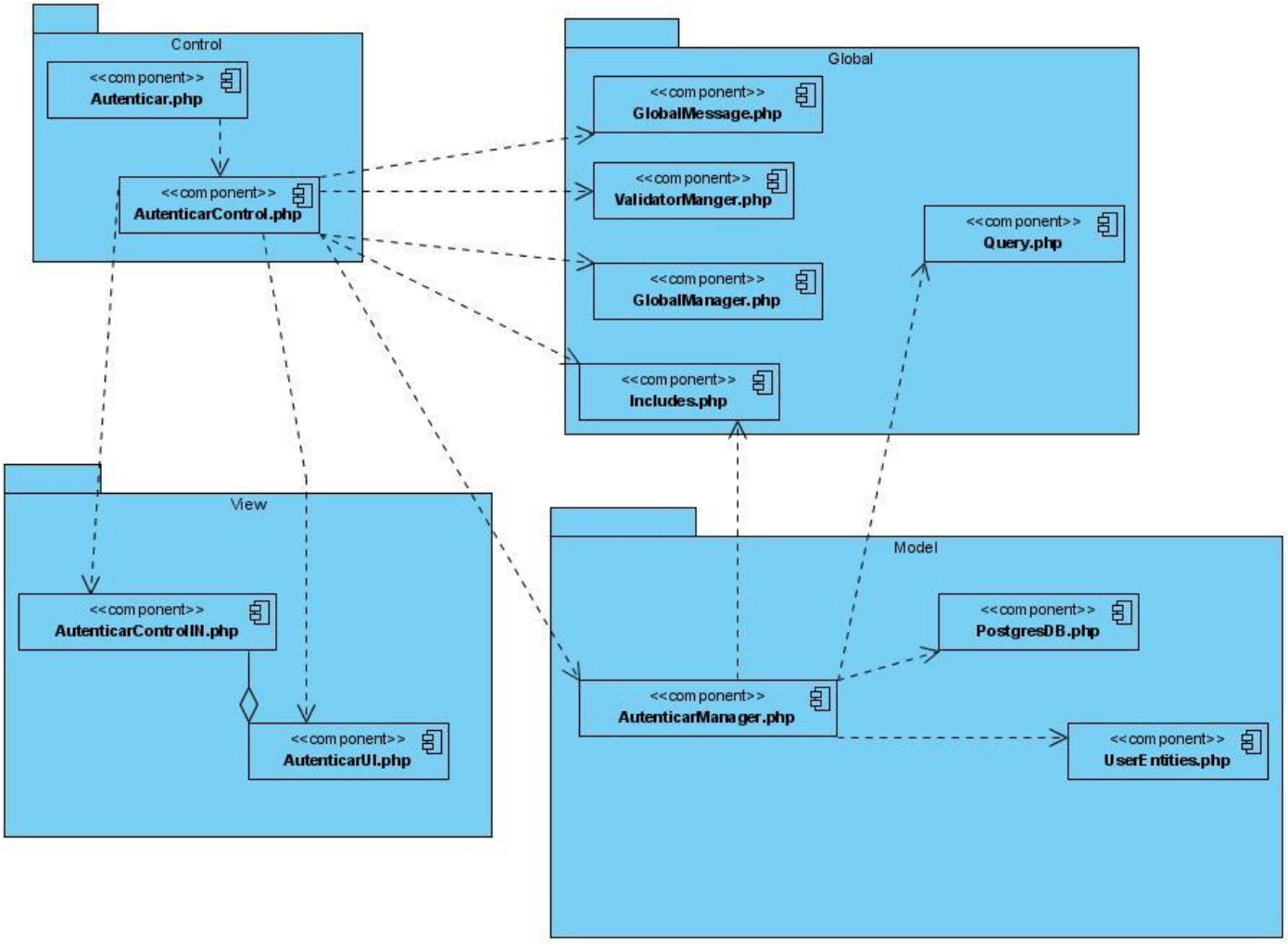

Figura 28 Diagrama de Componente CU Autenticar Usuario

- **CUS Gestionar Tiempo de Máquina**

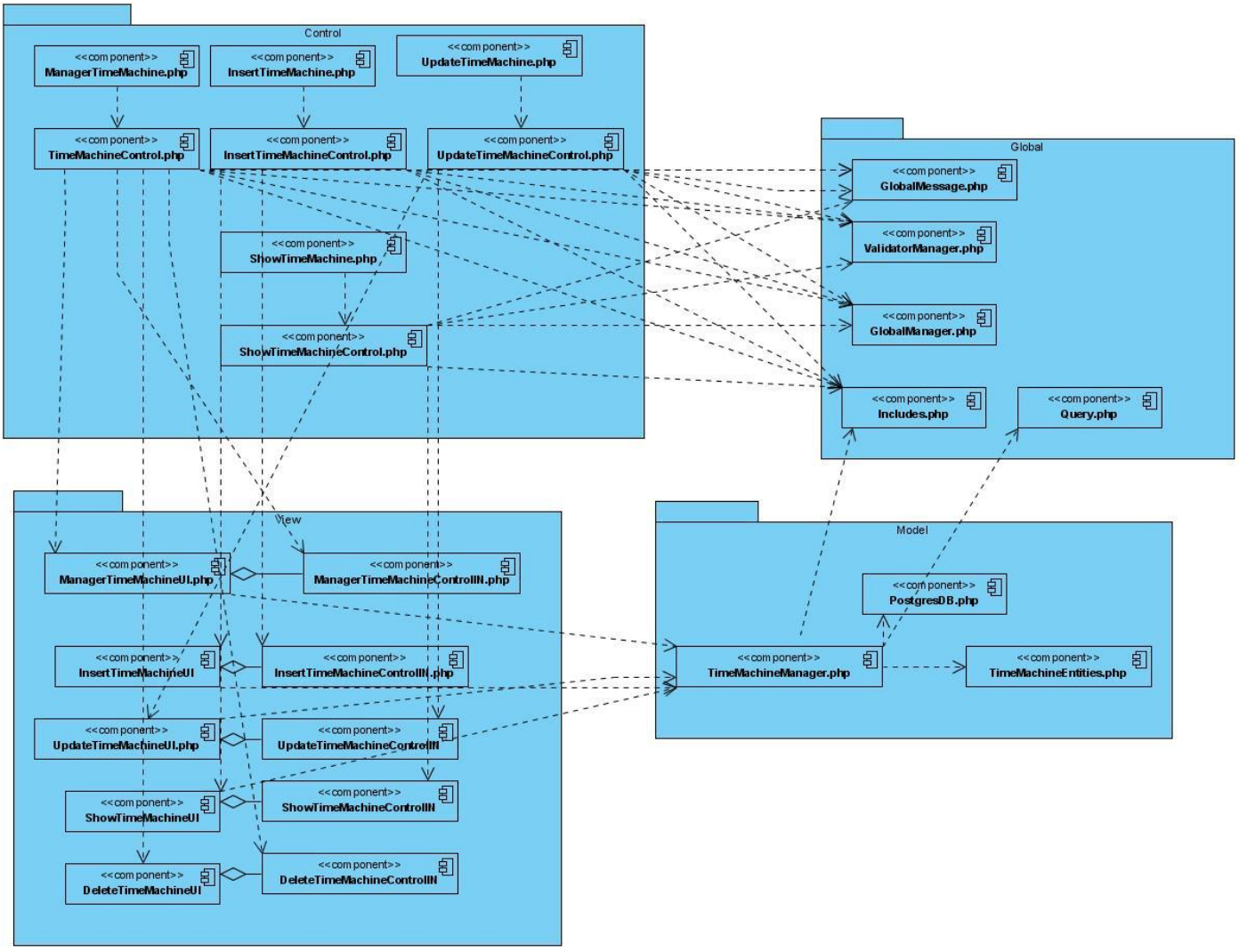

Figura 29 Diagrama Componente CU Gestionar Tiempo Máquina

- **CUS Gestionar Tarea**

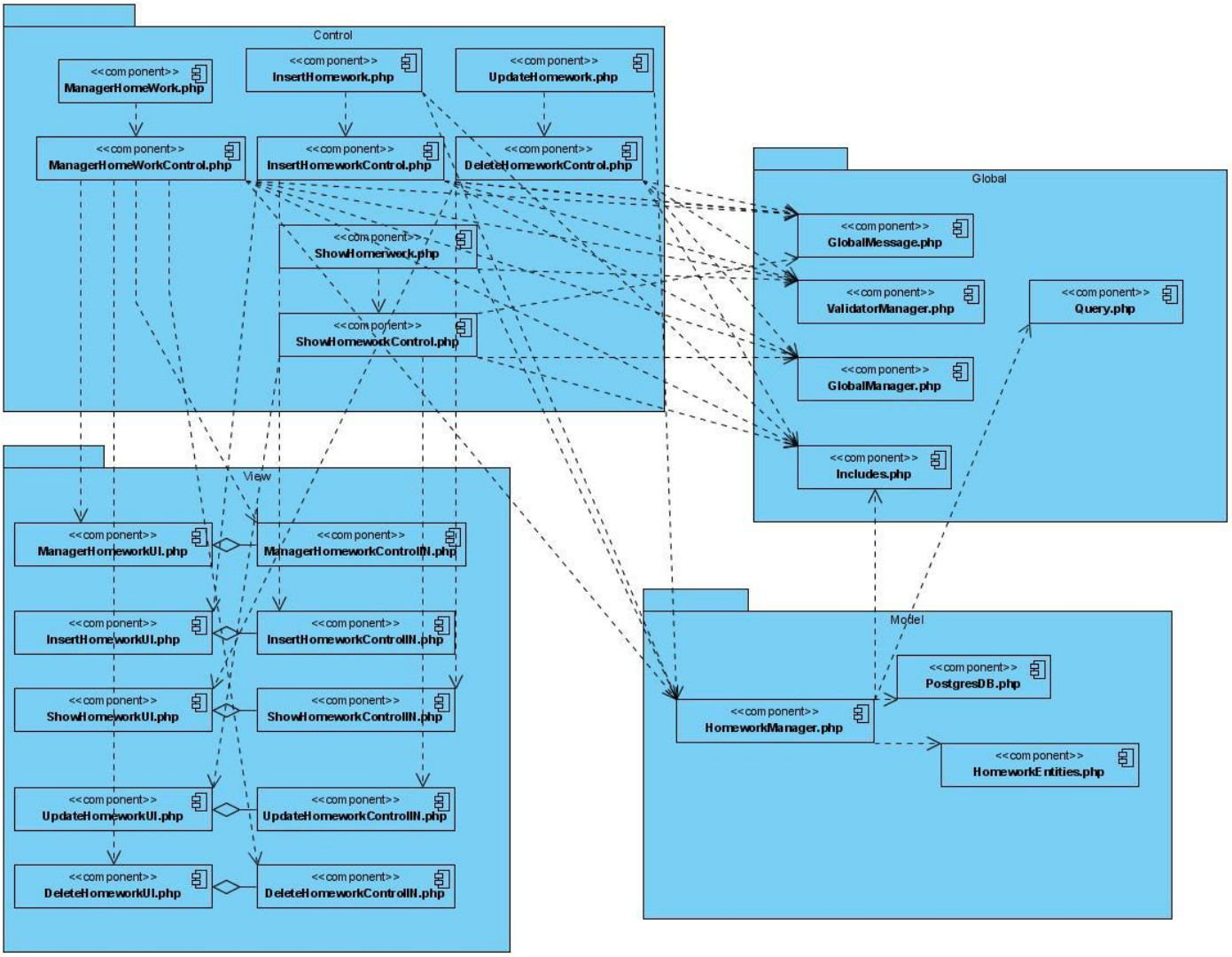

Figura 30 Diagrama Componente CU Gestionar Tarea

- **CUS Gestionar Evaluación**

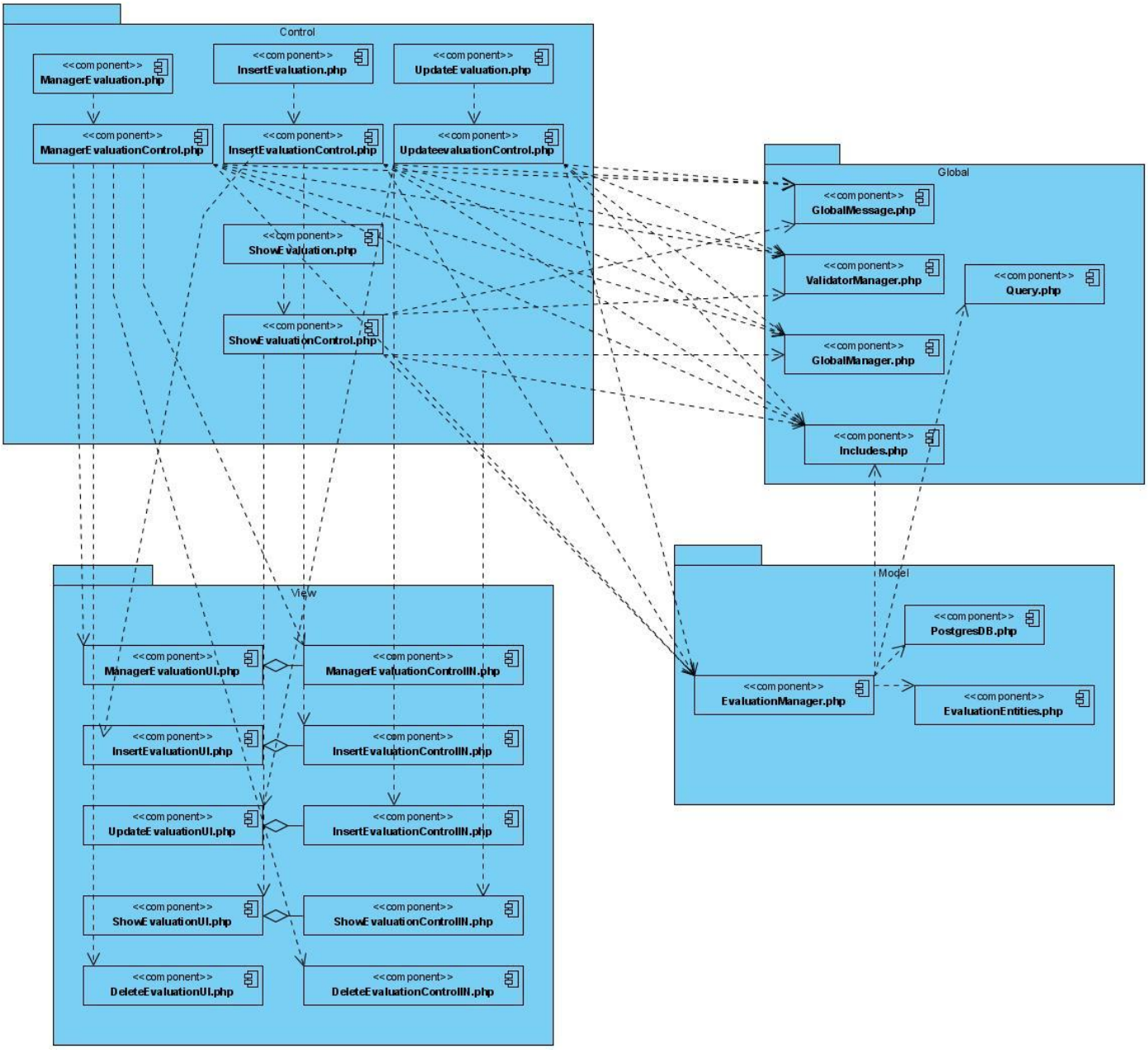

Figura 31 Diagrama de Componente CU Gestionar Evaluación

- **CUS Gestionar Polo**

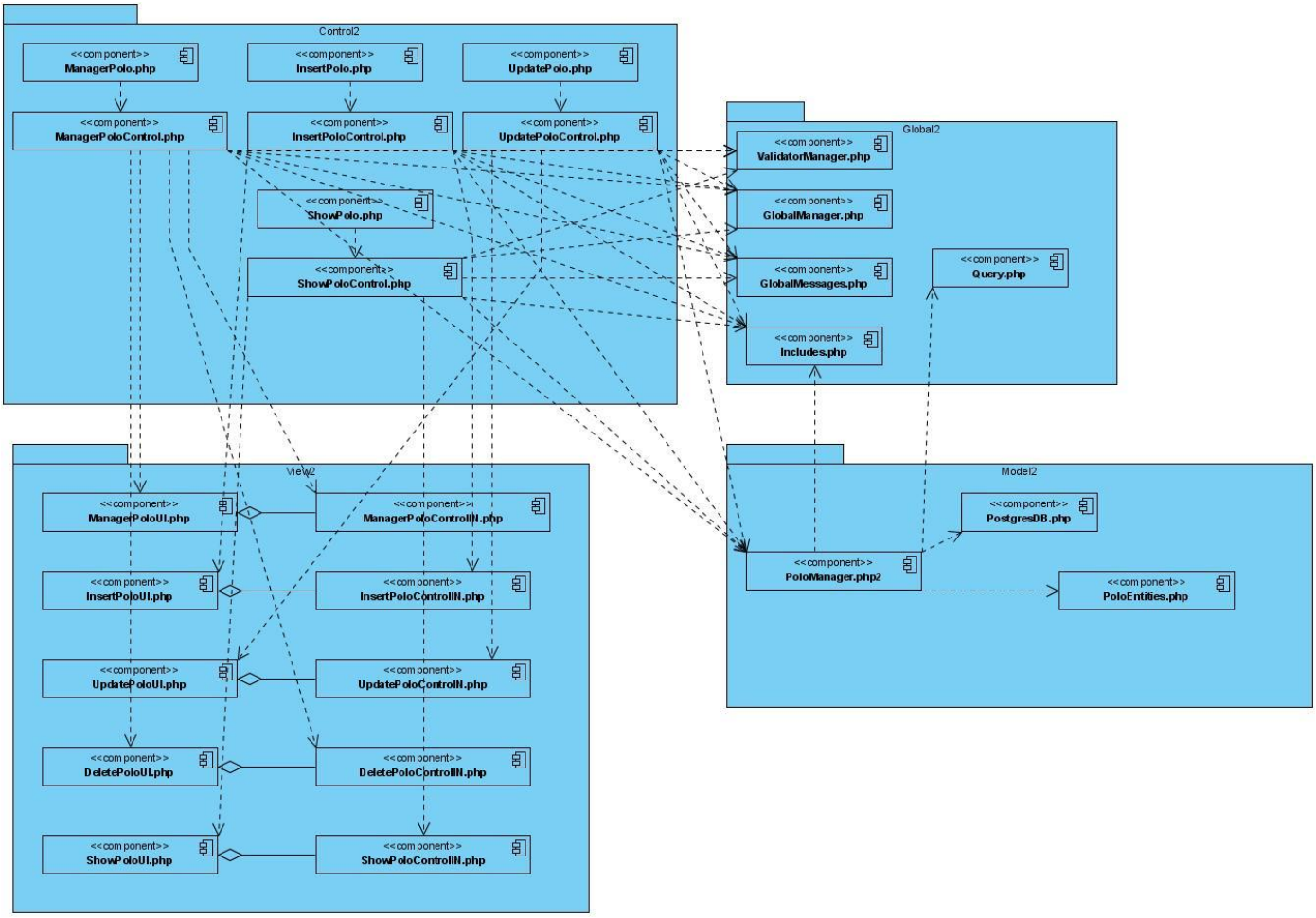

Figura 32 Diagrama de Componente CU Gestionar Polo

- **CUS Generar Reporte**

