Universidad de las Ciencias Informáticas

Centro de Soluciones de Gestión

Título: Diseño e Implementación de la Arquitectura del Subsistema Capital Humano del Sistema Cedrux

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores: Yuliet Stivens Heredia
Marcos Bencomo García

Tutor: Ing. Rosendo Leonardo Hernández Claro

Ciudad de La Habana, junio 2009

"Año del 50 aniversario del triunfo de la Revolución"

El mérito verdadero es el que el hombre adquiere con su voluntad, con su esfuerzo, con su constancia.

AGRADECIMIENTOS

Ambos le agradecemos, sobre todo, a nuestro super tutor, por haber sido a la vez que eso: cotutor, tesista, compañero, amigo.

También le agradezco el apoyo a mi novia, a Yuliet y a mis amigos.

Marcos

Un sincero agradecimiento a todos aquellos familiares y a la gran cantidad de amigos que de tanta preocupación que mostraron me hicieron sentir que hacíamos la tesis juntos, eso me llenó de voluntad para lograrlo.

Uno especial por su paciencia y por su tiempo para Albertini, Rene, Yovanoti, Fidel, Carlos, Guillermo, Javier, Pura, Ariagna, Rubén, Susana, Dayana, Alain, Yania, Grettel, Alién, Omarito, Matos, Marcos y otro más para mi tutor y en general a todas las personas que a pesar de todo estuvieron siempre dispuestas a aclararme mis innumerables dudas.

También le agradezco mucho al tribunal por obligarme a redoblar mis esfuerzos. Hoy me enorgullezco de que no haya sido fácil de recorrer el camino.

Y a la Virgencita de la Caridad del Cobre y a Dios, porque la fe tan profunda en ellos me dio optimismo y serenidad en los momentos más duros.

Yuliet

DEDICATORIA

A Rosendo, por parte de ambos, porque su dedicación fue constante y su apoyo, incondicional y porque nos ayudó más de lo que nos merecíamos.

A un amigo que quise mucho y que me enseñó a mirar la vida de otro modo: Manuel Alejandro Buch.

A toda mi familia, porque imaginarla orgullosa de mí me impulsó y en especial a mis ejemplares padres, porque en realidad he llegado hasta aquí tratando de ser como ellos. A mi hermanito, porque el amor que me ha demostrado me ayudó a continuar. Y a mi abuela...

A Linita, Dariencitin, Lis, Yoyi, Keila, Dunia, Clenda, Lisandra Roman, Kenia porque siempre estuvieron ahí.

A mis amistades del teatro, del kenpo y del taek won do.

Y en general a todos aquellos que sintieron miedo por mí, a los que dudaron, a los que siempre confiaron en que podía lograrlo y a quienes me trataron como familia aún sin serlo.

Yuliet

RESUMEN

Todo buen software debe estar soportado por una arquitectura que esclarezca al equipo de desarrollo los objetivos que se persiguen con el mismo. Un buen diseño arquitectónico exige la realización de un estudio profundo de tendencias actuales a nivel mundial y la aplicación de estilos y patrones existentes. Tanto de lo anterior como del empleo de herramientas y tecnologías acorde con la solución esperada, depende la obtención de un software con un nivel adecuado de calidad.

El presente trabajo está dirigido a lograr *el diseño y la implementación de una arquitectura de sistema* robusta y entendible que guíe el proceso de desarrollo de software de la línea Capital Humano del Proyecto ERP Cuba. Con dicho propósito fue realizado un estudio bibliográfico de aspectos estrechamente relacionados con la Arquitectura de Software. Los resultados se basan en la reutilización de varios esquemas que apoyaron el desarrollo de la estructura definida. Además, para su realización fueron empleadas diversas herramientas y tecnologías modernas; las cuales contribuyeron considerablemente a que la arquitectura implementada cumpliera con los atributos de calidad necesarios para ser avalada como eficiente.

De lograrse el objetivo que se persigue con el presente trabajo investigativo, se podrá integrar el subsistema Capital Humano al Sistema Integral de Gestión Cedrux, para que de este modo brinde su funcionalidad al país. Un importante beneficio asociado a la presente solución es que al estar soportada por una arquitectura basada en componentes, ofrece la posibilidad de que todos sus elementos sean reutilizados en otros sistemas de gestión empresarial.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

INTRODUCCIÓN

La gestión del capital humano es un aspecto que cada día cobra mayor importancia en Cuba y en el resto del mundo. Esto se debe fundamentalmente a que repercute en el éxito de cualquier empresa, al influir de una forma directa en sus niveles de calidad de servicios y en los de productividad del trabajo. Los procesos de administración del mismo se realizan en el departamento de Capital Humano de las organizaciones y su gran significación exige el análisis de los problemas existentes en dicha área, así como el establecimiento de directrices enfocadas a su solución.

Actualmente Cuba presenta deficiencias en la gestión de Capital Humano de entidades presupuestadas y empresariales; lo cual afecta considerablemente la economía del país. Entre otras cosas, ocurren gastos excesivos de materiales; la calidad de los servicios no es la óptima, así como tampoco lo es la calidad de la gestión del Capital Humano; por otro lado no se tiene un control centralizado de la información de cada una de las empresas y no existe una interrelación entre las diferentes áreas del negocio.

Como parte del proceso de informatización empresarial cubano surgió la necesidad de desarrollar un Sistema de Gestión de Capital Humano que se integre a un Sistema Integral de Gestión. El mismo deberá cumplir con el modelo de software que se pretende obtener a fin de satisfacer las necesidades de la sociedad y mejorar el rendimiento económico del país.

Para obtener una solución informática que facilite el establecimiento tanto de políticas, como de objetivos que resuelvan las necesidades de gestión de dicha área en cualquier entidad, se necesita crear la base tecnológica, normar el proceso de desarrollo de software y construir la integración de cada componente definido con otros del subsistema y a su vez con otros sistemas asociados al Sistema Integral de Gestión Cedrux. Con el propósito de garantizar el correcto funcionamiento del producto se define la Arquitectura de Software, la cual en este caso está basada en la reutilización de componentes y la utilización de un marco de trabajo, definidos por un equipo de arquitectura central.

La Arquitectura de Software es una práctica joven, de apenas unos pocos años de trabajo constante, sin embargo ya se ha convertido en un factor de vital importancia para lograr que las aplicaciones informáticas tengan un alto nivel de calidad. Poseer una buena Arquitectura de Software es de suma importancia, ya que ésta es el meollo de

toda aplicación informática y determina cuáles serán los niveles de calidad asociados al mismo.

Una vez analizado lo anterior, se plantea como problema científico: *la necesidad de desarrollar la arquitectura de sistema de la línea Capital Humano, para crear la base tecnológica y garantizar estabilidad en el desarrollo y la obtención de un producto portable y eficiente que cause un impacto favorable en la sociedad.*

Teniendo en cuenta el problema planteado, se especifica como objeto de estudio *la arquitectura de software*. Y queda enmarcado como campo de acción *la arquitectura de software basada en componentes*.

Como objetivo general quedó determinado *diseñar la arquitectura de sistema de la línea Capital Humano y su implementación, para guiar el proceso de desarrollo de software e integrar sus componentes entre sí y a otros sistemas asociados al Cedrux.*

De acuerdo al propósito fundamental de este trabajo han sido concebidos como objetivos específicos:

- Realizar un estudio del estado del arte, en el cual se tenga en cuenta la evolución de la arquitectura de software en la actualidad.
- Aplicar una guía de marco de trabajo que optimice el proceso de desarrollo de software.
- Definir los componentes del sistema y la integración interna y externa.
- Evaluar la arquitectura implementada.

Para lograr los objetivos anteriormente propuestos se proponen un conjunto de tareas:

1. Describir los patrones arquitectónicos aplicados a la solución informática implementada.
2. Controlar el cumplimiento de las buenas prácticas de programación y de los estándares dictados por el centro.
3. Definir la arquitectura de sistema acorde a las herramientas y tecnologías propuestas por el centro.
4. Identificar los componentes del subsistema.
5. Identificar la integración interna y externa para cada capa de la arquitectura.
6. Priorizar los componentes definidos de acuerdo a la complejidad y nivel de dependencia.

7. Evaluar la arquitectura definida.

Idea a defender:

Realizándose el diseño e implementación de la arquitectura de sistema de la línea Capital Humano se construirían las bases tecnológicas del sistema en cuestión, lo que garantizaría la integración entre cada uno de los componentes identificados así como con otros subsistemas afines al Sistema Integral de Gestión Cedrux.

Resultados esperados:

- ✓ Arquitectura de sistema de la línea Capital Humano.
- ✓ Documentación del proceso investigativo vinculado a la solución.

En el primer capítulo que conforma el presente trabajo se realiza un estudio de varios conceptos vinculados a la Arquitectura de Software, se caracterizan los estilos arquitectónicos aplicados, se analizan algunos patrones de diseño y de asignación de responsabilidades y son referidas las tecnologías y herramientas utilizadas. El segundo se centra en la definición de componentes del Sistema de Gestión de Capital Humano. Para el cual se realizó la arquitectura del negocio y se priorizaron requisitos y componentes para luego integrar estos últimos. En este capítulo también se explican el empleo de patrones específicos en el diseño arquitectónico y otros artefactos. En el tercer y último capítulo se evalúa la solución y se muestran los resultados.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

El desarrollo arquitectónico de un software implica el análisis de todo el soporte tecnológico empleado, así como el dominio de todos los aspectos referentes al estilo y a los patrones a utilizar. En el presente capítulo se realiza un estudio profundo de los elementos predeterminados por el equipo central de arquitectura para el diseño e implementación de la arquitectura del subsistema Capital Humano. También serán analizados los elementos que fueron seleccionados internamente en la línea donde se llevo a cabo el proceso de desarrollo.

1.1 Arquitectura de Software

La Arquitectura del Software es una disciplina relativamente nueva, que se encuentra contenida en el ámbito del proceso de desarrollo de software. Representa una vista de abstracción del sistema como un todo y el trabajo vinculado a esta se manifiesta durante todo el ciclo de vida del proyecto.

1.1.1. Definiciones

A lo largo de la evolución de la Arquitectura de Software han sido conformados varios conceptos para definirla entre los que se destacan los citados a continuación:

"Estructura de los componentes de un programa o sistema, sus interrelaciones, y los principios y reglas que gobiernan su diseño y evolución en el tiempo." (1)

"La AS[1] es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones." (2)

"Estructura o estructuras de un sistema, lo que incluye sus componentes de software, las propiedades observables de dichos componentes y las relaciones entre ellos." (3)

[1] Sigla para denominar a la Arquitectura de Software.

Una interesante definición, procedente de RUP[2], plantea lo siguiente:

"La arquitectura de software es un conjunto de decisiones cruciales que restringen las consideraciones de diseño y programación y que tienen un alto impacto para ser revertidas." (4)

Y según la definición teórica de IEEE[3] en su clásica guía de recomendaciones IEEE 1471 (5):

"La arquitectura de software de un programa o sistema de computación es la estructura o estructuras del sistema, que comprende elementos del software, las propiedades externamente visibles de los elementos y las relaciones existentes entre ellos."

Resumiendo las anteriores ideas en pocas palabras, es posible decir que la Arquitectura de Software constituye el esquema de más alto nivel de la estructura de un sistema.

Básicamente consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción de un sistema de información. Tanto la selección como el diseño de la arquitectura de cualquier producto software, tienen como soporte las limitaciones derivadas de las tecnologías disponibles para la implementación del sistema, limitaciones que representan restricciones elementales para su creación; también constituyen una base sólida de apoyo: los objetivos establecidos en el negocio. (6)

El análisis arquitectónico no se limita a contemplar únicamente las funcionalidades exigidas para el producto, sino que toma en cuenta además cualidades tales como: la mantenibilidad, la confiabilidad, la escalabilidad, la portabilidad, la disponibilidad, el nivel de adaptabilidad y la interacción con otros sistemas de información. De modo que esta forma de modelar sistemas resulta de ensamblar un cierto número de elementos arquitectónicos de manera tal que queden satisfechos la mayoría de los requerimientos de desempeño del software y requisitos no funcionales como los mencionados anteriormente.

Esta sub-disciplina del proceso de desarrollo del software establece los fundamentos para que analistas, diseñadores, programadores y otros trabajadores que

[2] Rational Unifique Process. Abreviatura dada en inglés al Proceso Racional de Software

[3] Corresponde a las siglas de The Institute of Electrical and Electronics Engineers, o sea, el Instituto de Ingenieros Eléctricos y Electrónicos.

desempeñan un rol en un proyecto de software laboren en una línea común que permita alcanzar los objetivos del sistema informático, cubriendo todas las necesidades señaladas. La AS sirve de comunicación entre las personas involucradas en el desarrollo y facilita la realización de diferentes análisis que orienten la toma de decisiones durante el proceso de desarrollo de software.

1.1.2. Ventajas que aporta la Arquitectura de Software

Comunicación mutua: La AS representa un alto nivel de abstracción común que la mayoría de los participantes, si no todos, pueden usar como base para crear un entendimiento mutuo, formar un consenso y comunicarse entre sí. En sus mejores expresiones, la descripción arquitectónica expone las restricciones de alto nivel sobre el diseño del sistema, así como la justificación de decisiones arquitectónicas fundamentales.

Decisiones tempranas de diseño: La AS representa la encarnación de las decisiones de diseño más tempranas sobre un sistema. Es una especie de puerta de peaje: el desarrollo no puede proseguir hasta que los participantes involucrados aprueben su diseño.

Restricciones constructivas: Una descripción arquitectónica proporciona modelos parciales para el desarrollo e indica los componentes y las dependencias entre ellos. Por ejemplo, una vista en capas de una arquitectura documenta típicamente los límites de abstracción entre las partes. Esto a su vez identifica las principales interfaces y establece las formas en que unas partes pueden interactuar con otras.

Reutilización, o abstracción transferible de un sistema: La AS personifica un modelo relativamente pequeño e intelectualmente tratable, de la forma en que un sistema se estructura y sus componentes se entienden entre sí. Este modelo es transferible a través de sistemas; en particular, se puede aplicar a otros sistemas que exhiben requerimientos parecidos y puede promover reutilización en gran escala. El diseño arquitectónico soporta reutilización de grandes componentes o incluso de frameworks en el que se pueden integrar componentes.

Evolución: La AS puede exponer las dimensiones a lo largo de las cuales puede esperarse que evolucione un sistema. Haciendo explícitas estas “paredes” perdurables, quienes mantienen un sistema pueden comprender mejor las ramificaciones de los cambios y estimar con mayor precisión los costos de las modificaciones. Esas

delimitaciones ayudan también a establecer mecanismos de conexión que permiten manejar requerimientos cambiantes de interoperabilidad, prototipado y reutilización.

Análisis: Las descripciones arquitectónicas aportan nuevas oportunidades para el análisis, lo cual incluye verificaciones de consistencia del sistema, conformidad con las restricciones impuestas por un estilo, conformidad con atributos de calidad, análisis de dependencias y análisis específicos de dominio y negocios.