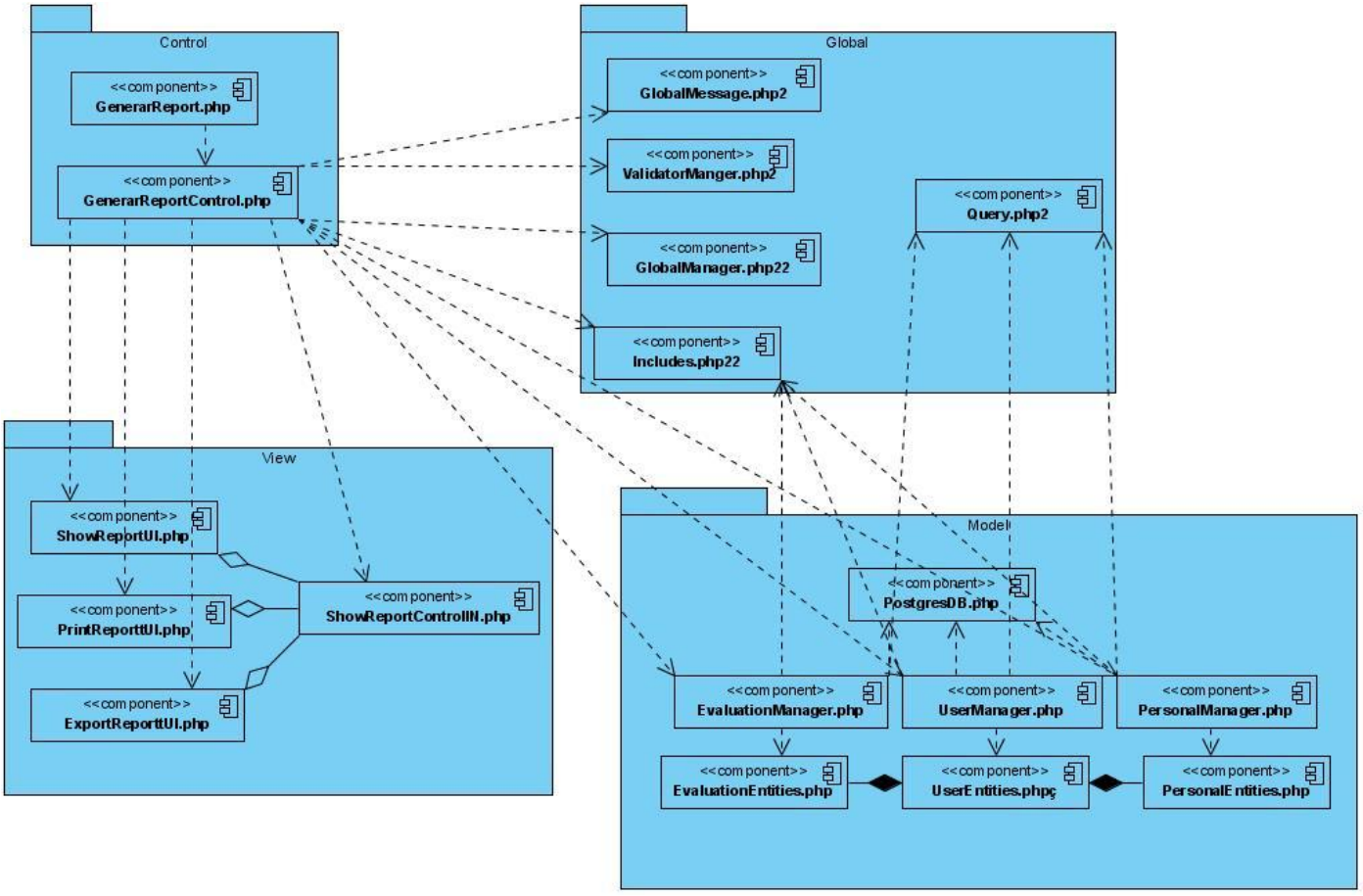

Figura 33 Diagrama de Componente CU Generar Reporte

- **CUS Gestionar Usuario**

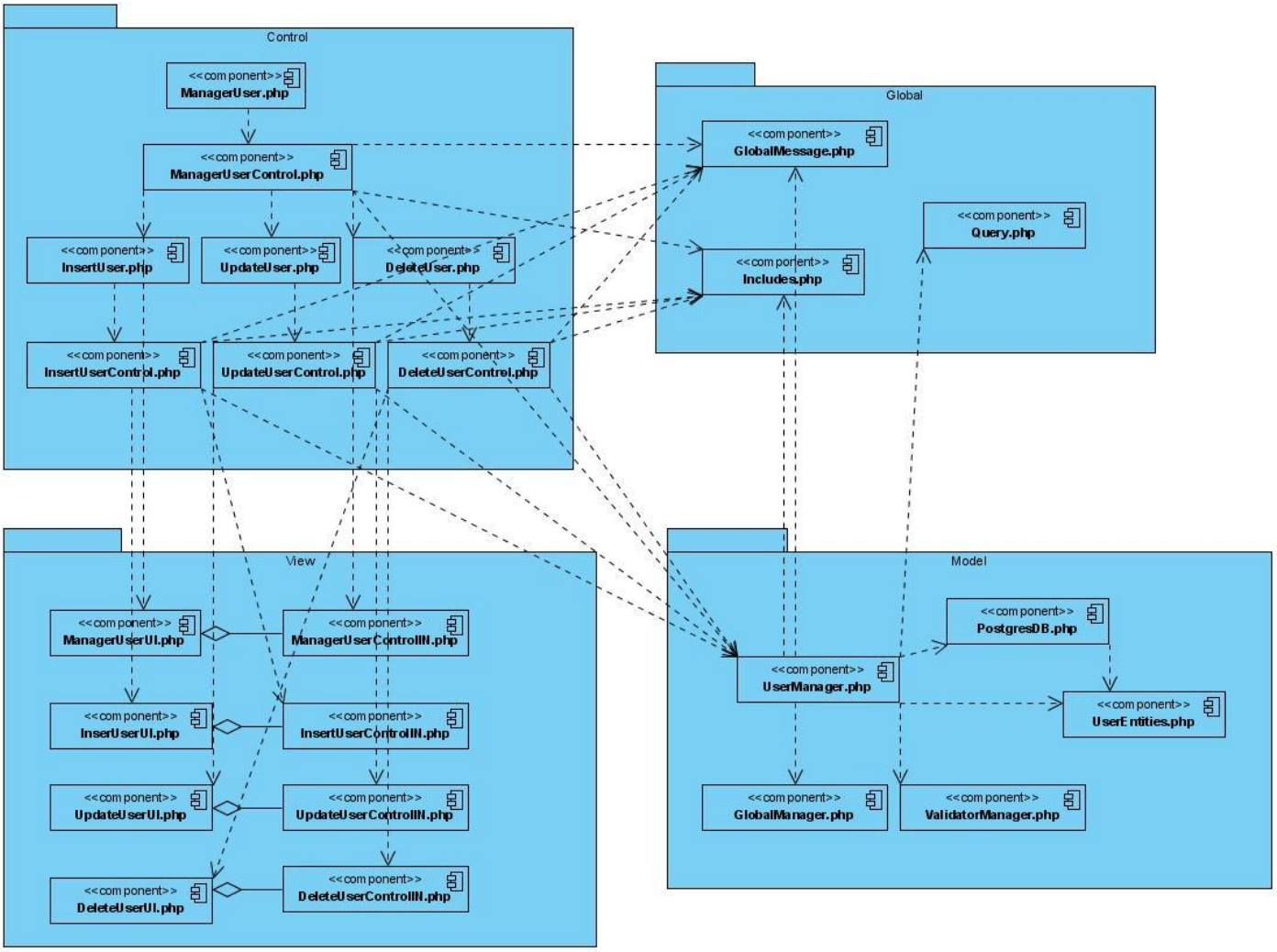

Figura 34 Diagrama de Componente CU Gestionar Usuario

- **CUS Gestionar Rol**

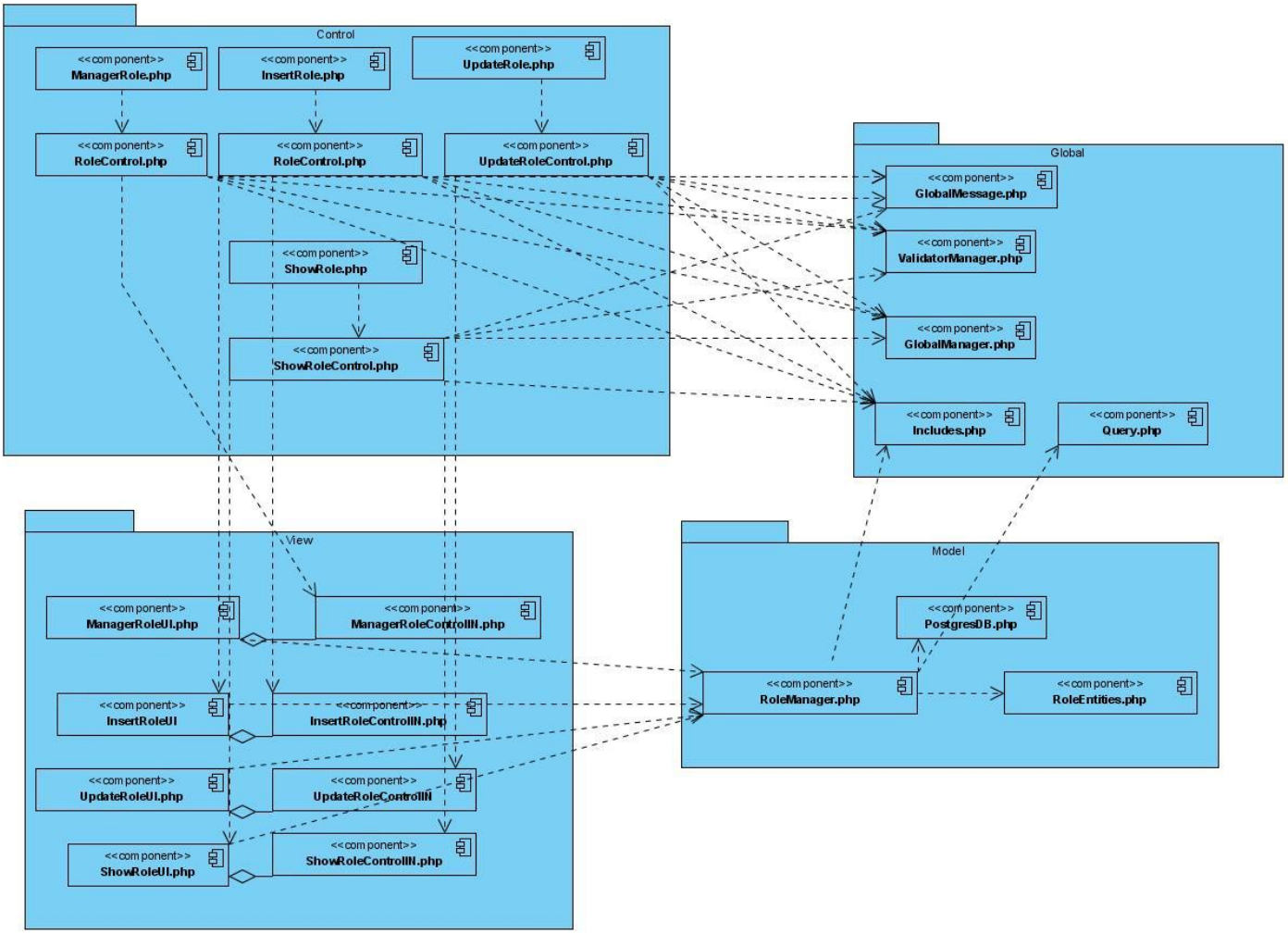

Figura 35 Diagrama de Componente CU Gestionar Rol

- **CUS Gestionar Dato a Pedir**

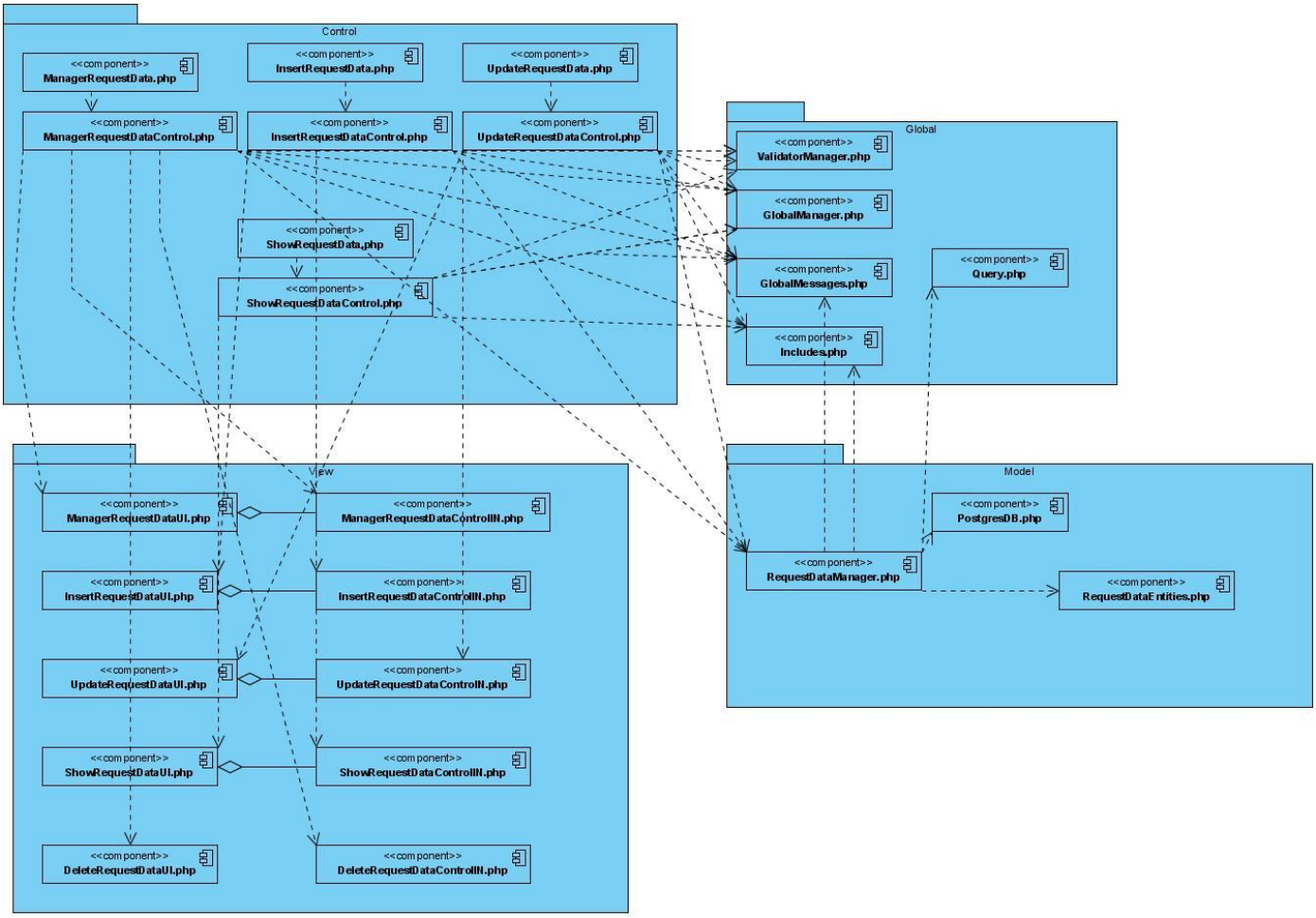

Figura 36 Diagrama de Componente CU Gestionar Datos a Pedir

- **CUS Ver Tareas Asignadas**

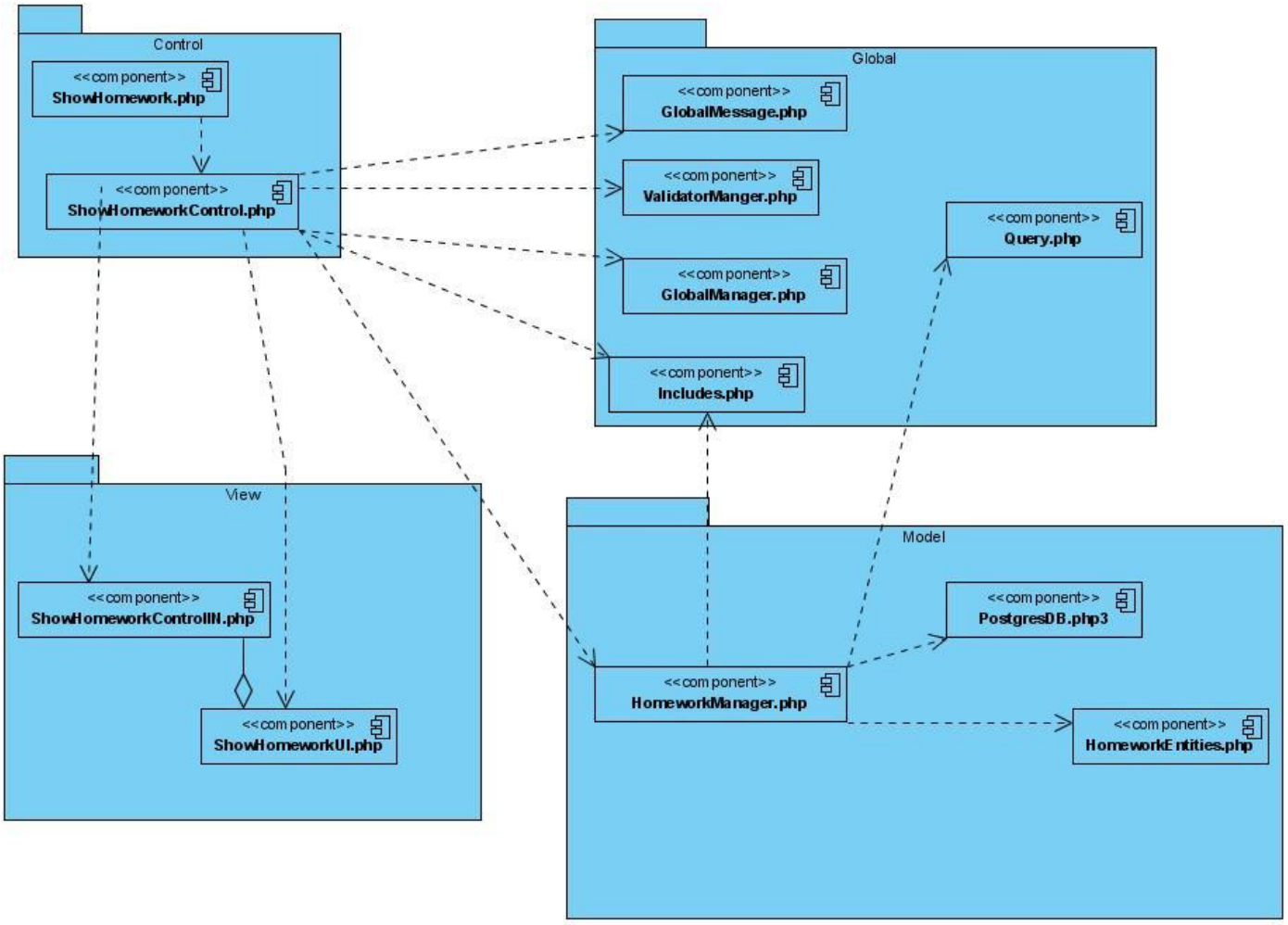

Figura 37 Diagrama de Componente CU Ver Tareas Asignadas

- **CUS Ver Tiempo de Máquina Asignado**

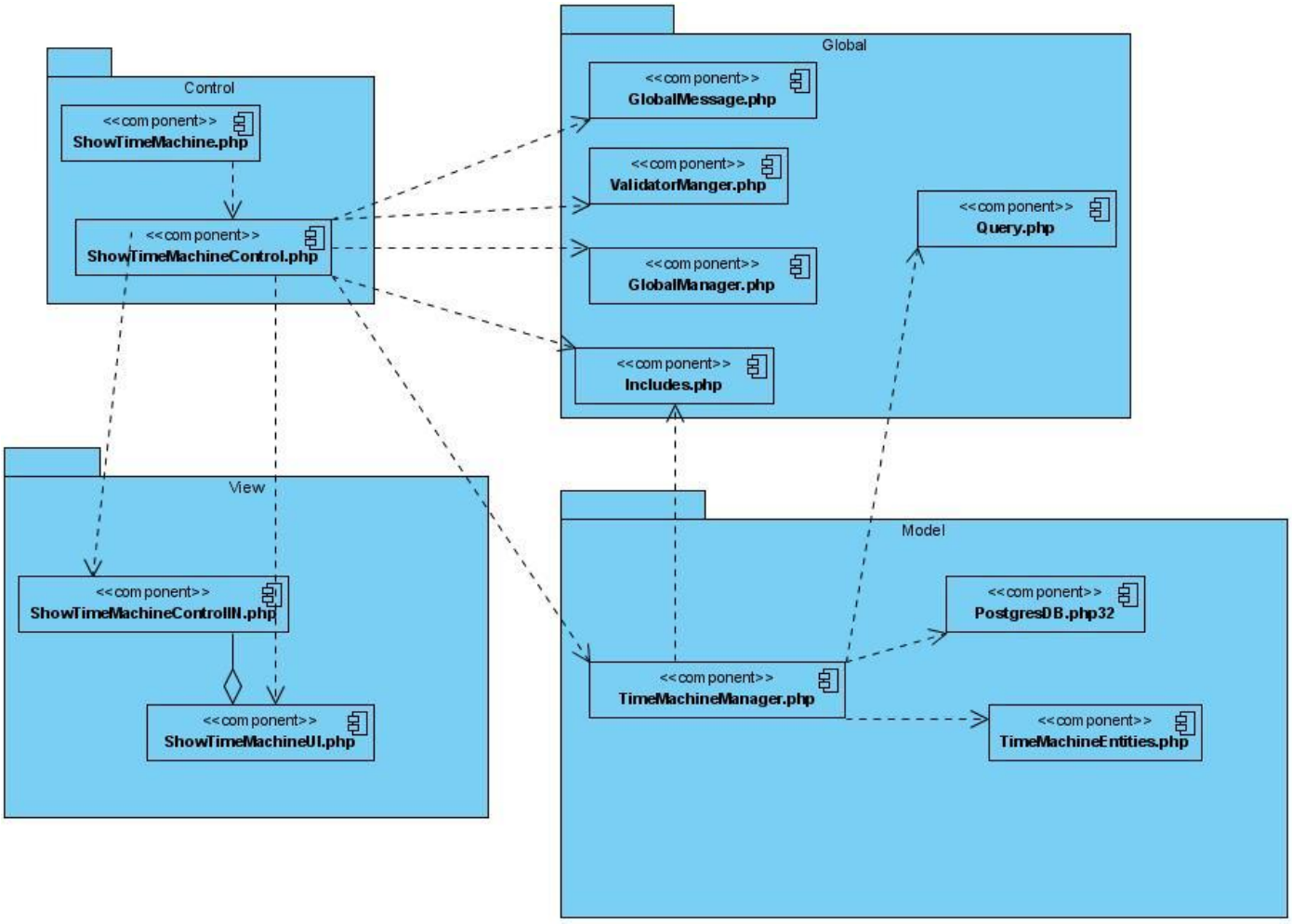