Administración: La experiencia demuestra que los proyectos exitosos consideran una arquitectura viable como un logro clave del proceso de desarrollo industrial. La evaluación crítica de una arquitectura conduce típicamente a una comprensión más clara de los requerimientos, las estrategias de implementación y los riegos potenciales.

Todo lo expresado en este subepigrafe fue basado en la referencia. (7)

1.1.3. Elementos estructurales de la Arquitectura de Software

Como la Arquitectura, a largos rasgos, es la organización fundamental de un sistema descrita en:

- Sus componentes.
- Relación entre ellos y con el ambiente.
- Principios que guían su diseño y evolución.

Resulta importante referir que el término componente está conceptualizado como la unidad básica reemplazable de un sistema, es una "*unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio*" (8)

En un componente son implementadas operaciones que a su vez son determinadas por un conjunto de *interfaces* cuya realización física es proporcionada por los componentes. Un componente puede tener múltiples interfaces, las cuales también se encargan de definir las operaciones que los componentes de los que ellas parten, utilizan de otros componentes durante su ejecución. Una interfaz es definida a través de un *conector,* el cual a la vez que vincula un componente con otro, encapsula los patrones de sincronización y coordinación entre ellos, define la interacción que media entre los mismos y describe las reglas que la gobiernan. (9)

1.2. Estilos Arquitectónicos

En 1996 Shaw y Garlan definen estilo arquitectónico como una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de cómo pueden ser combinadas. (10)

Para muchos estilos puede existir uno o más modelos semánticos que especifiquen cómo determinar las propiedades generales del sistema partiendo de las propiedades de sus partes. Los numerosos estilos en los que ha sido subdividida la disciplina tratada, permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos no funcionales, para luego sintetizar estructuras de soluciones y definir los patrones posibles de las aplicaciones. De modo que pocos estilos abstractos encapsulan una enorme variedad de configuraciones concretas. (11)

De la amplia gama de estilos de arquitectura que existen en la actualidad:

- Estilos de flujo de datos
- Estilos centrados en datos
- Estilos de llamada y retorno
- Estilos de código móvil
- Estilos heterogéneos

La presente investigación se concentrará en el funcionamiento de los que son aplicados directamente al desarrollo del subsistema en cuestión: el estilo de Arquitectura Basada en Componentes y el Modelo-Vista-Controlador. El uso de los mismos fue una decisión tomada por el equipo de arquitectura central del proyecto.

1.2.1 Estilos de Llamada y Retorno

En esta familia de estilos se enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala. (5) Miembros peculiares de la familia son los detallados en este epígrafe.

1.2.1.1 Modelo-Vista-Controlador

Se utiliza cuando es necesario modularizar la interfaz de usuario, las reglas de negocios y el control de eventos.

El patrón **MVC**[4] divide una aplicación interactiva en tres componentes en donde, de manera general, el modelo contiene la funcionalidad básica y los datos, la vista muestra la información al usuario, los controladores tramitan o manejan las aportaciones entradas por los usuarios y las vistas y controladores en conjunto constituyen la interfaz de usuario. El mecanismo propagación de cambio garantiza la coherencia entre la interfaz de usuario y el modelo.

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). Mantiene el conocimiento del sistema. No depende de ninguna vista y de ningún controlador.

Vista: Maneja la visualización de la información.

Controlador: Analiza los mensajes de eventos que recibe el sistema y modifica u obtiene datos del modelo en respuesta a las peticiones del usuario. Evita poner código indebido en la capa impropia, por ejemplo instrucciones de base de datos (modelo) en interfaz de usuario (vista).

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual. En aplicaciones Web la separación entre la vista (el browser) y el controlador (los componentes del lado del servidor que manejan los requerimientos de HTTP) está taxativamente muy bien definida. (12)

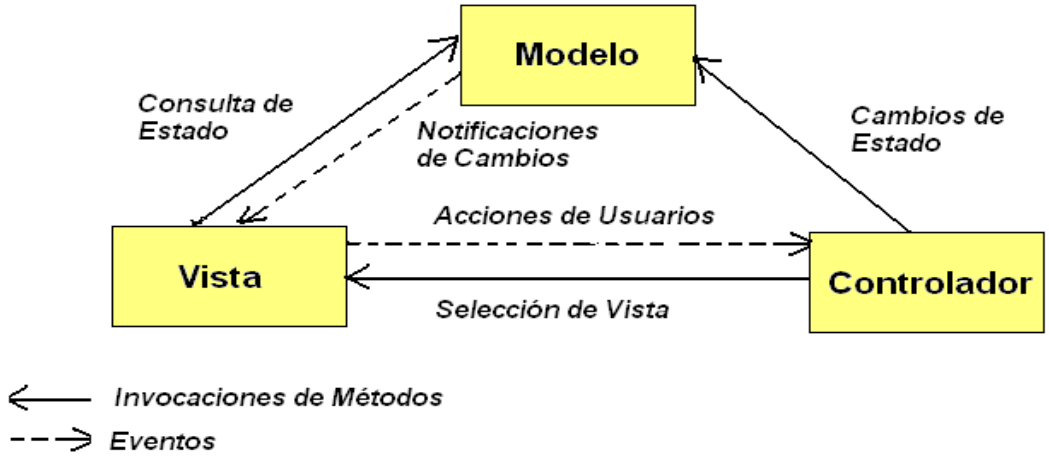


Figura 1 Estilo Modelo Vista Controlador.

[4] Sigla que denomina el estilo o patrón: Modelo-Vista-Controlador.

Ventajas

- *Adaptación al cambio*. Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos. Como el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.
- *Soporte de vistas múltiples*. Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes.

Desventajas

- *Costo de actualizaciones frecuentes*. Desacoplar el modelo de la vista no significa que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas. Si el modelo experimenta cambios frecuentes, por ejemplo, se podrían desbordar las vistas con una lluvia de requerimientos de actualización.
- *Complejidad.* El patrón introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución. También se profundiza la orientación a eventos del código de la interfaz de usuario, que puede llegar a ser difícil de depurar. (5)

1.2.1.2 Arquitectura Basada en Componentes

La Arquitectura de Software se define como la representación de alto nivel de la estructura de un sistema o aplicación, que describe, entre otros aspectos, las partes que la integran; en general, dicha representación se va a realizar en términos de una colección de componentes y de las interacciones que tienen lugar entre ellos. De esta forma aparecen las arquitecturas basadas en componentes y conectores (ambos aparecen explicados en 1.1.3). Este tipo de arquitectura es completamente modular y favorece la reutilización de todos sus elementos, incluyendo los que definen las distintas relaciones entre ellos.

Una arquitectura software viene determinada por las diferentes instancias de cada tipo de componentes y conectores que la componen, y por una serie de enlaces

específicos que definen la unión de todas ellas y forman una estructura. A esta estructura se le da el nombre de configuración y suele considerarse insertada en una jerarquía, pues toda entidad software, independientemente de su granularidad, dispone de una estructura que puede ser descrita mediante una arquitectura software.

El marco arquitectónico estándar para la tecnología de componentes está constituido por los cinco puntos de vista de RM-ODP (Empresa, Información, Computación, Ingeniería y Tecnología).

Las tecnologías de componentes del período de inmadurez se consideraban afectadas por problemas de incompatibilidad de versiones e inestabilidad que ya han sido ampliamente superados en la industria. (5)

Ventajas

- *Reutilización del software*. Nos lleva a alcanzar un mayor nivel de reutilización de software.
- *Simplifica las pruebas*. Permite que las pruebas sean ejecutadas a cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- *Simplifica el mantenimiento del sistema*. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- *Mayor calidad*. Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo. (9)

1.3 Patrones a usar

Un patrón es la descripción de un conjunto de interfaces, esquemas de clases y relaciones que permite hacer determinadas tareas de manera genérica. Muchos patrones son más útiles por la reutilización de la idea que por la reutilización de código (5).

1.3.1 Patrones de Distribución de Responsabilidades

La aplicación de estos patrones al modelar la arquitectura representa una distribución efectiva de la carga de trabajo entre las clases. El propósito de los mismos

de forma general es originar componentes robustos, entendibles y fáciles de mantener y reutilizar, lo cual explica que su adecuada utilización sea la clave para un exitoso diseño.

Controlador

Se utiliza la misma clase de control para manejar todos los eventos del sistema pertenecientes a un mismo componente.

Explicación: Los subsistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario operado por una persona. Otros medios de entrada son los mensajes externos o las señales procedentes de sensores como sucede en los sistemas de control de procesos. Incluso si no se recurre a un diseño orientado a objetos, es recomendable elegir los controladores que manejen esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. La misma clase controlador debería utilizarse con todos los eventos sistémicos de un proceso.

Beneficios: Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz. Al delegar a un controlador la responsabilidad de la operación de un sistema entre las clases del dominio favorece la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras.

Creador

Explicación: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

Beneficios: Brinda un soporte a un bajo acoplamiento –patrón que será descrito más adelante– lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

Experto

Explicación: Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Da origen a diseños donde el objeto de software realiza las operaciones que normalmente se aplican a lo que representan realmente, por lo que ofrece una analogía con el mundo real.

Beneficios: Con la utilización de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida, lo que favorece definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener.

Alta Cohesión

Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Explicación: El patrón Alta Cohesión es la meta principal que ha de tenerse en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. Además son difíciles de entender, de reutilizar y de mantener. Son clases al las cuales constantemente las afectan los cambios.

- Muy baja cohesión: Una clase es la única responsable de muchas cosas en áreas funcionales muy heterogéneas.
- Baja cohesión: Una clase tiene responsabilidad exclusiva de una tarea compleja dentro de un área funcional.
- Alta cohesión: Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para compartir el esfuerzo y llevar a cabo las tareas.
- Cohesión moderada: Una clase tiene responsabilidades exclusivas en unas cuantas áreas que están relacionadas lógicamente con el concepto de clase, pero no entre ellas.

Beneficios: Con el uso de este patrón mejoran la claridad y la facilidad con que se entiende el diseño. Se simplifican el mantenimiento y las mejoras en funcionalidad. A menudo se genera un bajo acoplamiento. La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico.

Bajo Acoplamiento

Explicación: El bajo acoplamiento es un principio que se debe tener siempre en cuenta durante las decisiones de diseño. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Este patrón estimula asignar una responsabilidad de modo que al hacerlo no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecienten la oportunidad de una mayor productividad. No puede considerarse en forma independiente de otros patrones como Experto o Alta Cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades.

Beneficios: Con el uso de este patrón los componentes no se afectan por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar.

Todo lo explicado anteriormente fue basado en ideas tomadas de las referencias: (13) (14)

1.3.2 Patrones de Diseño

En talleres efectuados con el equipo central de arquitectura del proyecto fueron analizados numerosos patrones que además de encontrarse entre los más comunes, quedaron definidos como los más posibles a ser aplicados en el diseño arquitectónico del subsistema de acuerdo a características generales del marco de trabajo y a otros aspectos valorados.

1.3.2.1 Patrones de Diseño Creacionales

Solitario

Propósito: Garantizar que una clase sólo tenga una instancia única y proporcione un punto de acceso global a la misma.

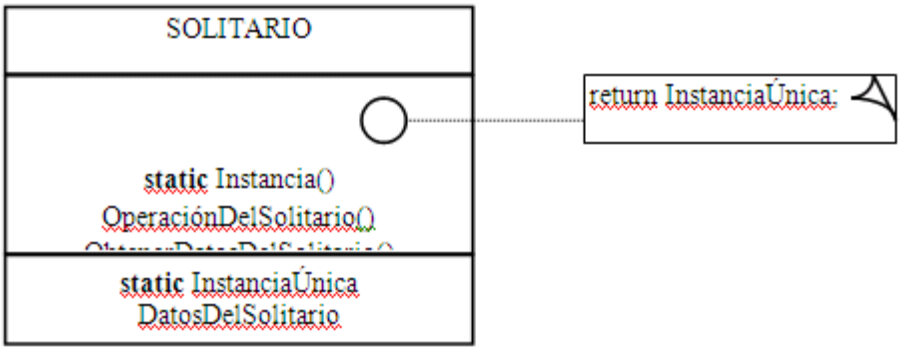


Figura 2 Estructura del patrón arquitectónico "Solitario".

Ventajas:

- El acceso a la "InstanciaÚnica" está más controlado.
- Se reduce el espacio de nombres (frente al uso de variables globales).
- *Permite refinamientos en las operaciones y en la representación, mediante la* especialización por herencia de "Solitario".
- Es fácilmente modificable para permitir más de una instancia y, en general, para controlar el número de las mismas (incluso si es variable).

1.3.2.2 Patrones Estructurales

Composicíón

Propósito: Permitir a los clientes tratar uniformemente a los objetos simples y compuestos de una estructura jerárquica recursiva.

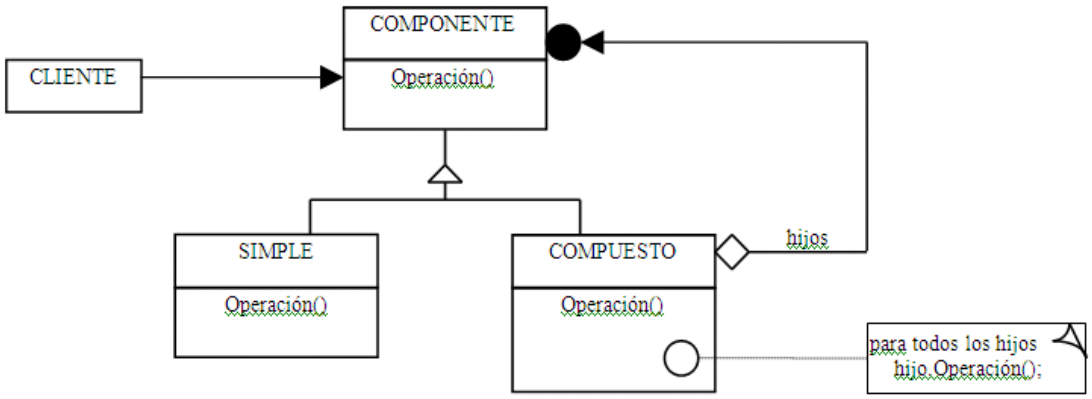


Figura 3 Estructura del patrón arquitectónico "Composición".

Ventajas:

- Simplifica notablemente al cliente, al no tener éste que distinguir entre objetos simples y compuestos.
- Favorece la extensibilidad, ya que es muy fácil añadir nuevos tipos de componentes, tanto simples como compuestos.

No obstante a las facilidades que aporta este patrón puede hacer el diseño peligrosamente genérico y esto dificulta la imposición de restricciones sobre ciertos componentes de la jerarquía (por ejemplo, las comprobaciones de tipo no se pueden hacer con garantía más que en tiempo de ejecución). Esta es la otra cara de la facilidad para la extensibilidad.

Fachada

Propósito: Proporcionar una interfaz unificada de alto nivel que, mediante la representación de todo un subsistema, facilite su uso. La "fachada" satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas.

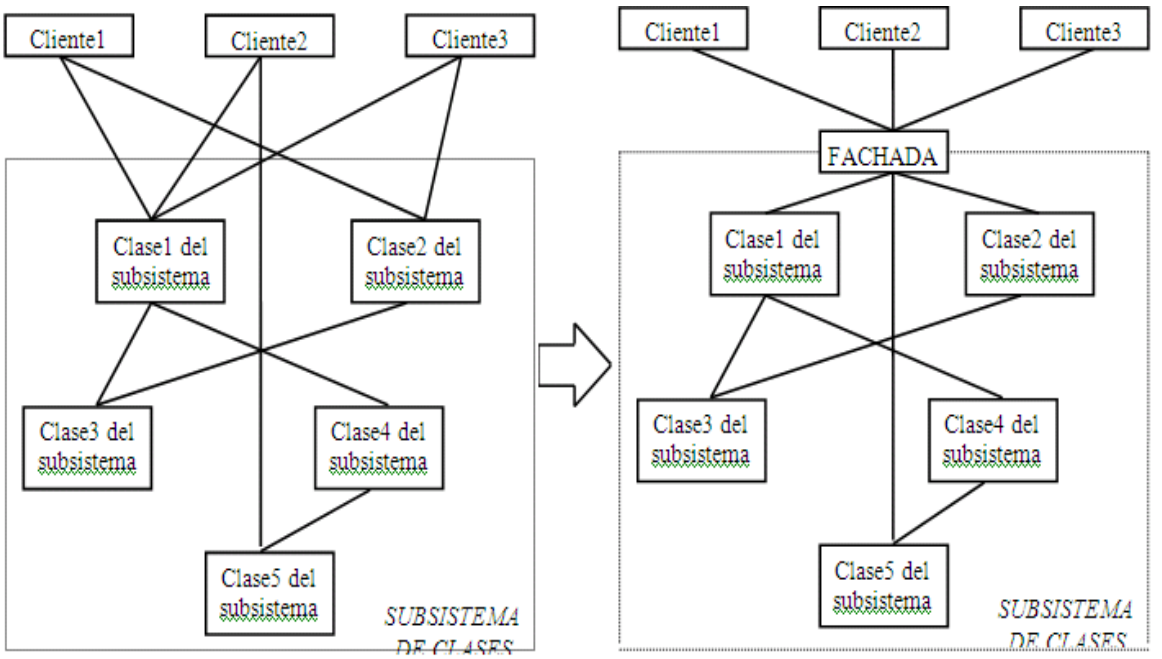


Figura 4 Estructura del patrón arquitectónico "Fachada".

Ventajas:

- Al separar al cliente de los componentes del subsistema, se reduce el número de objetos con los que el cliente trata, esto facilita el uso del subsistema.
- Se promueve un acoplamiento débil entre el subsistema y sus clientes, al ser eliminadas o reducidas las dependencias.
- No existen obstáculos para que las aplicaciones usen las clases del subsistema que necesiten. De esta forma se puede elegir entre facilidad de uso y generalidad.

1.3.2.3 Patrones de Comportamiento

Estado

Propósito: Permitir a un objeto modificar su comportamiento a medida que su estado interno va cambiando, dando así la impresión de que el objeto "cambia de clase".

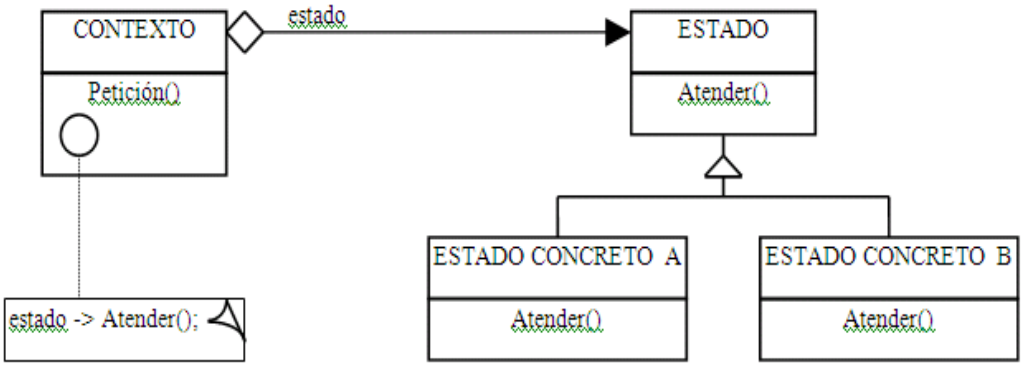


Figura 5 Estructura del patrón arquitectónico "Estado".

Ventajas:

- Dado que el comportamiento específico de cada estado está auto contenido en cada estado concreto, existe una gran flexibilidad para añadir nuevos estados y transiciones, mediante la definición de nuevas subclases.
- A pesar de que la distribución del comportamiento ante diferentes situaciones entre diferentes objetos incrementa el número de clases y reduce la compacidad, es la mejor solución cuando hay muchos estados, pues evita las definiciones multicondicionales y grandes.

- Las transiciones separadas entre estados se hacen más explícitas que si todo estuviera concentrado en una sola clase (esto hace más inteligible el diseño) e impide además los estados internos inconsistentes (desde el punto de vista del contexto, los cambios de estado son atómicos).
- Si los objetos que representan los estados no tienen variables de instancia (esto es, el estado está totalmente definido en su propio tipo) entonces diferentes contextos pueden compartir estados.

Mediador

Propósito: El diseño Orientado a Objeto enfatiza la distribución de responsabilidades entre objetos. Tal distribución puede devenir en una estructura con muchas conexiones entre objetos (en el peor caso, todos con todos). Aunque la distribución favorece la reutilización, la proliferación de relaciones entre objetos puede ser un grave obstáculo. Por ello, este patrón define un objeto que encapsula la forma en que interactúan un grupo de objetos y promueve de este modo un acoplamiento débil; evita las referencias explícitas entre los objetos y permite, por tanto, que su interacción se modifique de forma independiente.

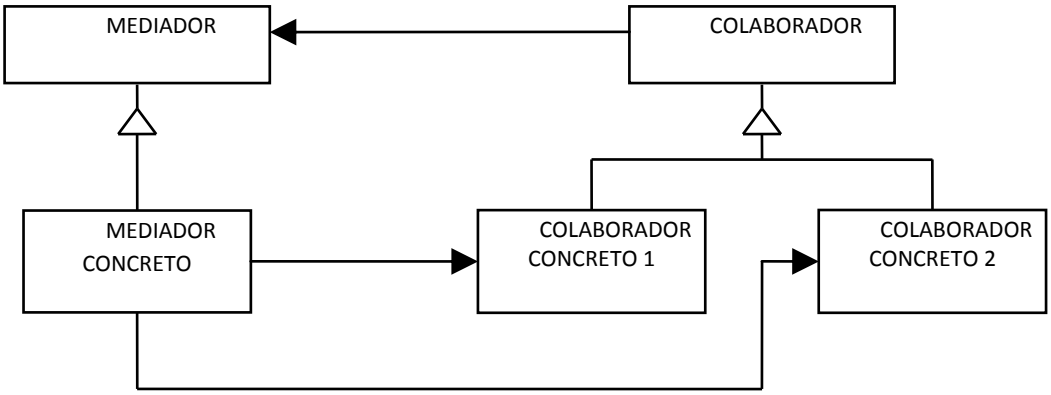


Figura 6 Estructura del patrón arquitectónico “Mediator”.

Ventajas:

- Se limita el uso de la herencia, al concentrar en el “Mediador” un comportamiento que, de otro modo, estaría distribuido entre diversos objetos. Modificar este comportamiento puede requerir especializar el “Mediador”, pero ya no afectará a los objetos que interactúan.
- Al reducir el acoplamiento entre los objetos que interactúan es más fácil variar o reutilizar dichos objetos y el “Mediador”, independientemente.

- Se simplifican los protocolos de comunicación entre los objetos, al sustituir las relaciones “muchos-a-muchos” por relaciones “uno-a-muchos”, que son más sencillas de entender, mantener y extender.
- Al hacer de la mediación un concepto independiente, encapsulado en un objeto aparte, se favorece la concentración de la atención en la interacción entre objetos, al margen del comportamiento individual de cada uno.

Independientemente de lo anterior es preciso destacar que al crear un objeto intermediario, disminuye la complejidad en la interacción; pero proporcionalmente aumenta la complejidad en este objeto, que se puede convertir en el elemento más complejo del conjunto.

Estrategia

Propósito: Definir una familia de algoritmos encapsulando por separado cada uno de ellos y haciéndolos, por tanto, intercambiables. Esto permite a los algoritmos variar con independencia de los clientes que los usan.

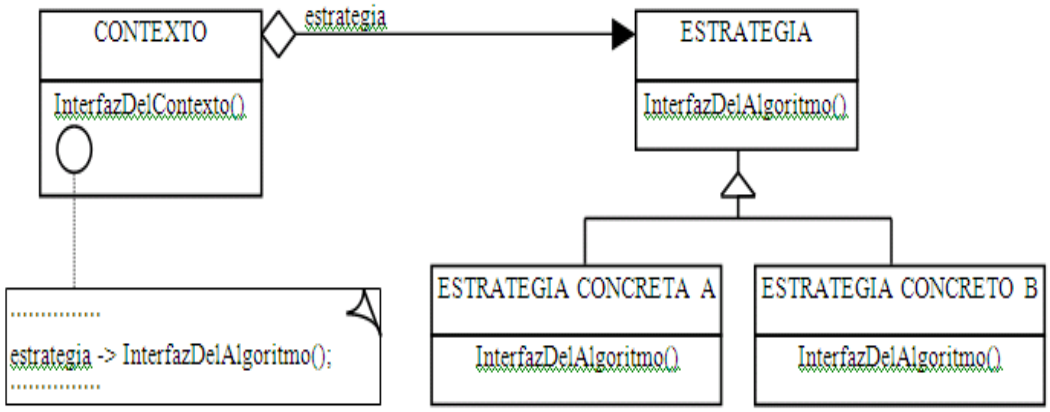


Figura 7 Estructura del patrón arquitectónico “Estrategia”.

Ventajas:

- Las familias jerárquicas de estrategias definen conjuntos de pasos y comportamientos que enfatizan la reutilización. La herencia puede ayudar a sacar factor común a la funcionalidad de los algoritmos.
- El encapsulamiento de algoritmos en clases separadas ofrece una ventajosa alternativa a la especialización por herencia del contexto para obtener un

comportamiento diferente, que promueve la independencia, la facilidad de entender el diseño y la posibilidad de futuras extensiones.

- Se eliminan las costosas definiciones de comportamientos multicondicionales.
- Se posibilita ofrecer diferentes implementaciones del mismo comportamiento, en función de restricciones como el espacio en memoria o el tiempo de respuesta.

Mas si bien proporciona todos estos beneficios también trae como consecuencia que los clientes deben tener un cierto conocimiento de cada estrategia, para entonces poder elegir en cada situación cual es la más apropiada. Además puede producirse una gran explosión en el número de objetos del sistema. Dado que todas las estrategias comparten una interfaz común, si las diferencias entre ellas es grande, es probable que mucha de la información que se les pasa no sea de utilidad más que a las más complejas. Todo lo anterior se puede aliviar si las estrategias se implementan como objetos sin estado que los contextos pueden compartir.

Cadena de Responsabilidades

Propósito: Proporcionar a más de un objeto la capacidad de atender una petición, para así evitar el acoplamiento con el que objeto que hace la petición. Se forma con estos objetos una cadena, en la cual cada objeto o satisface la petición o la pasa al siguiente.

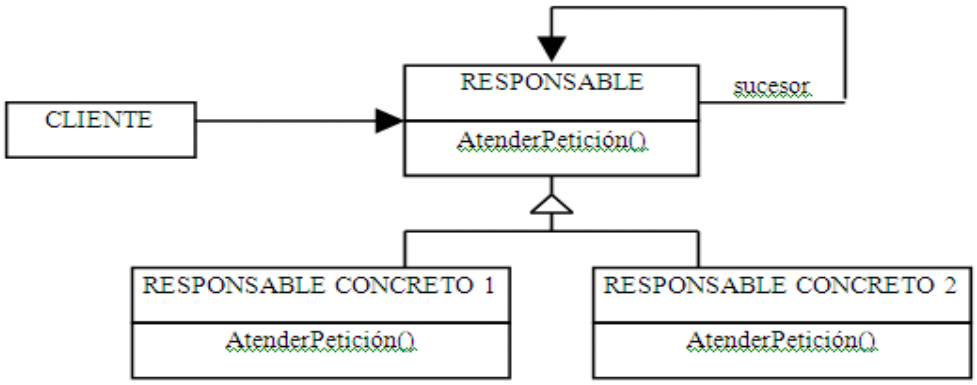


Figura 8 Estructura del patrón arquitectónico “Cadena de Responsabilidades”.

Ventajas:

- Se reduce el acoplamiento, puesto que se libera al objeto que hace la petición de conocer quien se la atiende. No sólo eso, sino que, además, los miembros de la cadena no conocen la estructura entera, sino sólo su sucesor en la misma.
- Mayor flexibilidad, puesto que se puede añadir o modificar la capacidad de atender una petición, simplemente haciendo cambios en la cadena de responsabilidades, dinámicamente (en tiempo de ejecución).

Su inconveniencia es que no queda garantizada la recepción y satisfacción de una petición, pues puede haber errores en la configuración de la cadena o también puede ocurrir que ningún elemento de la cadena sepa atender la petición.

(15) (14)

1.4 Tecnologías para el desarrollo de Sistemas Web

Las aplicaciones Web se han convertido en pocos años en complejos sistemas con interfaces de usuario cada vez más parecidas a las aplicaciones de escritorio. Sobre estos han establecido requisitos estrictos de accesibilidad y respuesta. De este modo han dado servicios a procesos de negocio de magnitud considerable.

Lo anterior ha exigido reflexiones sobre la mejor arquitectura y las técnicas de diseño más adecuadas. En los últimos años, la rápida expansión de Internet y del uso de intranets corporativas ha supuesto una transformación en las necesidades de información de las organizaciones. En particular esto afecta a la necesidad de que la información sea: accesible desde cualquier lugar dentro de la organización e incluso desde el exterior y compartida entre todas las partes interesadas, de manera que todas tengan acceso a la información completa (o a aquella parte que les corresponda según su función) en cada momento.

Estas necesidades han provocado un movimiento creciente de cambio de las aplicaciones tradicionales de escritorio hacia las aplicaciones Web, que por su idiosincrasia, cumplen a la perfección con las necesidades mencionadas anteriormente. Por tanto, los sitios Web tradicionales que se limitaban a mostrar información se han convertido en aplicaciones capaces de una interacción más o menos sofisticada con el usuario. Inevitablemente, esto ha provocado un aumento progresivo de la complejidad de estos sistemas y, por ende, la necesidad de buscar opciones de diseño nuevas, que permitan dar con la arquitectura óptima que facilite la construcción de los mismos. . (16)

1.4.1 Tecnologías del lado del cliente

XML

Son las siglas correspondientes al Extensible Markup Language, un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Representa una manera de definir lenguajes para diferentes necesidades. Está diseñado especialmente para los documentos de la Web, pero no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas y se puede usar en bases de datos, editores de texto, hojas de cálculo y en otros tipos de sistemas informáticos. Su carácter extensible permite que después de diseñado y puesto en producción, se le puedan adicionar nuevas etiquetas, de modo que no es una sintaxis, sino varias.