Figura 38 Diagrama de Componente CU Ver Tiempo Máquina Asignado

- **CUS Ver Evaluaciones**

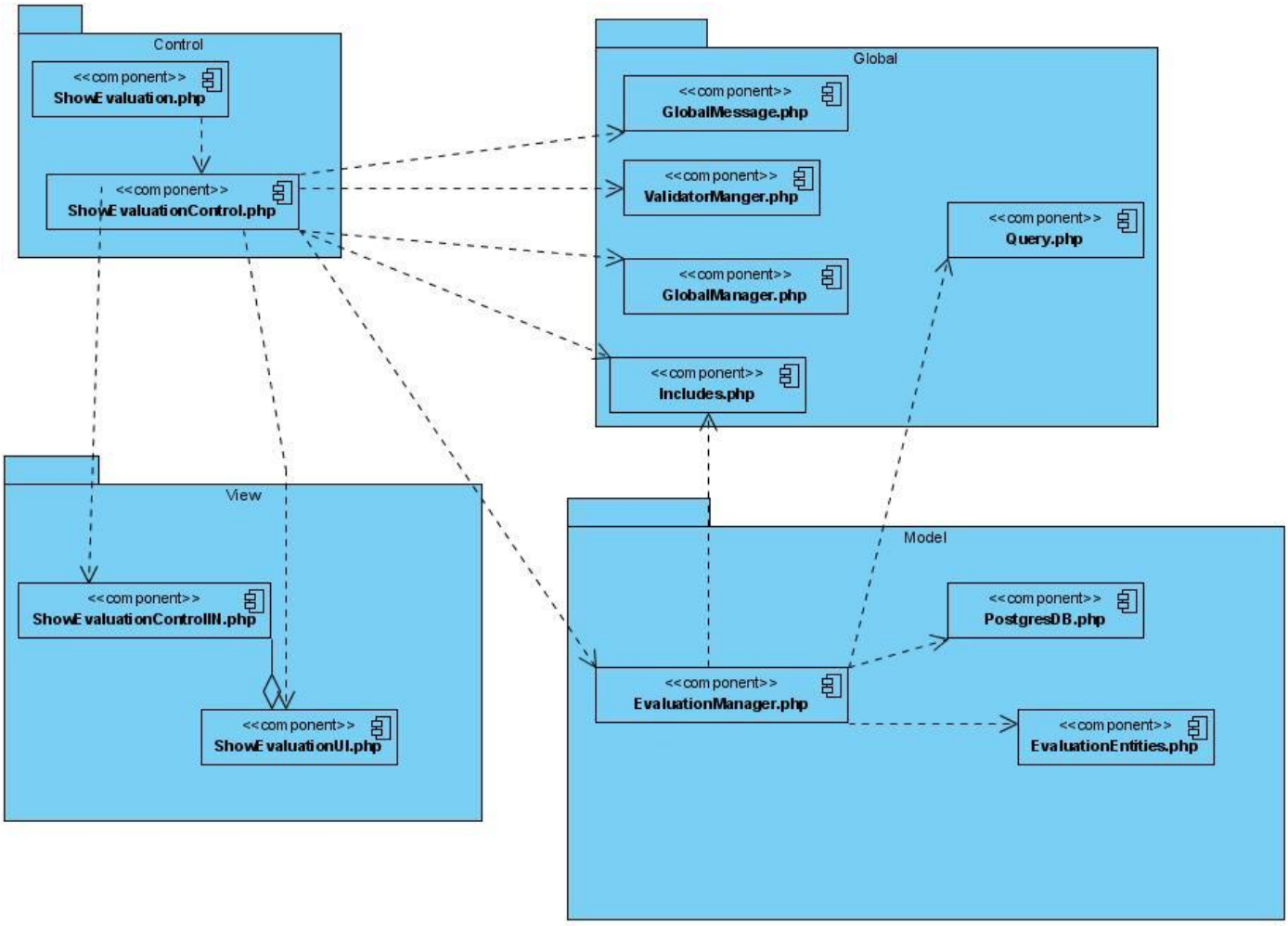

Figura 39 Diagrama de Componente CU Ver Evaluaciones Asignadas

- **CUS Gestionar Datos Personales**

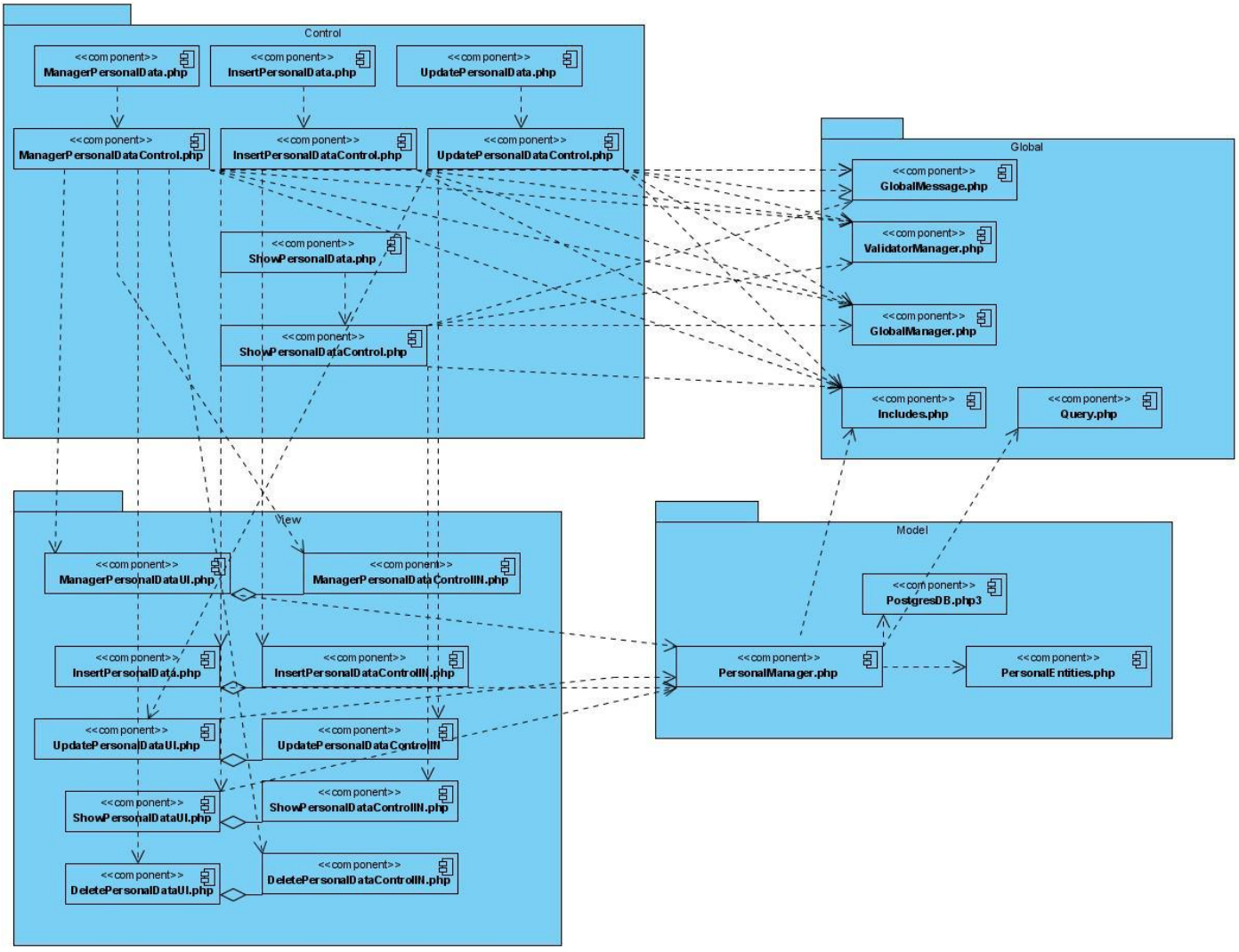

Figura 40 Diagrama de Componente CU Gestionar Datos Personales

2.4.4 Vista despliegue.

El arquitecto es responsable de considerar los aspectos referentes a la futura distribución física (del sistema, llamada además arquitectura de hardware, el hardware es un punto clave para el correcto funcionamiento del sistema.