XML posibilita que los diseñadores creen sus propias etiquetas, lo cual hace posible la definición, transmisión, validación e interpretación de datos entre aplicaciones y entre organizaciones. Además, permite al programador dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, ya que algunas tareas tediosas como la validación de estos y el recorrido de las estructuras corren a cargo del lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse por ello.

Es una tecnología rodeada de otras que la complementan, la hacen mucho más grande y le aportan mayores posibilidades. Junto a todas las tecnologías relacionadas representa una manera distinta y más avanzada de hacer las cosas. Su principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. Así pues, el XML juega un papel primordial en el mundo actual, que tiende a la globalización y la compatibilidad entre los sistemas, ya que es una tecnología que permite compartir la información de una manera segura, fiable y fácil. (17)

(18)

JavaScript

JavaScript es un lenguaje de programación interpretado, por lo que no requiere compilación. Es utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al igual que Java, JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de herencia. En este caso las

nuevas clases son generadas a través de la clonación de las clases base (prototipos) y con la extención de su funcionalidad, lo que manifiesta que se sigue el paradigma de programación basada en prototipos. (19)

Javascript puede incluirse en cualquier documento HTML[5], o en todo aquel que se traduzca en HTML en el navegador del cliente; ya sea PHP, ASP[6], JSP[7], SVG[8] u otro. Incluir código directamente en una estructura HTML es una práctica invasiva, y no recomendada. El método correcto que define la W3C es incluir JavaScript como un archivo externo, tanto por cuestiones de accesibilidad, como practicidad y velocidad en la navegación. (20)

(21)

AJAX

AJAX, acrónimo en inglés de Asynchronous JavaScript And XML («JavaScript y XML asíncronos»). Es una técnica de desarrollo Web para crear aplicaciones interactivas mediante la combinación de tres tecnologías ya existentes:

HTML (o XHTML) y Hojas de Estilo en Cascada (CSS) para presentar la información.

Document Object Model (DOM) y JavaScript, para interactuar dinámicamente con los datos.

XML y XSLT[9], para intercambiar y manipular datos de manera no sincronizada con un servidor Web (aunque las aplicaciones AJAX pueden usar otro tipo de tecnologías, incluyendo texto llano, para realizar esta labor).

Este término en sí no constituye una tecnología, es un que engloba a un grupo de éstas que trabajan conjuntamente. Todos los navegadores Web usados por las aplicaciones AJAX soportan las tres tecnologías mencionadas anteriormente. Entre estos se incluyen: Mozilla Firefox, Internet Explorer, Safari y Opera.

Ventajas:

1. Utiliza tecnologías ya existentes.
2. Soportada por la mayoría de los navegadores modernos.

[5] Siglas en inglés Hyper Text Markup Language (Lenguaje de Marca de Hipertextos).

[6] Active Server Pages

[7] (Java Server Page) Página de Servidor Java

[8] Scalable Vector Graphics

[9] Extensible Stylesheet Language, Lenguaje extendido de hojas de estilo

3. Interactividad. El usuario no tiene que esperar hasta que lleguen los datos del servidor.
4. Portabilidad (no requiere plug-in).
5. Mayor velocidad, debido a que no hay que retornar toda la página nuevamente.
6. La página se asemeja a una aplicación de escritorio.

Desventajas:

1. Se pierde el concepto de volver a la página anterior.
2. Si se guarda en favoritos no necesariamente al visitar de nuevo el sitio, se sitúe en la ubicación grabarla.
3. La existencia de páginas con AJAX y otras sin esta tecnología hace que el usuario se desoriente.
4. Problemas con navegadores antiguos que no implementan esta tecnología.
5. No funciona si el usuario tiene desactivado el Java Script en su navegador.
6. Requiere programadores que conozcan todas las tecnologías que intervienen en AJAX.
7. En dependencia de la carga del servidor, es posible experimentar tiempos tardíos de respuesta que desconciertan al visitante.

(22) (23)

1.4.2 Tecnologías del lado del servidor

PHP 5.2

Hypertext Preprocessor, más conocido como PHP, es un lenguaje para el desarrollo de páginas Web dinámicas del lado del servidor. Al ser script no se compila para conseguir códigos máquina si no que existe un intérprete que lee el código y se encarga de ejecutar las instrucciones que contiene éste. Sus fragmentos de código se intercalan fácilmente en páginas HTML. Lo anterior unido a que es de Open Source (código abierto), lo ha vuelto muy popular y extendido en la Web.

PHP ofrece un extenso conjunto de funciones para la explotación de bases de datos sin complicaciones. Es capaz de realizar determinadas acciones de una forma fácil y eficaz, en la cual no es necesario generar programas creados en un lenguaje distinto al HTML. La versión 5.2 salió al mercado desde el 2 de noviembre del 2006 y además de manejar excepciones, tiene habilitado el filtro de extensiones por defecto. (24)

Apache Web Server 2.0

Apache es el servidor Web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa, cuyas particularidades constituyen ventajas.

Características:

- Corre en numerosos Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierto. Esto último es de mucha importancia, pues le da gran transparencia al software de manera que si se quiere ver que es lo que se está instalando como servidor, es posible saberlo sin ningún secreto.
- Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de logs. Ofrece la oportunidad de crear ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.

(25)

WAMP 5

Es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

- Windows, como sistema operativo;
- Apache, como servidor Web;
- MySQL, como gestor de bases de datos;
- PHP (generalmente), Perl, o Python, como lenguajes de programación.

Estos programas no fueron diseñados para trabajar conjuntamente entre ellos, pero la compatibilidad de sus componentes ha hecho efectiva la combinación, así como la ha hecho famosa su bajo coste. Este conjunto de aplicaciones es una alternativa viable a los paquetes comerciales existentes y tiene una fácil instalación, ya que se puede obtener en un instalador único que dejará Apache, MySQL y PHP funcionando en un sistema operativo Linux sin tener que configurar nada.

WAMP es una forma de mini-servidor que puede ejecutarse en casi cualquier sistema operativo Windows. El uso de un WAMP permite servir páginas HTML a

Internet, además de poder gestionar datos en ellas, al mismo tiempo un WAMP, proporciona lenguajes de programación para desarrollar aplicaciones Web.

(26)

1.4.3 Evolución de la tecnología

Framework

El concepto framework se emplea en muchos ámbitos del desarrollo de sistemas de software, no solo en el ámbito de aplicaciones Web, más en general, el término hace referencia a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que es posible añadirle las últimas piezas para construir una aplicación concreta.

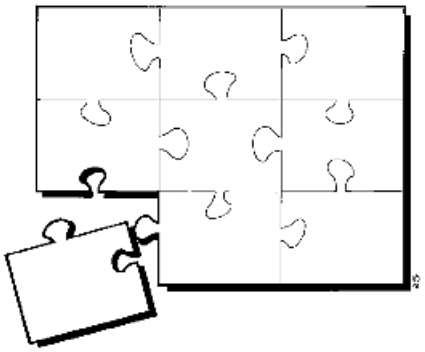


Figura 9 Framework.

Son diseñados con el intento de facilitar el desarrollo de software, al permitir a los diseñadores y programadores pasar más tiempo en la identificación de requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

Características:

A continuación son enunciadas una serie de particularidades posibles a ser encontradas en prácticamente todos los frameworks existentes:

Abstracción de URLs y sesiones

No es necesario manipular directamente las URLs ni las sesiones, el framework ya se encarga de hacerlo.

Acceso a datos

Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos, en BBDD, XML y otros.

Controladores.

La mayoría de frameworks implementa una serie de controladores para gestionar eventos, como una introducción de datos mediante un formulario o el acceso a una página.

Autentificación y control de acceso

Incluyen mecanismos para la identificación de usuarios mediante login y password, lo que permite restringir el acceso de los usuarios a las diferentes páginas, en correspondencia con el rol definido para los mismos.

Internacionalización

Separación entre diseño y contenido.

(27)

1.4.3.1 Frameworks aplicados mediante el marco de trabajo

EXT

Es un framework para JavaScript utilizado en el desarrollo de aplicaciones Web con AJAX. Tiene una gran librería que permite configurar las interfaces Web de manera semejante a aplicaciones de escritorio.

Tiene incluidos la mayoría de los controles de los formularios Web incluyendo Grids para mostrar datos y elementos semejantes a la programación de escritorio como los formularios, paneles, barras de herramienta, menús y varios otros. Dentro de su librería de componentes incluye componentes para el manejo de datos, lectura de XML, lectura de datos JSON[10] e implementaciones basadas en AJAX.

(28)

[10] **JSON:** acrónimo de "**J**ava**S**cript **O**bject **N**otation", es un formato ligero para el intercambio de datos, es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

Zend Framework

Se trata de un framework para desarrollo de aplicaciones y servicios con PHP, que brinda soluciones para construir sitios Web modernos, robustos y seguros. Es Open Source y trabaja con PHP5. Entre sus principales características figuran las siguientes:

- Trabaja con MVC (Model View Controller)
- Cuenta con módulos para manejar archivos PDF y Web Services (Amazon, Flickr, Yahoo) entre otros.
- El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Una solución para el acceso a base de datos que balancea el ORM con eficiencia y simplicidad.
- Completa documentación y tests de alta calidad.
- Un buscador compatible con Lucene.
- Robustas clases para autenticación y filtrado de entrada.
- Clientes para servicios Web, incluidos Google Data APIs y StrikeIron.
- Muchas otras clases útiles para hacerlo tan productivo como sea posible.

(29)

Doctrine PHP

Se trata de un potente y completo sistema ORM (Object Relational Mapper) para PHP 5.2+ con un DBAL (Database Abstraction Layer) incorporado. Entre muchas otras cosas ofrece la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos, lo cual constituye su ventaja mas importante. (30)

1.4.4 Control de versiones

Subversion

Es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS[11], producto a las numerosas deficiencias de este último. Es un software libre bajo una licencia de tipo Apache/BSD y se le conoce también como

[11] Concurrent Versions System

SVN por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Ventajas:

- La creación de ramas y etiquetas es una operación muy eficiente. Tiene costo de complejidad constante (O(1)).
- Puede ser servido mediante Apache, sobre WebDAV/DeltaV. Esto permite que clientes WebDAV utilicen Subversion en forma transparente.
- Maneja eficientemente archivos binarios.
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos.
- Los programas asociados a Subversion que se ejecutan por línea de comandos tales como svn pueden ejecutarse tanto en plataformas Unix, Linux, Solaris o Microsoft Windows.

Pero lamentablemente el manejo de cambio de nombres de archivos no es completo. Lo maneja como la suma de una operación de copia y una de borrado. Además no resuelve el problema de aplicar repetidamente parches entre ramas y tampoco facilita el llevar la cuenta de qué cambios se han trasladado. Esto se resuelve siendo cuidadoso con los mensajes de commit.

(31)

TortoiseSVN

TortoiseSVN es un cliente Subversion, implementado como una extensión al Shell de Windows. Es software libre liberado bajo la licencia GNU[12] GPL[13].

Maneja directorios y ficheros a lo largo del tiempo. Estos últimos se almacenan en un repositorio central. El repositorio es prácticamente lo mismo que un servidor de ficheros ordinario, salvo que recuerda todos los cambios que se hayan hecho a sus ficheros y directorios. Esto permite que pueda recuperar versiones antiguas de sus

[12] Gnu is Not Unix

[13] General Public License

ficheros y examinar la historia de cuándo y cómo cambiaron sus datos, y quién hizo el cambio.

Generalidades:

- Puede ser usado sin un entorno de desarrollo.
- Disponible en 28 idiomas diferentes.
- Maneja el mostrar la diferencia de documentos de Office tales como los creados con Microsoft Word.

Entre las características de TortoiseSVN cabe destacar las siguientes:

✓ 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 que no tiene que cambiar a una aplicación diferente cada vez que necesite las funciones del control de versiones.

Los menús contextuales de TortoiseSVN también funcionan en otros administradores de archivos, y en el diálogo Archivo/Abrir que es común a la mayoría de aplicaciones estándar de Windows.

✓ Iconos sobreimpresionados

El estado de cada carpeta y fichero versionado se indica por pequeños iconos sobreimpresionados. 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.

TortoiseSVN proporciona una serie de herramientas internas:

- **TortoiseMerge:** Permite ver las diferencias entre ficheros de texto, fusionar esos cambios e incluso revisar y aplicar ficheros que a menudo son llamados *parches*.
- **TortoiseBlame:** Hace más fácil la lectura de los ficheros de autoría.

- **SubWCRev:** Es un programa de consola para Windows que puede utilizarse para leer el estado de una copia de trabajo local y opcionalmente realizar sustituciones de palabras clave en un fichero plantilla.

(32)

1.4.5 Sistema Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) o DBMA (DataBase Management System) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. Algunos ejemplos de SGBD son Oracle, DB2, PostgreSQL, MySQL, MS SQL Server.

Un SGBD debe permitir:

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarlas, generar informes.

(33)

PostgreSQL 8.3

Es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD[14].

Como muchos otros proyectos Open Source, el desarrollo de PostgreSQL no es manejado por una sola compañía, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo.

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una

14Berkeley Software Distribution.

visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, pues elimina la necesidad del uso de bloqueos explícitos. Otras de sus características son:

- Posee capacidad para soportar cambios, o sea facilidad de mantenimiento y flexibilidad, además de facilidad de prueba.
- Replicación síncrona y asincrónica.
- Cumple con factores que determinan la calidad del software.
- Integridad de los datos: claves primarias, llaves foráneas con capacidad de actualizar en cascada o restringir la acción y restricción not null.
- Características operativas: Corrección, Fiabilidad, Eficiencia, Integridad, Facilidad de uso.
- Resistencia a fallas. Escritura adelantada de registros (WAL) para evitar pérdidas de datos en caso de fallos por: Energía, Sistema Operativo, Hardware.

Y específicamente esta versión del gestor, posee las peculiaridades siguientes:

- *Heap Only Tuples* (HOT), que mejora de forma crítica el rendimiento de las bases de datos que se actualizan con una frecuencia muy alta.
- TSearch2, la avanzada herramienta de búsquedas "full text", incorporada al corazón de PostgreSQL.
- *COMMIT* asincrónico, para unas muchas más rápidas respuestas en algunas transacciones.
- Importación y exportación de XML estándar.

(34) (35)

1.4.6 Entorno de desarrollo integrado (IDE)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic, por ejemplo, puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word.

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic y otros. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en

donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. En este caso se puede poner de ejemplo a Eclipse, al que mediante plugins se le puede añadir soporte de lenguajes adicionales.