En ocasiones las condiciones del hardware tienen que ser específicas, la configuración de red existente tiene que cumplir ciertos parámetros, hay protocolos que deben estar disponibles. Al imaginar una aplicación cliente que se conecta a una base de datos empleando ADO, si los componentes de acceso a datos no están instalados en la máquina, la petición será inefectiva, porque la máquina no está capacitada para interpretar el protocolo necesario para dar respuesta a dicha solicitud, por tanto no se podrá "entender" con el servidor de datos [11].

Al imaginar ahora que el mismo sistema deberá ser capaz de soportar un millón de peticiones a la base de datos en un segundo. Si el gestor de datos empleado es un Pentium MMX con 64 Mb de RAM, la aplicación no podrá satisfacer dichos requisitos, ya que el Procesador (computadora), seleccionada como gestor de datos, no fue el correcto. El sistema no funcionará [11].

Trabajar además con los dispositivos reales es de suma importancia, puesto que una mala selección de los mismos podría encaminar una respuesta no deseada o irrealizable por parte de la aplicación.

La vista de despliegue incluye además una representación de los principales componentes del sistema y la ubicación que tendrán esos componentes dentro de cada nodo.

2.4.4.1 Diagrama de Despliegue

La aplicación se encontrará distribuida en cinco nodos principales. El nodo PC Cliente encargado del funcionamiento del navegador Web donde será visualizada la aplicación, el nodo Servidor de Aplicación con la funcionalidad de soportar sobre él la aplicación Web, en este es donde se montará el sistema, el nodo Servidor de Base Dato que es donde estará montada la base de dato de la aplicación, un nodo denominado Servidor SOAP que es el servidor que brinda los servicios Web para la aplicación y un quinto nodo que es donde estará el servidor de nombres de dominio de la universidad. A continuación se muestra como quedan distribuidos estos nodos y sus relaciones de comunicación.