El equipo central de arquitectura predefinió en el marco de trabajo el uso del Zend Studio como IDE; mas producto a que dicho aspecto no fue determinado como algo estricto, con el objetivo de proveerle cierta comodidad al equipo de programadores durante su trabajo y acelerar así el desarrollo, el equipo interno de arquitectura de la línea decidió que cada cual utilizara el ambiente de desarrollo según su experiencia y al rendimiento del ordenador de acuerdo al uso de memoria a consumir por cada uno.

(36)

Eclipse

Es un IDE de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Fue desarrollado originalmente por IBM[15] como el sucesor de su familia de herramientas para VisualAge. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE del lenguaje de programación Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entregan como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones clientes.

La base arquitectónica para Eclipse es la Plataforma de cliente enriquecido (del Inglés Rich Client Platform RCP) y está constituido por los siguientes componentes:

- Plataforma principal - inicio de Eclipse y ejecución de plugins
- OSGi - una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT) - un widget toolkit portable.
- JFace – manejador de texto y de archivos, incluye editores de texto.
- El Workbench de Eclipse – posee vistas, editores, perspectivas y asistentes.

La interfaz de usuario de Eclipse también tiene una capa GUI intermedia llamada JFace, que simplifica la construcción de aplicaciones basada en SWT.

[15] International Business Machines

El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico.

(37) (38)

Aptana

Es un entorno de desarrollo dirigido hacia las aplicaciones Web escritas en Ajax/JavaScript. Está basado en Eclipse y es posible encontrarla para las tres plataformas mayoritarias (Window, Mac y Linux), ya sea como plugin del mismo Eclipse, o como aplicación por separado. Las características de este IDE son similares a las de otros más generales: gestión de proyectos, vista outline y vista previa, autocompletado, macros (en este caso, escritos en JavaScript), gestión de documentación, entre otros.

Soporta las librerías más populares: Prototype, Scriptaculous, Dojo, MochiKit, Yahoo UI, Aflax, JQuery y Rico, pudiendo combinarlas fácilmente en la aplicación. Se integra con el navegador preferido por el usuario y permite editar fácilmente HTML, CSS y JavaScript. Una de sus características más peculiares es que precarga las funciones/atributos del DOM y del resto de las especificaciones de los tres lenguajes que soporta, ofreciendo en un ameno tooltip y de un vistazo rápido todos esos datos, junto a los navegadores y versiones que soportan esas funciones/atributos.

(39)

Zend Studio 6.0

Es un editor de texto para páginas PHP que proporciona ayudas, que van desde la creación y gestión de proyectos hasta la depuración del código. Facilita la edición de código, realiza coloreado de la sintaxis y figura la terminación automática del código que es probablemente una de las características más útiles. Considerando el número de comandos PHP, el hecho de que el sistema muestre un breve resumen de lo que cada función realiza, es verdaderamente cómodo. Una lista de parámetros aparece cuando una función se identifica correctamente; hace completamiento de código.

El programa entero está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. 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 del 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 de la parte del servidor,

que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración.

A partir de esta versión el editor dee texto contiene semántica de sensibilización para los View Helpers y apoyo para Zend Framework. Además cuando se va a crear un nuevo proyecto PHP basado en Zend Framework (ZF) es posible escoger las opciones entre ZF 1.0 o ZF 1.5. Por otro lado trae mejoras en los Debuger y ayudantes de código. Además, ofrece soporte a extensiones de PHP y a un ayudante nativo del "Zend Studio for Eclipse" para HTML, JS y CSS.

(40) (41)

1.5 Lenguaje de Modelado

UML 6.1

Es un conjunto de herramientas, que permite modelar (analizar y diseñar) sistemas orientados a objetos. Contiene:

- Diagrama de casos de uso
- Diagrama de clases
- Diagrama de estados
- Diagrama de secuencias
- Diagrama de actividades
- Diagrama de colaboraciones
- Diagrama de componentes
- Diagrama de distribución

UML no es una metodología de desarrollo, por lo que no te va a decir cómo pasar del análisis al diseño y de este al código. No es una serie de pasos que te llevan a producir código a partir de unas especificaciones y al no ser ninguno de los aspectos anteriores es independiente del ciclo de desarrollo a seguir, puede encajar en un tradicional ciclo en cascada, o en un evolutivo ciclo en espiral, o incluso, en los métodos ágiles de desarrollo.

(42)

1.6 Herramienta CASE

Visual Paradigm 3.1

El Visual Paradigm para UML es una herramienta CASE que utiliza como lenguaje de modelado al UML y soporta todas las etapas del ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue.

Ventajas:

- Entorno de creación de diagramas para UML.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs que existen en la actualidad.
- Disponibilidad en múltiples plataformas.

Esta versión soporta además de ingeniería inversa, generación de código, importación desde Rational Rose, exportación/importación XMI, generador de informes, editor de figuras y otros elementos particulares.

(43)

A través del estudio realizado al estado del arte en el cual estará centrada la presente investigación, es posible arribar a la conclusión de que en el desarrollo de todo sistema es tan importante el análisis de la tecnologías y lenguajes a emplear como de las herramientas que servirán de apoyo durante el desarrollo del sistema; ya que esto permitirá explotar mejor sus potencialidades.

Se enfatiza además la necesidad de ser cuidadosos en la adaptación del modelo estructural que se implemente, al estilo de arquitectura seleccionado y ser muy responsable en la determinación de los patrones que serán aplicados, puesto que la calidad de cualquier sistema está estrechamente vinculada al rigor con el que se hayan creado las partes que lo componen y las colaboraciones entre estas.

Mas una vez asimilada la descripción de los elementos que conformarán la arquitectura a desarrollar, no es difícil prever la posibilidad de estructurar un sistema que gestione los procesos concernientes al área de Capital Humano. Debido a que las ventajas asociadas estos aspectos básicos, advierten su capacidad para esquematizar procesos de negocio de tal envergadura.

CAPÍTULO II: PROPUESTA DE ARQUITECTURA

Este capítulo trata acerca del trabajo realizado para la definición de los componentes a partir de los requisitos capturados en el negocio. Mediante una panorámica se abordan las implicaciones de: la creación de los componentes, la integración asociada a cada componente, la priorización de componentes de acuerdo con los requisitos que contengan. Es importante tener en cuenta que elementos tales como el lenguaje de programación, las herramientas, las tecnologías y los estilos arquitectónicos, fueron predefinidos por el equipo de arquitectura central del proyecto.

2.2 Descripción General de la Arquitectura Subsistema Capital Humano.

El subsistema Capital Humano está compuesto fundamentalmente por los módulos referidos a continuación:

El primero de los mismos es Persona en el cual se manejan todos los datos personales necesarios a tener en cuenta cuando un individuo se convierte en trabajador.

Un elemento primordial en la arquitectura del sistema en cuestión, lo constituye el módulo Trabajador: encargado de gestionar los datos respectivos a los trabajadores. Su importancia radica, especialmente, en que en el mismo será controlado cada cambio asociado al trabajador de la empresa.

Otro módulo esencial es Nómina, pues tiene como función principal gestionar todo lo referente a conceptos de pago, y el resto de las operaciones que tributan de una forma u otra a la elaboración de la nómina de cada trabajador. Los procesos que atiende son imprescindibles en el negocio tratado.

Submayores por su parte más que crucial es complejo, ya que gestiona los procesos de retenciones asociados al trabajador y la acumulación de vacaciones de los mismos. También tramita los procedimientos que giran en torno a dichos procesos y le sirven de soporte.

Existe, además, un módulos que atiende las operaciones vinculadas a la gestión de los puestos de trabajo de la entidad. El mismo, entre otros aspectos, gestiona la asignación de pagos adicionales a los trabajadores a través de su puesto de trabajo.

Y precisamente Pagos Adicionales es otro de los módulos modelados en la arquitectura de sistema de la línea Capital Humano, mediante este serán concedidos, anulados y modificados los diferentes pagos adicionales, según se estipule.

Por otro lado, se tiene que las relevancias positivas y negativas relacionadas con los trabajadores serán manejadas mediante el módulo Incidencias. Entre las mismas se destacan los impuestos.

Por último está el módulo Cierre, cuya tarea principal es llevar a cabo el proceso de Cierre Contable en las entidades.

Es preciso destacar que el diseño de la arquitectura del presente subsistema obedece un modelo arquitectónico más general donde además del uso de las herramientas, lenguajes, tecnologías (frameworks) y otros aspectos, fueron definidos estándares y una serie de artefactos organizados a los que fue necesario ajustar el diseño e implementación de los diferentes componentes definidos, así como la integración de los mismos. El esquema del sistema Cedrux, además, provee al esquema del subsistema Capital Humano elementos de reutilización que resultaron imprescindibles en el desarrollo de su arquitectura.

2.3 Arquitectura de Negocio

La arquitectura del negocio puede verse como un conjunto de elementos relacionados entre sí por medio de una funcionalidad determinada que representa la estructura y la organización de un “Sistema de Negocio”. Incluye, a un nivel de abstracción alto, la descripción de los procesos y estructuras básicas del negocio. Se usa para facilitar la comprensión y el análisis del sistema en base a un estudio del negocio.

Una de las funciones importantes de la arquitectura del negocio es marcar la base sobre el negocio y las necesidades que de él se generan, ya que aquí, desde el punto de vista del negocio, se describe qué tendrá que hacer el sistema y cómo tendrá que hacerlo. Sirve de punto de partida para el desarrollo del software.

2.3.1 Análisis del modelo conceptual de la arquitectura de negocio

El modelo conceptual está formado por 28 conceptos propios de la línea y 3 conceptos externos que son: Elemento de gasto, que pertenece a Costos y Procesos;

Cuenta, que pertenece a Contabilidad y Plantilla de Cargo que pertenece a Estructura y Composición, con los cuales tienen relación.

Entre los principales conceptos se encuentran:

Trabajador (En esta entidad se registran los trabajadores de la empresa.) Tiene relación con otras 6 entidades que son: Persona (un Trabajador es una Persona), Tipo Puesto (un Trabajador tiene un Tipo de Puesto), Modelo de Movimiento de Nómina (un Trabajador tiene un Modelo de Movimiento de Nómina), Registro de Incidencia (un Trabajador tiene varios Registros de Incidencias), Registro de Pagos por Trabajador (un Registro de Pagos por Trabajador tiene varios Trabajadores), Registro de Retenciones (un Trabajador tiene un Registro de Retenciones).

Modelo de Movimiento de Nómina (Esta entidad representa los movimiento de nóminas que son documentos mediante los que se modifican datos del trabajador, que deben quedar registrados.) Se relaciona con otras 6 entidades: Trabajador (un Trabajador tiene un Modelo de Movimiento de Nómina), Grupo de Puesto (un Grupo de Puesto tiene un Modelo de Movimiento de Nómina), Tipo Puesto (un Tipo de Puesto tiene un Modelo de Movimiento de Nómina), Registro de Pagos Adicionales (un Registro de Pagos Adicionales tiene un Modelo de Movimiento de Nómina), Tipo de Modelo (un Modelo de Movimiento de Nómina tiene un Tipo de Modelo), Registro de Pagos por Trabajador (un Registro de Pagos por Trabajador tiene un Modelo de Movimiento de Nómina).

Nómina (La nómina contiene pagos de muy diversas naturaleza por trabajadores.) Tiene relación con otras 5 entidades que son: Período de Pago (una Nómina pertenece a un Período de Pago) y Tipo de Nómina (una Nómina es de un Tipo de Nómina). Además, una Nómina está asociada a un Comprobante de Operaciones, a un Submayor de Vacaciones y a uno de Retenciones.

Incidencia (En esta entidad se definen los distintos tipos de incidencias.) Tiene relación con otras 6 entidades que son: Registro de Incidencia (un Registro de Incidencia tiene varias Incidencias), Clasificación (una Incidencia tiene una Clasificación), Tipo de Nómina (una Incidencia es de un Tipo de Nómina), Impuesto por Incidencia (una Incidencia tiene un Impuesto por Incidencia). Por otro lado cada incidencia genera un Elemento de Gasto y está asociado a una Cuenta.

Además están:

- Comprobante de operaciones (Registra las operaciones contables del módulo de Nómina.)
- Documento de Ajuste (Son modificaciones a los documentos de nóminas, corrigen errores de los mismos.)
- Período de pago (En esta entidad se definen los distintos períodos de pago de la empresa.)
- Grupo de puesto (En esta entidad se definen los grupos de puesto de trabajo. Son grupos de puestos con características similares.)
- Puesto de Trabajo (En esta entidad se definen los distintos puestos de trabajo.)
- Registro de impuesto (En esta entidad se registra por cada incidencia el o los impuestos asociados que debe pagar la empresa por la misma en caso de que lo tenga.)
- Registro de Incidencias (En esta entidad se registran las incidencias de un trabajador en un período de pago.)
- Registro de Pagos Adicionales por trabajador (Registra los pagos adicionales asociados a los trabajadores.)
- Registro de Pagos Adicionales por grupo de puesto (Registra los pagos adicionales asociados a grupo de puesto de trabajo determinado.)
- Submayor de Retenciones (Registro de las retenciones del trabajador.)
- Submayor de vacaciones (Registra el acumulado para las vacaciones de los trabajadores.)
- Tipo de ajuste (En esta entidad se definen los tipos de ajustes.)
- Tipo de Pago Adicional (En esta entidad se definen los distintos tipos pagos adicionales que son fijos al trabajador.)
- Tipo de impuesto (En esta entidad se definen los distintos conceptos de impuestos empresariales que afectan a la empresa.)
- Tipo incidencia (En esta entidad se definen los distintos tipos de incidencias.)
- Tipo movimiento nómina (En esta entidad se definen los distintos tipos de movimientos de nóminas.)
- Tipo Nómina (En esta entidad se definen los distintos tipos de nóminas.)
- Tipo de Retención (En esta entidad se definen los distintos tipos de retenciones.)

Han sido separados un grupo de cuatro conceptos del resto y se ha expuesto la relación que guardan con los demás conceptos. Lo anterior se debe a que son los conceptos de mayor concurrencia dentro del modelo conceptual.

2.3.2 Análisis de la Arquitectura de Negocio

A través de una exploración de los procesos de la organización que serán modelados, fueron identificados como los requisitos más horizontales y de mayor nivel de referencia en los procesos, los vinculados directamente a la gestión de:

1. Los datos de las personas
2. Los pagos adicionales
3. Los puestos de trabajo que serán asignados individualmente a los trabajadores
4. Los grupos de puestos de trabajo disponibles en el centro laboral

También pertenecen al orden de requisitos más elementales del negocio desde el punto de vista arquitectónico los de:

5. Emisión de movimiento de nómina
6. Procesamiento de nómina
7. Apertura y cierre de submayor de vacaciones

Y por último se incluye la gestión de todos los conceptos que requieren ser clasificados, puesto que constituyen la razón de ser de los nomencladores del sistema, entre ellos se encuentran:

8. Los tipos de impuestos que afectan el salario del trabajador a raíz de deudas contraídas por el mismo, lo cual se traduce en retenciones.
9. Los tipos de programas que permiten definir cuál será aplicado a los trabajos en la empresa, para hacer el descuento por una retención determinada.
10. Los tipos de incidencias que reducen o aumentan el salario del trabajador y se utilizan para el cálculo del salario a devengar, utilizando las cuentas de Contabilidad[16] y los elementos de gasto de Costo y Procesos[17].

[16] El módulo de Contabilidad General del ERP Cubano encapsula funcionalidades que representan el núcleo del sistema de planificación de recursos empresariales y norman las principales actividades contables y financieras.

11. Los tipos de impuestos que se tendrán en una entidad y que serán, posteriormente, asociados a una determinada incidencia.
12. Los tipos de período de pago
13. Los tipos de nóminas
14. Los tipos de ajustes

Durante los primeros análisis arquitectónicos de los requisitos se hizo notable que los requisitos que comprenden las principales colaboraciones entre los procesos son los reflejados en la tabla siguiente:

Tabla 1 Síntesis de colaboraciones.

Colaboración	Requisito síntesis
Pagos adicionales asignados al puesto de trabajo.	Gestionar Asociación de pagos adicionales por puesto de trabajo.
Determinadas incidencias se relacionan con impuestos específicos.	Gestionar asociación de incidencia-impuesto.
A un determinado trabajador se le asigna un puesto de trabajo.	Emitir movimiento de nómina
A un determinado trabajador se le asigna algún pago adicional.	

Posteriormente la observación minuciosa reveló que el conjunto de procesos que el área de Capital Humano provee o contrata son los especificados a continuación:

Tabla 2 Procesos proveídos y contratados.

Proceso proveedor	Proceso contratista	Recurso	Área externa
Emitir Comprobante		Comprobante de operaciones	Contabilidad
	Conciliación con	Saldo	Contabilidad

[17] El módulo de Costos y Procesos se encarga de automatizar toda la gestión de los costos en la entidad que lo utilice, pudiendo medir la suma de gastos de toda naturaleza expresada monetariamente y aplicada a una producción o servicio determinado de la empresa.

	contabilidad del submayor de retenciones		
	Adicionar tipo de incidencia	Nomenclador de elemento de gasto y centro de costo	Costo y Proceso
		Nomenclador de cuentas	Contabilidad
	Adicionar Puesto de trabajo	Área y cargo o plantilla de cargos	Estructura y Composición
	Conciliación con contabilidad del submayor de vacaciones	Saldo	Contabilidad
	Adicionar tipo de impuesto	Nomenclador de cuentas	Contabilidad
	Adicionar tipo de impuesto	Nomenclador de elemento de gasto y centro de costo	Costo y Proceso
	Adicionar pago adicional	Nomenclador de elemento de gasto y centro de costo	Costo y Proceso
	Cierre	Período	Configuración

2.3.3 Agrupamiento de los requerimientos funcionales por módulo

Durante el flujo de análisis del negocio se planteó que, de acuerdo a los procesos básicos que se llevaban a cabo en el departamento de Capital humano de las empresas, la aplicación debía poseer la siguiente estructura de módulos que encierran a requisitos con sus respectivos sub-requisitos:

Tabla 3 Distribución de requisitos funcionales por componente.

Componente	Agrupación de requisitos	Requisitos

Persona	Gestionar datos de persona	○ Adicionar persona ○ Modificar persona ○ Eliminar persona
Pagos Adicionales	Gestionar pago adicional	○ Adicionar pago adicional ○ Modificar pago adicional ○ Eliminar pago adicional
Puesto de Trabajo	Gestionar puesto de trabajo	○ Adicionar puesto de trabajo ○ Modificar puesto de trabajo ○ Eliminar puesto de trabajo
	Gestionar grupo de puesto de trabajo	○ Adicionar grupo puesto de trabajo ○ Modificar grupo puesto de trabajo ○ Eliminar grupo puesto de trabajo ○ Generar puestos de trabajo
	Gestionar Asociación de pagos adicionales por puesto de trabajo	○ Adicionar pago adicional a grupo de trabajo ○ Modificar pago adicional de un grupo de trabajo Eliminar pago adicional al grupo de trabajo
Trabajador	Emitir movimiento de nómina	○ Movimiento de altas ○ Movimiento de reubicación ○ Movimiento de bajas ○ Actualizar datos contables
Incidencias	Gestionar tipo de impuesto	○ Adicionar tipo de impuesto ○ Modificar tipo de impuesto ○ Eliminar tipo de impuesto
	Gestionar tipo de incidencia	○ Adicionar tipo de incidencia ○ Modificar tipo de incidencias ○ Eliminar tipo de incidencia
	Gestionar	○ Adicionar asociación de

	asociación de incidencia-impuesto	incidencia-impuesto o Eliminar asociación de incidencia-impuesto
	Gestionar registro de incidencia	o Adicionar registro de incidencia o Modificar registro de incidencia o Eliminar registro de incidencia
Submayores	Gestionar tipo de retención	o Adicionar tipo de retención o Modificar tipo de retención o Eliminar tipo de retención
	Gestionar tipo de programa	o Adicionar tipo de programa o Modificar tipo de programa. o Eliminar tipo de programa
	Apertura y Cierre Submayor de Vacaciones	o Apertura del submayor de vacaciones o Cierre del submayor de vacaciones
		o Actualización del submayor de vacaciones
		o Detalles del submayor de vacaciones
		o Conciliación con contabilidad del submayor de vacaciones
	Gestionar retenciones	o Apertura del submayor de retenciones o Modificar Registro Retenciones-Trabajador o Cierre del submayor de retenciones
		o Detalles del submayor de retenciones de un trabajador
		o Actualizar Submayor de Retención

	Configuración de submayor de vacaciones	○ Definir porciento de vacaciones ○ Definir cuenta de vacaciones
Nómina	Gestionar tipo de período de pago	○ Adicionar período de pago ○ Modificar período de pago ○ Eliminar período de pago
	Gestionar tipos de ajustes	○ Adicionar tipos de ajustes ○ Modificar tipos de ajustes ○ Eliminar tipos de ajustes
	Gestionar tipos de nóminas	○ Adicionar tipo de nómina ○ Modificar tipo de nómina ○ Eliminar tipo de nómina
	Gestionar documentos de ajustes	○ Crear documento de ajuste ○ Modificar documento de ajuste ○ Eliminar documento de ajuste
	Procesar Nómina	○ Crear nómina ○ Agregar trabajadores a la nómina ○ Procesar nómina
		○ Procesar documento de ajuste
	Revisar nómina	○ Confirmar nómina (aceptar, revertirla, eliminarla) ○ Imprimir Nómina
		○ Emitir Comprobante
	Configuración de nómina	○ Definir si se aplica el redondeo
Cierre	Cierre	○ Cierre contable

2.4 Aplicación de los patrones

MVC

En la implementación específica de dicho patrón en el sistema Capital Humano la Vista, que viene siendo una especie de capa de presentación y está englobada en el paquete *views*, le brinda la posibilidad al usuario de interactuar con el Controlador mediante los mensajes de eventos y además le permite ver las respuestas a sus peticiones.

El Controlador, por su parte representa el gestor de eventos del sistema y se encuentra dentro del paquete c*ontrollers*; este atiende los pedidos que llegan de la Vista accediendo al paquete *models*. Este último paquete a su vez está dividido en los paquetes *business* y *domin*, respectivamente, el primero encierra las clases donde se realiza la lógica del negocio y el segundo comprende las clases del dominio, o sea, las que leen y/o escriben datos en las tablas de la base de datos.

El patrón fue reimplementado, por el equipo central de arquitectura, de forma tal que las clases controladoras de los diferentes componentes tienen la posibilidad de obtener los datos mediante instanciaciones a objetos de clases del modelo del negocio, o tomándolos directamente del modelo del dominio.

Empleo de otros patrones

Luego de varios talleres realizados en la línea, internamente fue aprobada la aplicación de solo tres patrones Gof (Gan of Four) entre todos los analizados y de cinco patrones GRAPS ("General Responsibility Assignment Software Patterns"). Los mismos son representados a continuación:

Tabla 4 Aplicación de patrones GoF por componente.

Patrón GoF / Componente	Fachada	Composición	Mediador
Nómina	X	X	X
Submayores	X		
Puesto de Trabajo	X	X	X
Incidencias	X		
Trabajador	X		X

Persona	X		
Pagos Adicionales	X		X
Cierre			

Tabla 5 Aplicación de patrones GRAPS por componente.

Patrón GRASP / Componente	Alta Cohesión	Bajo Acoplamiento	Experto	Controlador	Creador
Nómina	X	X	X	X	X
Submayores	X	X	X	X	X
Puesto de Trabajo	X	X	X	X	X
Incidencias	X	X	X	X	X
Trabajador	X	X	X	X	X
Persona	X	X	X	X	X
Pagos Adicionales	X	X	X	X	X
Cierre	X	X	X	X	X

El patrón estructural aplicado en prácticamente todos los componentes del subsistema fue el patrón Fachada (solamente no se usó en Cierre) Mediante el mismo se logró proporcionar una interfaz de servicios simple para un subsistema tan complejo como lo es Capital Humano.

Su uso en el nivel de abstracción más alto, obedece el diseño general de la arquitectura del sistema, puesto que la existencia de dependencias entre el subsistema Capital Humano y otros como: Contabilidad y Costos y Procesos, solicita la implementación de una *clase fachada* donde se publiquen los servicios necesarios para los otros subsistemas, para facilitar la interacción.

A nivel interno también se emplea este patrón para disminuir el grado de dependencia entre los componentes. De este modo se garantiza además, hasta cierto

punto, la portabilidad del subsistema e incluso de sus componentes, también aporta un bajo acoplamiento. Las clases fachada definidas son:

- PersonaService
- PuestoTrabajoService
- PagosAdicionalesService
- TrabajadorService
- IncidenciasService
- SubmayorVacService
- SubmayorRetService
- RegRetencionService
- NominaService
- PeríodoPagoService

El número de clases fachada excede la cantidad de componentes en los que fue empleado puesto que existe el caso de Submayores y Nómina que encierran numeroso procesos y se consideró pertinente crear en todos los componentes en semejante situación, más de una clase Servicio para reforzar al mismo tiempo el empleo del método de asignación de responsabilidades: Experto y mantener además una alta cohesión.

El patrón Composición fue necesario emplearlo puesto que de manera general se pretendía que cuando cualquier cliente requiriera un servicio, no reparara en las diferencias entre objetos simples y compuestos, lo cual se evidencia con exactitud en la asignación igualitaria de pagos adicionales tanto al grupo de puestos de trabajo como al concepto embebido en este: puesto de trabajo.

También se manifiesta mediante la composición de “procesamiento de nómina”, que representa en este caso el elemento jerárquico complejo y está compuesto por la nómina a procesar, que constituye el elemento simple.

Durante la construcción de las tablas de la base de datos se hizo notable en varios componentes la existencia de un conjunto de objetos cuya comunicación estaba bien definida, pero era compleja, las interdependencias que surgían estaban poco estructuradas y eran difíciles de entender. Resultó entonces engorroso reutilizar determinados objetos, pues lo dificultaba la cantidad de referencias que tenía a otros objetos.

Para resolver dicho problema la solución encontrada fue aplicar el patrón de comportamiento Mediador, creando una nueva tabla entre todas las tablas unidas mediante una relación de muchos a muchos. La tabla mediadora posee una relación de uno a muchos con las vinculadas a ella. De esta forma queda adaptado a las circunstancias un comportamiento distribuido entre varias clases. Dicha operación se hizo entre tablas que asociaban a los componentes Pagos Adicionales y Puesto de Trabajo donde la mediadora fue dat_plantillapuestotrabajopagosadicionales, para asociar Pagos Adicionales y Trabajador (dat_registropagosadicionalestrabajador).

También se aplicó internamente en Nómina donde el identificador del Registro de impuesto y el de Tipo de ajuste pasaron a ser llaves foráneas de la tabla dat_registroimpuestotipoajuste.

Durante el diseño de la arquitectura del subsistema con el propósito de adoptar los patrones de asignación de responsabilidades se tuvo en cuenta que: la optimización de código exige que en todo sistema globalmente sea creada una clase que se encargue de atender los eventos vinculados al mismo. Lo cual aplicado a los componentes se interpreta como la necesidad de que existan clases controladoras de los eventos asociados a los diferentes procesos gestionados en el módulo.

Un evento de todo sistema es un evento de alto nivel generado por un actor externo. Se asocia a operaciones del sistema: las que emite, precisamente, en respuesta a los eventos del sistema. Por su parte, un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar los eventos del sistema. Define además el método de su operación. De modo que en todos los componentes fue asignada la responsabilidad del manejo de eventos a clases que constituyeran los controladores globales de los mismos.

El patrón Experto se puso en práctica en todos los componentes, al partir de que es el principio fundamental de asignación de las responsabilidades durante el diseño, pues siempre se le debe atribuir la determinada responsabilidad al *experto en la información*, es decir, la clase que cuenta con la información necesaria para cumplir dicha responsabilidad.

Con el objetivo de mantener la complejidad de los componentes dentro de límites manejables se acordó asignar cada responsabilidad de forma tal que la cohesión siguiera siendo alta. La cohesión es una medida de cuan relacionadas con otras, o

enfocadas en sí misma están las responsabilidades de una clase. En dicho sentido una clase con baja cohesión hace muchas cosas no afines, o un trabajo excesivo y a menudo, representan un alto grado de abstracción, o han asumido responsabilidades que deberían haber delegado a otros objetos. No conviene este tipo de clases pues son difíciles de comprender, conservar y reutilizar. Además son delicadas, pues las afectan constantemente los cambios.

Para dar soporte a una dependencia escasa y al aumento de la reutilización se siguió el patrón de Bajo Acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases. Una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente, ya que presentan los siguientes problemas:

- Los cambios de las clases afines ocasionan cambios locales.
- Son más difíciles de entender cuando están aisladas.
- Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

En torno a todo lo anterior se determinó que fueran asignadas las responsabilidades en todos los componentes del subsistema de forma tal que se garantizara en el diseño un bajo acoplamiento.

Un ejemplo que refleja claramente el bajo acoplamiento es que la clase GestionarpuestodetrabajoController realiza una integración con el componente Trabajador y no conoce la existencia de la clase NomTipomovnomina de dicho componente que es la que contiene los datos solicitados, simplemente conoce que la fachada le pasa los valores que pide.

Tomando como guía al patrón Creador para designar el responsable de crear una nueva instancia de alguna clase, se consideró otorgarle a la clase A la responsabilidad de crear una instancia de la clase B en cualquiera de los siguientes casos:

1. A agrega los objetos B.
2. A contiene los objetos B.
3. A registra las instancias de los objetos B.
4. A utiliza específicamente los objetos B.

5. A tiene los datos de inicialización que serán trasmitidos a B cuando este objeto sea creado (así que A es un Experto respecto a la creación de B).