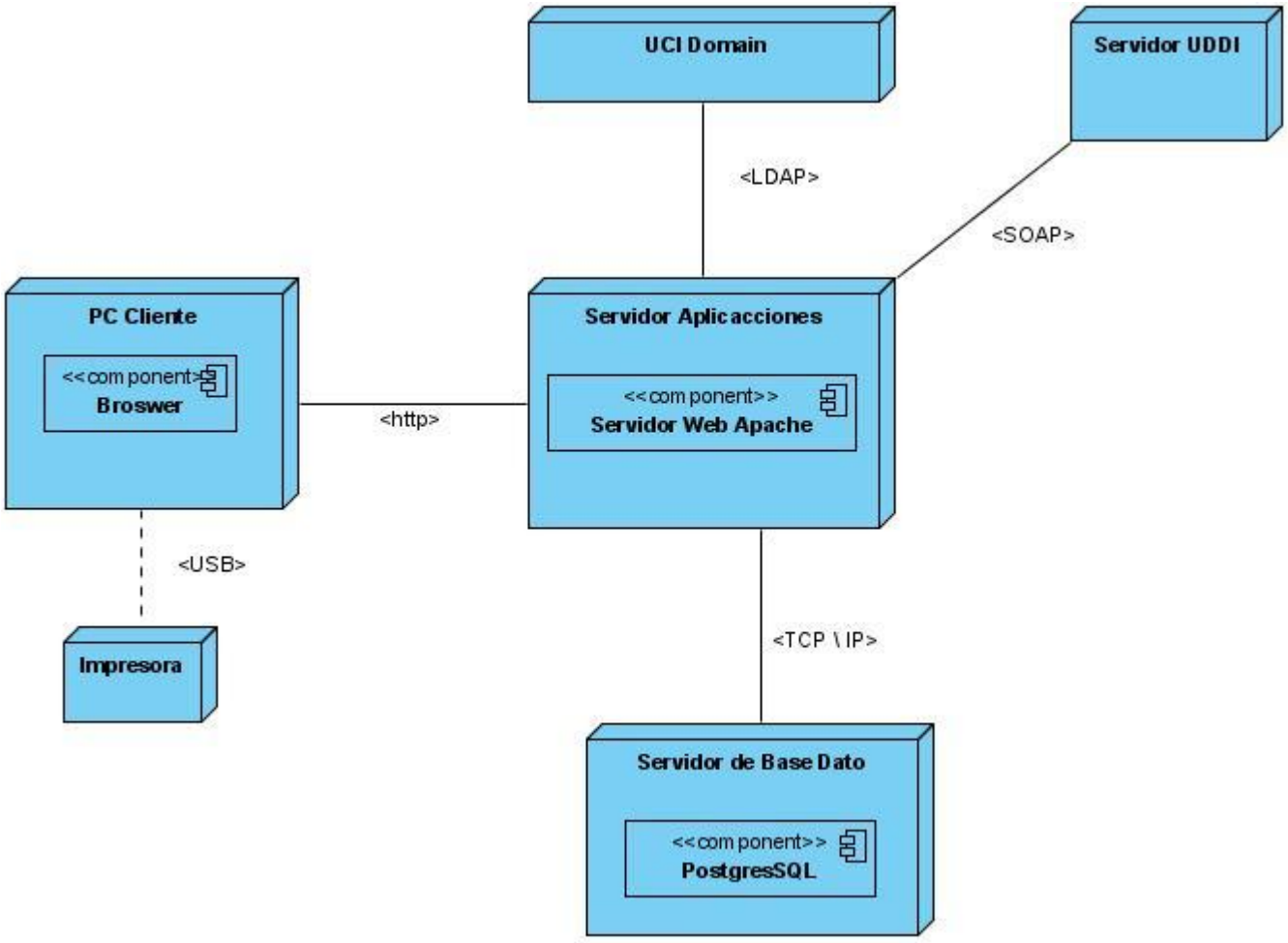

Figura 41 Diagrama de Despliegue para la aplicación

Conclusiones:

En el capítulo se abordaron todo los temas referentes a la alternativa propuesta, se definió como quedará el sistema que da razón de ser al desarrollo de este trabajo. En este capítulo se expusieron las características desde el más alto nivel de abstracción de la arquitectura. Para ello se dio a conocer la línea base de la arquitectura y se definieron las cuatro mas una vistas que define RUP para el desarrollo de una arquitectura eficiente, que logre un mayor entendimiento de los desarrolladores y clientes del producto de cómo será el sistema que se pretende desarrollar.

Capítulo #3 Evaluación de La Alternativa Propuesta

Introducción:

El objetivo inicial de este capítulo es evaluar la alternativa propuesta para ello existen un grupo de métodos como (SAAM) propuesto por Rick Pasman, Leng Bass, Gregory Abowd y Mike Webb quienes proponen tres perspectivas para el análisis [15].

El método de evaluación SAAM se enfoca en la enumeración de un conjunto de escenarios que representan los cambios probables a los que estará sometido el sistema en el futuro. Como entrada principal, es necesaria alguna forma de descripción de la arquitectura a ser evaluada. De acuerdo con Kazman et al. (2001), las salidas de la evaluación del método SAAM son las siguientes:

- Una proyección sobre la arquitectura de los escenarios que representan los cambios posibles ante los que puede estar expuesto el sistema.
- Entendimiento de la funcionalidad del sistema, e incluso una comparación de múltiples arquitecturas con respecto al nivel de funcionalidad que cada una soporta sin modificación [15].

Con la aplicación de este método, si el objetivo de la evaluación es una sola arquitectura, se obtienen los lugares en los que la misma puede fallar, en términos de los requerimientos de modificabilidad. Para el caso en el que se cuenta con varias arquitecturas candidatas, el método produce una escala relativa que permite observar qué opción satisface mejor los requerimientos de calidad con la menor cantidad de modificaciones.

Por otra parte el Método de Análisis para Desventaja de Arquitectura (ATAM) el cual está diseñado para producir como respuesta las metas comerciales tanto del sistema como de la arquitectura, y usar esas metas y la participación de los trabajadores para centrar la atención de los evaluadores en la porción de la arquitectura que es esencial para el cumplimiento de dichas metas [16].

Este método está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM, explicado anteriormente. El nombre del método ATAM surge del hecho de que revela la forma en que una arquitectura específica satisface ciertos atributos de calidad y provee una visión de cómo los atributos de calidad interactúan con otros; los tipos de acuerdos que se establecen entre ellos.

El método se concentra en la identificación de los estilos arquitectónicos o enfoques arquitectónicos utilizados. Kazman et al. (2001) proponen el término enfoque arquitectónico dado que no todos los arquitectos están familiarizados con el lenguaje de estilos arquitectónicos, aún haciendo uso indirecto de estos. De cualquier forma, estos elementos representan los medios empleados por la arquitectura para alcanzar los atributos de calidad, así como también permiten describir la forma en la que el sistema puede crecer, responder a cambios, e integrarse con otros sistemas [16].

3.1 Evaluando la alternativa propuesta

Para realizar la evaluación correspondiente se utilizarán los atributos de calidad establecidos en el estándar ISO 9126 para la medida de calidad de software.

3.1.1 Parámetros de calidad según la ISO 9126.

Funcionabilidad

- Educación: Capacidad del producto de software para proporcionar un conjunto de funciones para tareas específicas.
- Exactitud: Capacidad para proporcionar los resultados correctos.
- Interoperabilidad: Capacidad del producto para interactuar con más sistemas.
- Seguridad de acceso: Capacidad para proteger la información y los datos.

Fiabilidad

- Madurez: Capacidad del producto para evitar fallas.
- Tolerancia a fallos: Capacidad de mantener un nivel de prestaciones en caso de fallos.
- Capacidad de recuperación: Capacidad del software para restablecer las prestaciones y recuperar los datos dañados.

Usabilidad

- Capacidad para ser entendido: Capacidad que permite que el usuario pueda entender si el producto es el adecuado.
- Capacidad para ser aprendido: Capacidad que permite al usuario operar y controlar el software.
- Capacidad de atracción: Capacidad del producto de ser atractivo al usuario.

Eficiencia

- Comportamiento temporal: Capacidad del producto para proporcionar tiempos de respuestas bajo condiciones determinadas.
- Utilización de recursos: Capacidad del software para utilizar la cantidad y tipo de recurso más adecuado.

Mantenibilidad

- Capacidad para ser analizado: Capacidad del producto para diagnosticar deficiencias o fallos.
- Capacidad de ser cambiado: Capacidad de permitir hacerle modificaciones.
- Estabilidad: Capacidad del producto para evitar efectos inesperados.
- Capacidad para ser probado: Capacidad que debe permitir que el software modificado sea válido.

Portabilidad

- Adaptabilidad: Capacidad del producto para adaptarse a diferentes entornos.
- Instalabilidad: Capacidad del producto de ser instalado en un entorno determinado.
- Coexistencia: Capacidad que tiene el producto de poder coexistir con otros software en un entorno determinado.
- Capacidad para remplazar: Capacidad que tiene el producto para remplazar a otro software.

[17]

3.1.2 Como cumple la arquitectura con los atributos de calidad.

A continuación se explica como cumple la arquitectura con los atributos de calidad antes mencionados.

Funcionabilidad

La arquitectura propuesta cumple con este atributo de calidad ya que la misma centra su desarrollo en dos características esenciales.

- Se basa en un proceso que establece a los casos de uso como un elemento de vital importancia. Esto da la medida de cómo se concibe un sistema que responda realmente a las necesidades del cliente.

- Se basa en un proceso iterativo e incremental. Esto da la medida de cómo la arquitectura propone un prototipo funcional del sistema escogiendo las funcionalidades básicas o casos de uso críticos que son priorizados en las primeras iteraciones del ciclo de vida del proyecto.

Fiabilidad

La arquitectura tiene definidos componentes que no permiten la entrada de cadenas no válidas, y en caso de que el usuario se valga de otros recursos para lograrlo, se le enseña a través de recursos visuales, cual fue el error y no procesará la información hasta que esta sea entrada correctamente. Además de esta funcionalidad la aplicación es capaz de identificar el usuario con sus privilegios y solamente podrá acceder a las acciones que estén definidas para ese usuario. Esta acción se realiza obligando al usuario a que entre su usuario y contraseña para acceder al sistema verificando que este sea un usuario de la aplicación.

Usabilidad

La arquitectura de la aplicación está diseñada para que se le informe al usuario de acciones que sean irreversibles que puedan tentar contra la integridad de los datos o puedan ocasionar algún fallo en la aplicación un ejemplo de esto es cuando el usuario pretende eliminar una información determinada el sistema siempre muestra un mensaje al usuario para que este confirme si en verdad desea realizar esta operación. Otras de las opciones que garantiza la aplicación es que el usuario avanzado define los datos que se quieren gestionar del personal así como en que componentes desea que sea visualizada esta información.

Eficiencia

La arquitectura propuesta cumple con este parámetro ya que la aplicación destina los componentes esenciales en cinco nodos, uno contendrá todo lo necesario para la visualización de la aplicación, es decir navegador Web impresora para impresión de reportes entre otras, el segundo nodo contendrá la aplicación, el otro nodo la Base de Datos, el resto de los nodos restantes están destinado a los

WebService y el servidor de nombre de dominio de la universidad. De esta forma se garantiza que las tareas queden separadas y destinadas a cada nodo en específico y este utilice los recursos del sistema de cómputo en ejercer su función.

Portabilidad

La primera decisión importante con el objetivo de desarrollar una aplicación portable fue la selección de la tecnología de implementación. Para el desarrollo del sistema se utilizará el lenguaje de programación PHP con este lenguaje se pueden desarrollar aplicaciones para cualquier sistema operativo con la restricción que para interpretar este lenguaje se debe de contar con una tecnología de servidor capaz de interpretar este lenguaje. El hecho de la separación de la lógica de la aplicación de la interfaz garantiza que se pueda realizar cambios en esta parte y estas modificaciones no afecten a los usuarios y se realicen de manera abstracta para los clientes.