6. A es un creador de los objetos B.

De acuerdo a lo anterior se llegó a la conclusión de que en todos los componentes existían *clases creadoras*, puesto que entre otras cosas, todos responden específicamente al caso 4, su gran mayoría esta comprendido también en el caso 1 y además se manifiesta el caso 6.

2.5 Formalización de los componentes

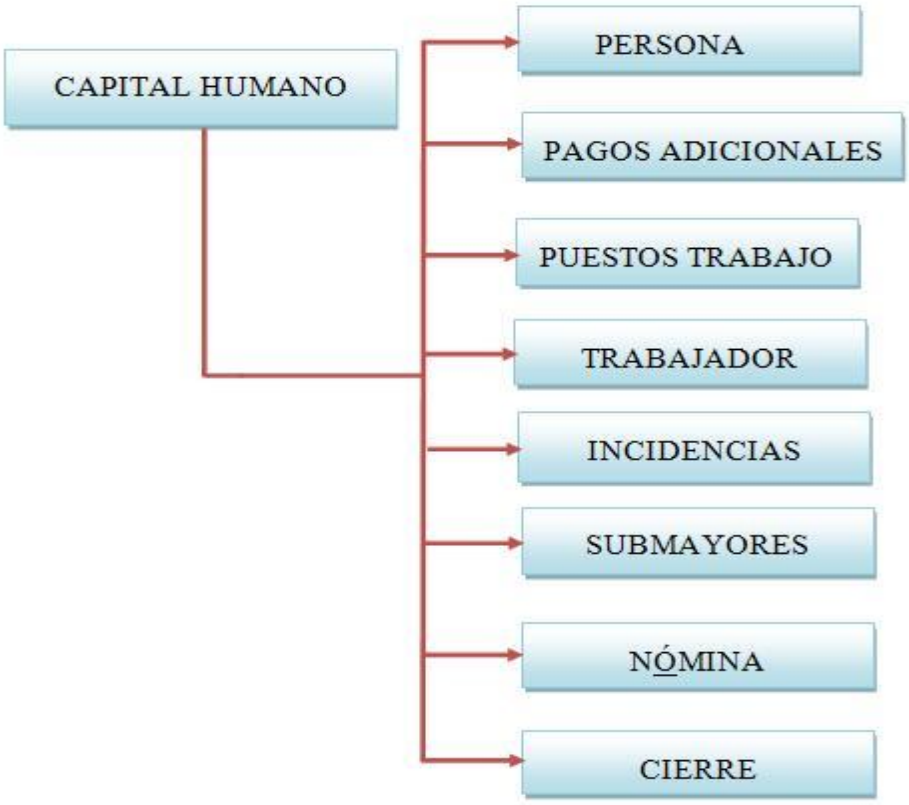


Figura 10 Gráfico estructural de la Arquitectura del Subsistema Capital Humano.

2.6 Línea Base

La arquitectura que se presenta define la línea que deben seguir los procesos de análisis, diseño e implementación. Toma en cuenta para ello las mejores prácticas y estilos vigentes en la actualidad referentes a la Arquitectura de Software, Permita a los desarrolladores y demás involucrados tener una idea clara de lo que debe ser

implementado. La línea base permitirá definir los elementos de configuración, asi como guiar y evaluar el desarrollo del proyecto, según las normas definidas por la Arquitectura,

Subsistema Capital Humano

1. **Componente Persona**
   - R1. Adicionar Persona
   - R2. Modificar Persona
   - R3. Eliminar Persona
2. **Componente Puesto de Trabajo**
   - R4. Adicionar Puesto de trabajo
   - R5. Modificar Puesto de Trabajo
   - R6. Eliminar Puesto de Trabajo
   - R7. Generar puestos de trabajo
   - R8. Recuperar datos de Puesto de Trabajo
   - R9. Adicionar grupo puesto de trabajo
   - R10. Modificar grupo puesto de trabajo
   - R11. Eliminar grupo puesto de trabajo
   - R12. Recuperar grupo datos del puesto de trabajo
   - R13. Adicionar pago adicional a grupo de trabajo
   - R14. Modificar pago adicional de un grupo de trabajo
   - R15. Eliminar pago adicional al grupo de trabajo
3. **Componente Pagos Adicionales**
   - R16. Adicionar Pago Adicional
   - R17. Modificar Pago Adicional
   - R18. Eliminar Pago Adicional
   - R19. Recuperar Datos de Tipo de Pago Adicional
4. **Componente Trabajador**
   - R20. Movimiento de Alta
   - R21. Movimiento de Reubicación
   - R22. Movimiento de Bajas
   - R23. Actualizar Datos Contables Trabajador
5. **Componente Incidencias**
   - R24. Adicionar Asociación de Incidencia-Impuesto

R25. Eliminar Asociación de Incidencia-Impuesto

R26. Adicionar Registro de Incidencia

R27. Modificar Registro de Incidencia

R28. Eliminar Registro de Incidencia

R29. Adicionar Tipo Impuesto

R30. Modificar Tipo Impuesto

R31. Eliminar Tipo Impuesto

R32. Adicionar Tipo Incidencias

R33. Modificar Tipo Incidencias

R34. Eliminar Tipo Incidencias

6. Componente Nómina

R35. Crear Nómina

R36. Agregar trabajadores a la nomina

R37. Adicionar Períodos de Pago

R38. Modificar Períodos de Pago

R39. Eliminar Períodos de Pago

R40. Confirmar Nómina

R41. Imprimir Nómina

R42. Crear Documento de Ajuste

R43. Modificar Documento de Ajuste

R44. Eliminar Documento de Ajuste

R45. Adicionar Tipos de Ajuste

R46. Modificar Tipos de Ajuste

R47. Eliminar Tipos de Ajuste

R48. Procesar Documento de Ajuste

R49. Adicionar Tipo de Nómina

R50. Modificar Tipo de Nómina

R51. Eliminar Tipo de Nómina

R52. Emitir Comprobante Operaciones

7. Componente Submayores

R53. Adicionar Tipo de Retención

R54. Modificar Tipo de Retención

R55. Eliminar Tipo de Retención

R56. Adicionar Tipo de Programa

R57. Modificar Tipo de Programa

R58. Eliminar Tipo de Programa

R59. Apertura Sub Mayor de Vacaciones

R60. Cierre Sub Mayor de Vacaciones

R61. Actualizar Sub Mayor de Vacaciones

R62. Detalles Sub Mayor de Vacaciones

R63. Conciliación con Contabilidad de Sub Mayor de Vacaciones

R64. Apertura Sub Mayor de Retenciones

R65. Modificar Registro Retenciones-Trabajador

R66. Cierre Submayor de Retenciones

R67. Detalles Sub Mayor de Retenciones de un Trabajador

R68. Actualizar Sub Mayor de Retenciones

R69. Definir porciento de Vacaciones

R70. Definir cuenta de Vacaciones

2.7 Responsabilidad arquitectónica de los componentes

El subsistema Capital Humano se encarga de administrar los recursos de una persona, desde el proceso de selección para realizar el contrato del trabajador hasta emitir la nómina. De acuerdo con los requisitos capturados en el negocio se identificaron 8 componentes:

- Persona: Administra todos los datos de la persona.
- Puesto de Trabajo: Gestiona los puestos de trabajo asociados a un cargo que pertenecen a un área determinada.
- Trabajador: Administra datos contables así como los movimientos de alta, baja y/o reubicación en un puesto de trabajo.
- Pagos Adicionales: Administra los pagos adicionales de un trabajador y los asocia además a un puesto de trabajo.
- Incidencias: Registra las incidencias de un trabajador y por un período de pago.
- Submayores: Actualiza los submayores de vacaciones y retenciones de un trabajador, y verifica que el saldo del submayor coincida con el saldo de contabilidad.
- Nómina: Procesa la nómina de un área de trabajo, teniendo en cuenta las incidencias y los pagos adicionales de los trabajadores.

- Cierre: Realiza el proceso de cierre contable teniendo en cuenta datos de la Nómina.

2.8 Elementos globales utilizados por el Subsistema Capital Humano

Los elementos globales (EG) son conceptos generales definidos a nivel central y les que posibilitan al sistema trabajar de acuerdo a datos actuales en tiempo y espacio. A continuación se muestran, por componente, los elementos globales que fue necesario utilizar en Capital Humano:

Tabla 6 Distribución del uso de elementos globales por componentes.

EG / Componente	Estructura	Subsistema	Período	Ejercicio	Fecha Contable
Persona					
Trabajador	X				
Puesto de Trabajo	X				
Pagos Adicionales	X				
Incidencias	X				
Submayores	X				X
Nómina	X				
Cierre	X	X	X	X	X

2.9 Diagrama de componentes

Una vez que están identificados los módulos en los que se va a dividir el subsistema, así como los elementos que pueda necesitar de otros subsistemas, el que atiende, específicamente, los procesos de Capital Humano; se procede a conformar el diagrama de componentes. En este artefacto quedan claras las interfaces provistas y requeridas por componentes. Las primeras se traducen en lo que sale de cada componente para que sea utilizado por otro y las otras interfaces representan las necesidades de cada componente, o sea, los elementos que solicitan estos de otro componente (interno o externo) para poder realizar su función. El diagrama se muestra a continuación:

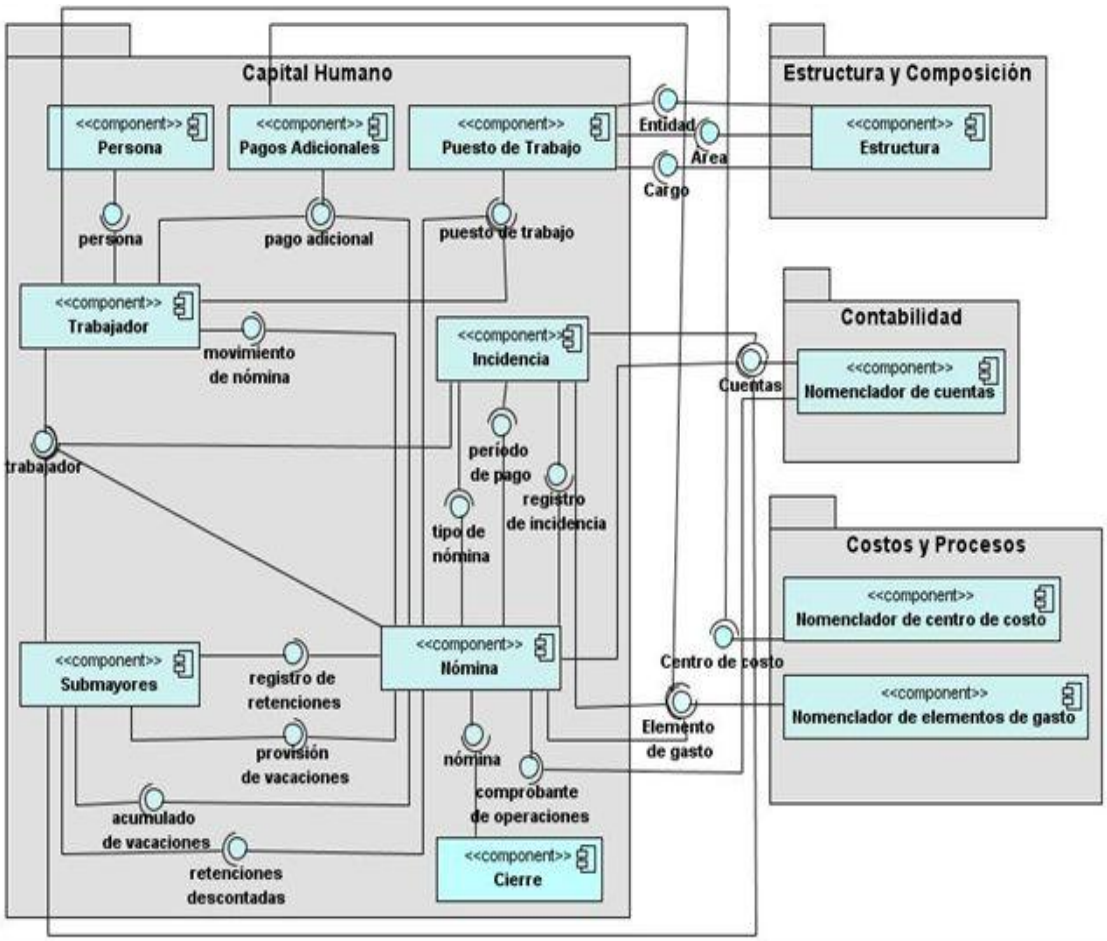


Figura 11 Integración de componentes del Subsistema Capital Humano.

2.10 Priorización de los requisitos

La actividad tiene como responsabilidad realizar la priorización de los requisitos, con el fin de identificar los de mayor significancia arquitectónica, los cuales por su impacto en el desarrollo tendrán mayor atención por parte del equipo de desarrollo en la gestión de los requisitos.

$$SARF = \text{CD+CC+CCP+CN}$$

DONDE:

SARF: Significancia arquitectónica de un requisito funcional.

CD: Cantidad de dependencias de otro requisito asociado a él.

CC: Cantidad de conceptos.

CCP: Cantidad de conceptos persistentes.

CN: Complejidad del negocio.

Anexo 1 Priorización de los requisitos..

2.11 Priorización de los componentes

En el subsistema se identificaron como se describió anteriormente 8 componentes, que son:

- Persona
- Puesto de Trabajo
- Pagos Adicionales
- Trabajador
- Incidencias
- Nomina
- Submayores
- Cierre

A la hora de decidir que componentes desarrollar primero, se tuvo en cuenta que los que generan mayor cantidad de dependencia y a su vez son los más complejos se desarrollan primero. Mas para determinar esto se hizo un estudio donde fue analizado por cada componente su complejidad. Para ello fue tomado en cuenta requisito por requisito de cada componente y se analizó su complejidad (la cual oscila entre 1 y 10); la cantidad de entidades que lo atienden, la cantidad de gestores de negocio que utiliza, así como el número de tablas de la base de datos que son necesarias para la implementación del requisito en cuestión.

De la suma de estos factores resulta una variable que se le conoce como factor de esfuerzo, el cual sumado a la prioridad del negocio (1-10) da como resultado otra variable que representa la prioridad final.

Este análisis se lleva a cabo con cada requisito del componente y cuando ya se tienen todos los valores de las prioridades finales, estas se suman y dan como resultado una variable que se denomina: *Complejidad del componente*, la cual va a decidir el orden en que van a ser desarrollados los componentes.

Luego de hacer este análisis en todos los componentes de la línea, quedaron ordenados de la siguiente forma:

Tabla 7 Valor de complejidad por componente.

Componente	Complejidad componente
Nómina	282
Submayores	256
Puesto de Trabajo	236
Incidencias	196
Trabajador	100
Persona	92
Pagos Adicionales	64
Cierre	13

En los Anexos se encuentra llevado hasta el nivel de detalle de requisitos la relacionada con el anterior análisis hecho por componente.