3.1.3 Comparando la arquitectura propuesta.

Para el desarrollo de este subepígrafe se seleccionaron dos propuestas arquitectónicas una la propuesta que se realizó para el desarrollo de un software para control de tráfico aéreo en Estados Unidos. El segundo caso es la propuesta arquitectónica para el software dedicado a la asistencia sanitaria en hospitales. Para esta comparación se darán como cumplen estas soluciones con los atributos de calidad.

Control de Tráfico Aéreo (ATC) de los Estados Unidos

Atributos de Calidad	Como Son Cumplidos
Disponibilidad	El programa se crea a partir de uno o más archivos de origen, que generalmente comprende una serie de subprogramas, algunos de los cuales se reunieron por separado en el paquete compilables. Está compuesto de varios de estos programas, muchos de los cuales operan en un modo cliente-servidor.
Desempeño	Esta aplicación debe utilizar para logra un alto desempeño multiprocesadores distribuidos.

Modificabilidad	Plantillas y mesa impulsada por la adaptación de datos; cuidadosa asignación de responsabilidad a cada módulo; estricto uso de interfaces específicas
Interoperabilidad	La potente funcionabilidad de la tecnología Cliente-Servido de hacer la división en la comunicación basada en mensajes garantiza este atributo.
Fiabilidad	Garantizando una separación de los datos de la aplicación así como una asignación de responsabilidades.

El Sistema Nightingale

Atributos de Calidad	Como Son Cumplidos
Rendimiento	En carga máxima, el sistema es capaz de completar 150 transacciones por segundo normalizado. (H, M)
Cofigurabilidad	La configuración del equipo hace que el cambio en 1 día hábil, no el código fuente tiene que cambiar. (H,L)
Seguridad	El sistema se resiste a la intrusión no autorizada. (H,M)
Mantenibilidad	Requiere un cambio en el informe de generación de metadatos. (M,L)

Conclusiones:

Al realizar la evaluación de dos arquitecturas de software diferentes y de países diferente, un país desarrollado (Estados Unidos) y uno subdesarrollado (México). Se llega a la conclusión que si las arquitecturas antes mencionadas cumpliendo con un número de parámetros de calidad son capases de satisfacer las necesidades para la cual fueron creadas y están acordes a las expectativas que se esperaban de ellas. La arquitectura propuesta para el desarrollo de la aplicación Web para la gestión de

los recursos humanos en los polos productivos de la Facultad 9 es la adecuada y logra el objetivo para el cual fue creada. Por tanto este software puede desarrollarse rigiéndose por esta primera versión arquitectónica ya que esta arquitectura cumple con un gran número de los atributos de calidad que expone la ISO 9126 para la medida de calidad de software como se muestra en este capítulo.

CONCLUSIONES

La propuesta arquitectónica que se presenta abarca dos documentos importantes: la definición de la línea base de la arquitectura y la descripción de la arquitectura, de constante refinamiento por el arquitecto encargado de la evolución del sistema. En el presente trabajo se realizó un estudio de las distintas metodologías para el desarrollo de software. Se definieron los estilos arquitectónicos, sus componentes, configuraciones y las restricciones impuestas por los requisitos del cliente. Ya entonces se documentó la línea base de la arquitectura donde se seleccionaron el grupo de herramientas con las que se desarrollaría el software. Seguido de esto se definieron y documentaron las vistas arquitectónicas. Al poder evaluar y comparar la alternativa propuesta con otras dos soluciones se llegó a la conclusión que quedó establecida y probada una primera versión de una arquitectura para el desarrollo del software de gestión de los recursos humanos en los polos productivos de la Facultad 9 de La Universidad de las Ciencias Informáticas que sea reusable, flexible, garantice la entrega del sistema en tiempo y que se cumplan con las restricciones impuestas por los clientes.

RECOMENDACIONES

1. El análisis de las dependencias entre los subsistemas de implementación para evitar que existan inconsistencias en alguno de ellos y que se vean afectados en cuanto a tiempo la entrega de cada una de las actividades de implementación.
2. Tener en cuenta la propuesta de arquitectura desarrollada en esta investigación, para futuros desarrollos dentro del área temática.
3. Valorar la posibilidad de hacer extensiva esta propuesta a otras áreas relacionadas con la gestión de los recursos humanos.

[1] (Pressman R. (2005). Ingeniería del Software. Un enfoque práctico. Parte 1. La Habana Cuba: Félix Varela.)

[3] (Paradigma visual para UML (Plataforma Java) (Visual Paradigm for UML [Java Platform]) 6.0 http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/)

[2](Rumbaugh, J. (1999). The Unified Modeling Language Reference Manual,Addison Wesley.)

[4](Giraldo y Zapata (2005).HERRAMIENTAS DE DESARROLLO DE INGENIERIA DE SW PARA LINUX, Monitoria de Ingesoft.)

[5] (Gutiérrez, E. (2007). www.embarcadero.com/products/erstudio/index.html.)

[6](González, D. (2008). Base de Datos PostgreSQL, SQL avanzado y PHP. http://www.usabilidadweb.com.ar/postgre.php)

[7] (Luis García. Sistema de control de versiones: SUBVERSION http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=548)

[8] (Reynoso, C. B. (2004). Introducción a la Arquitectura de Software. Consultado en 28 de noviembre, 2008 en http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/introarq.pdf)

[9](Booch, G., Rumbaugh, J. y Jacobson, I. (2000). El Proceso Unificado de Desarrollo del Software. Madrid: Pearson Educación S.A.)

[10] (http://www.netcraft.com/Survey/index-200204.Html).

[11] (Domínguez, C, D, A, “PROPUESTA DE LA ARQUITECTURA PARA EL SISTEMA INTEGRAL DEL CALCULO DE INDICES AL CONSUMIDOR”, 2007, Universidad de las Ciencias Informáticas: Ciudad de la Habana.)

[12] (Kicillof, C. R. N. (2004). "Estilos y Patrones en la Estrategia de Arquitectura de Microsoft Versión 1.0 ".)

[13] (MÖHRKE, C., Exploración del Entorno de Desarrollo Zend Studio DESARROLLO RÁPIDO. 2006.)

[14] (Stefan Küng, L.O., Simon Large, Un cliente de Subversion para Windows: Versión 1.4.0. 2006.)

[15] (Kazman, R., Blass, L., Abowd, G., Webb, M. (1994). SAAM A Method for Analysing the Properties of Software Architecture. Consultado en Febrero del 2009 en http://sunset.usc.edu/~neno/teaching/s99/SAAM.pdf.)

[16] (Blass, L., Clements, P. y Kazman, R. (2003) Software Architecture in Practice. :Addison Wesley)

[17] (Querétaro, marzo de 2006 ISO 9126-3: Métricas Internas de la Calidad del Producto de Software)

1. Reynoso, B. C.; et al. Introducción a la Arquitectura de Software publicado en el año 2004, UNIVERSIDAD DE BUENOS AIRES.

2. Reynoso, B. C.; Kicillof, N.; et al Estilos y Patrones en la Estrategia de Arquitectura de Microsoft Versión 1.0 publicado en el año 2004, UNIVERSIDAD DE BUENOS AIRES.

3. JACOBSON, Ivar; BOOCH, Grady, RUMBAUGH, James, “El Proceso Unificado de Desarrollo de Software”.2007. Addison Wesley. CAPITULO 6.

4. Racional Unified Process, Racional Suite 2007

5. Ivar Jacobson, G. B., J. Rumbaugh (2000). El proceso de desarrollo unificado de software. Madrid,
6. Pearson Educación. S.A.

7. Larman, C. (2004). UML y Patrones. La Habana, Félix Varela.

8. Pfleeger, S. L. (2002). Ingeniería de Software: Teoría y Práctica. Madrid, Prentice-Hall.

9. Pressman, R. (2001). Ingeniería del Software: Un enfoque práctico. Madrid, McGraw Hill.

10. Rolando Alfredo, S. C. (2002). EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA. Ciudad de la Habana.

11. Society, S. E. S. C. o. t. I. C. (2000). "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems."

12. Bass, L., Clements, P., y Kazman, R. (1998) Software Architecture in Practice. Addison Wesley.

13. Bastarrica, M. C. (2005) Atributos de Calidad y Arquitectura del Software. Volume, 4 Booch, G. (1999). The Unified Modeling Language User Guide, Addison Wesley.

14. IEEE-1471 (2000) Recommended Practice for. Architectural Description of Software-Intensive Systems.

15. Monroe, R. T., Kompanek, D., Melton, R. y Garlan, D. (1996). Stylized Architecture, Design Patterns, and Objects. Consultado en diciembre 2, 2008 en http://www.cs.cmu.edu/afs/cs.cmu.edu/project/able-10/OldFiles/ftp/ieee_arch_pattern_obj_submitted.pdf.

16. Casanovas, J. (2004). Usabilidad y arquitectura del software. Consultado en Enero 10, 2007 en http://www.alzado.org/articulo.php?id_art=355.

17. Fowler, M. (2004). ¿Ha muerto el diseño? Consultado en diciembre 20, 2008 en http://www.programacionextrema.org/articulos/designdead.es.html.