Anexo 2 Priorización de los componentes

2.12 Diagramas de clases del diseño

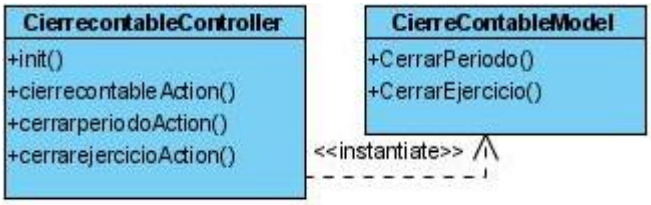


Figura 12 Diagrama de clases del diseño del componente Cierre.

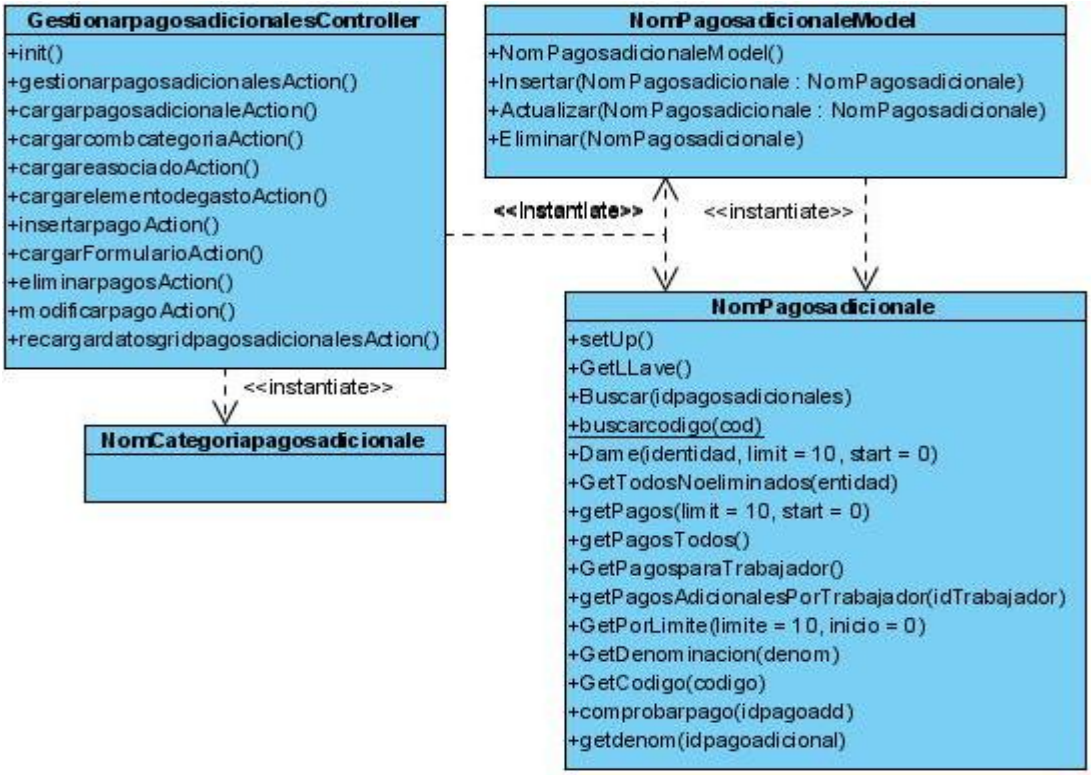


Figura 13 Diagrama de clases del diseño del componente Pagos Adicionales.

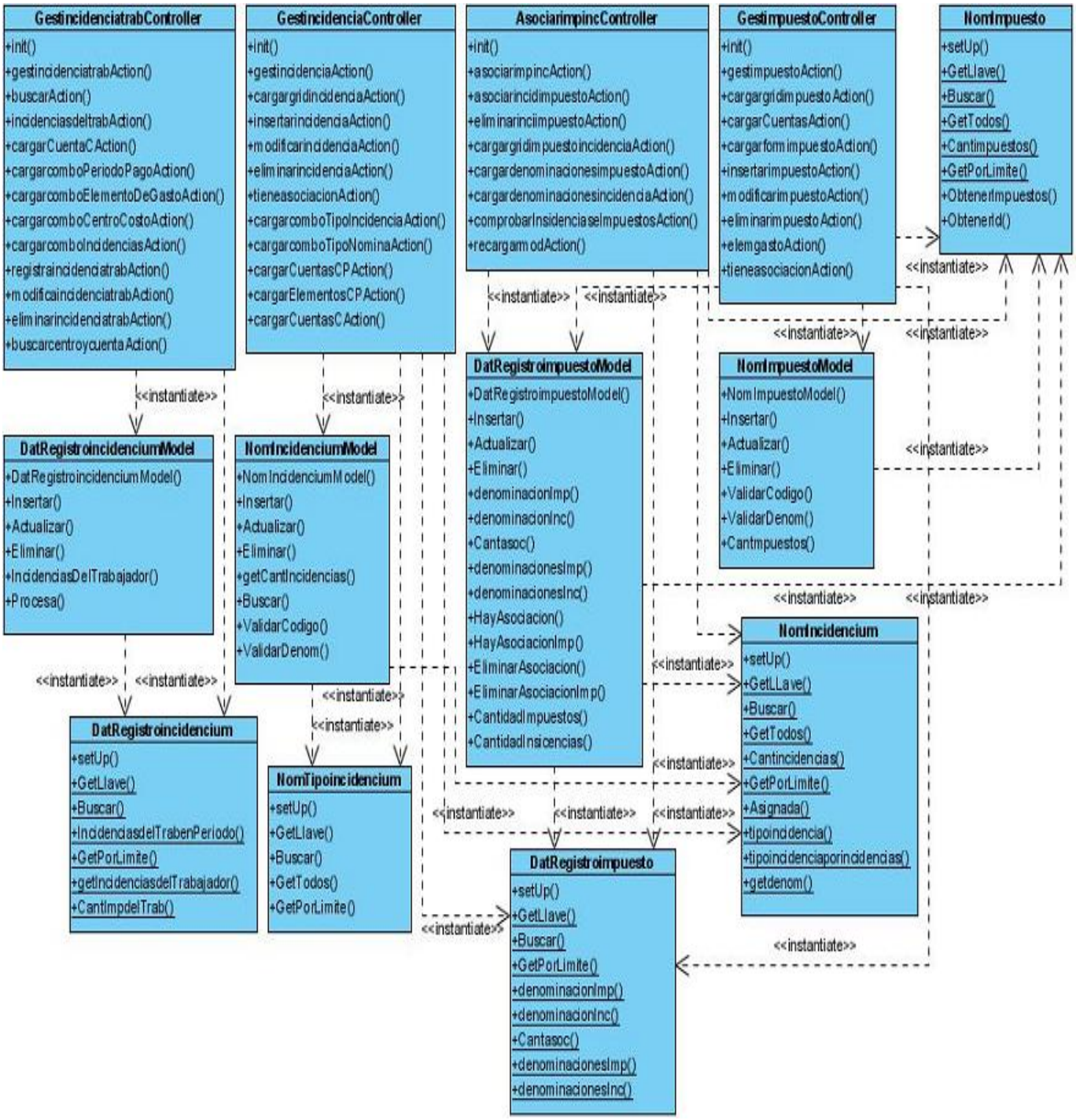


Figura 14 Diagrama de clases del diseño del componente Trabajador.

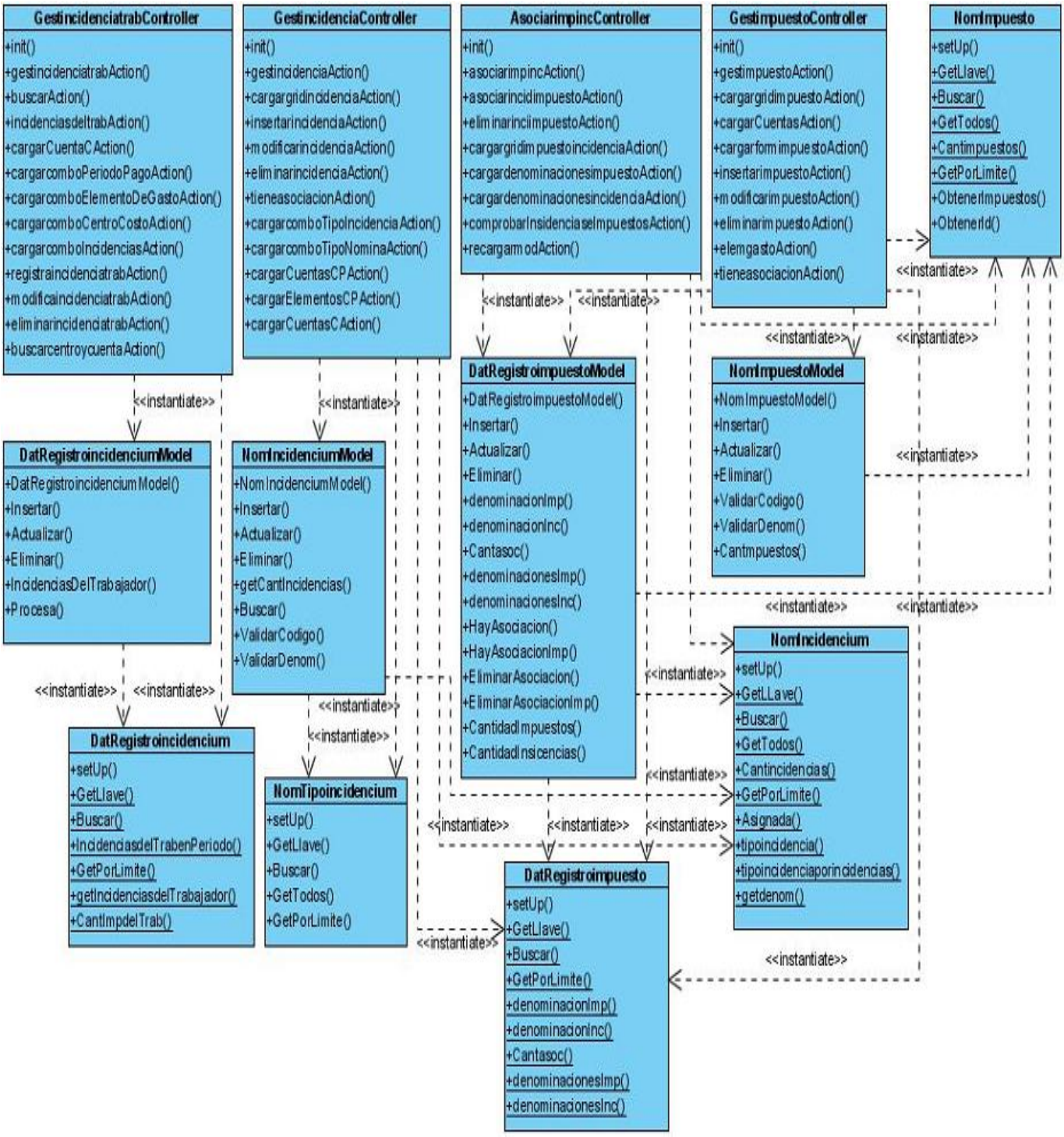


Figura 15 Diagrama de clases del diseño del componente Incidencia.

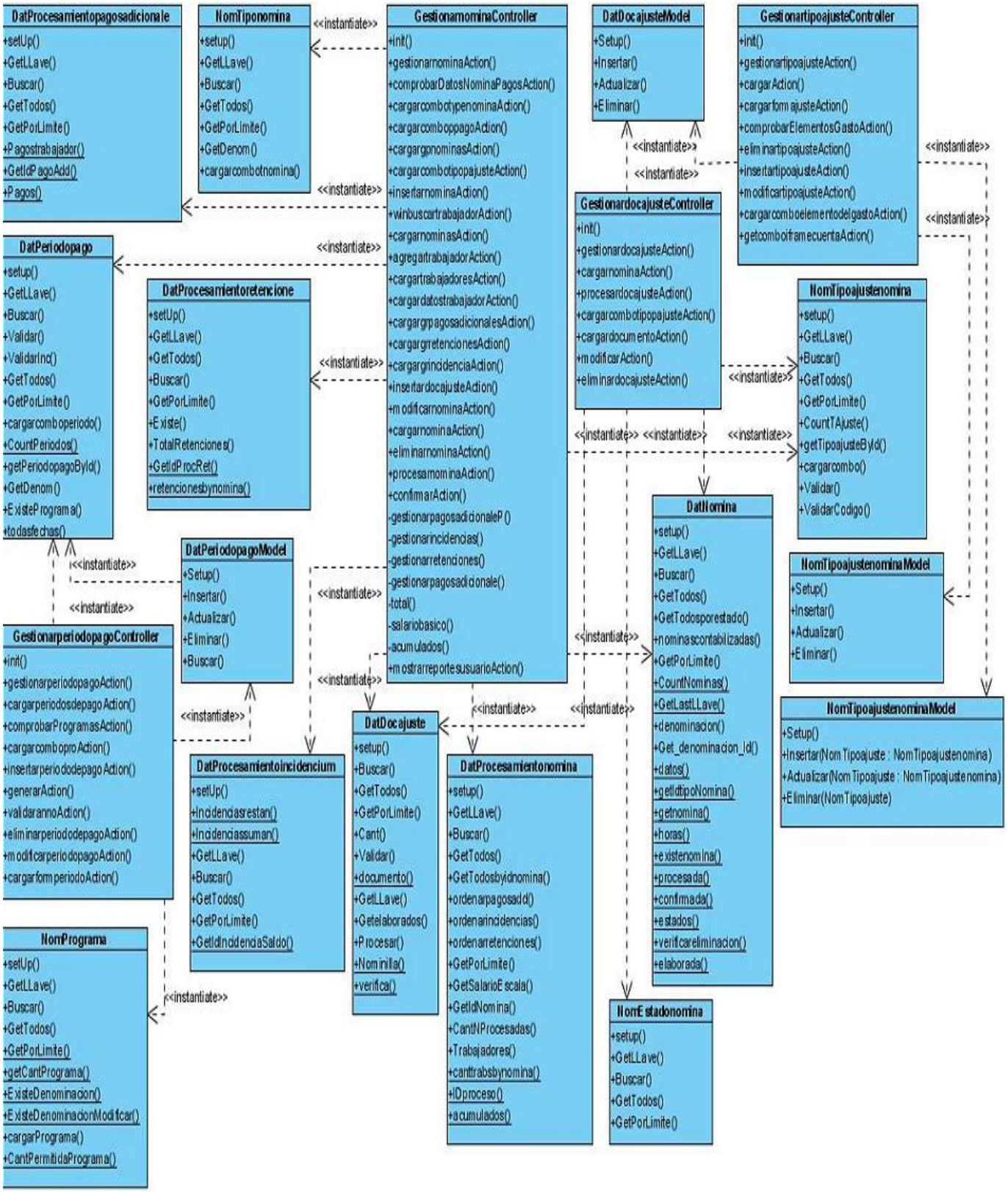


Figura 16 Diagrama de clases del diseño del componente Nómina.

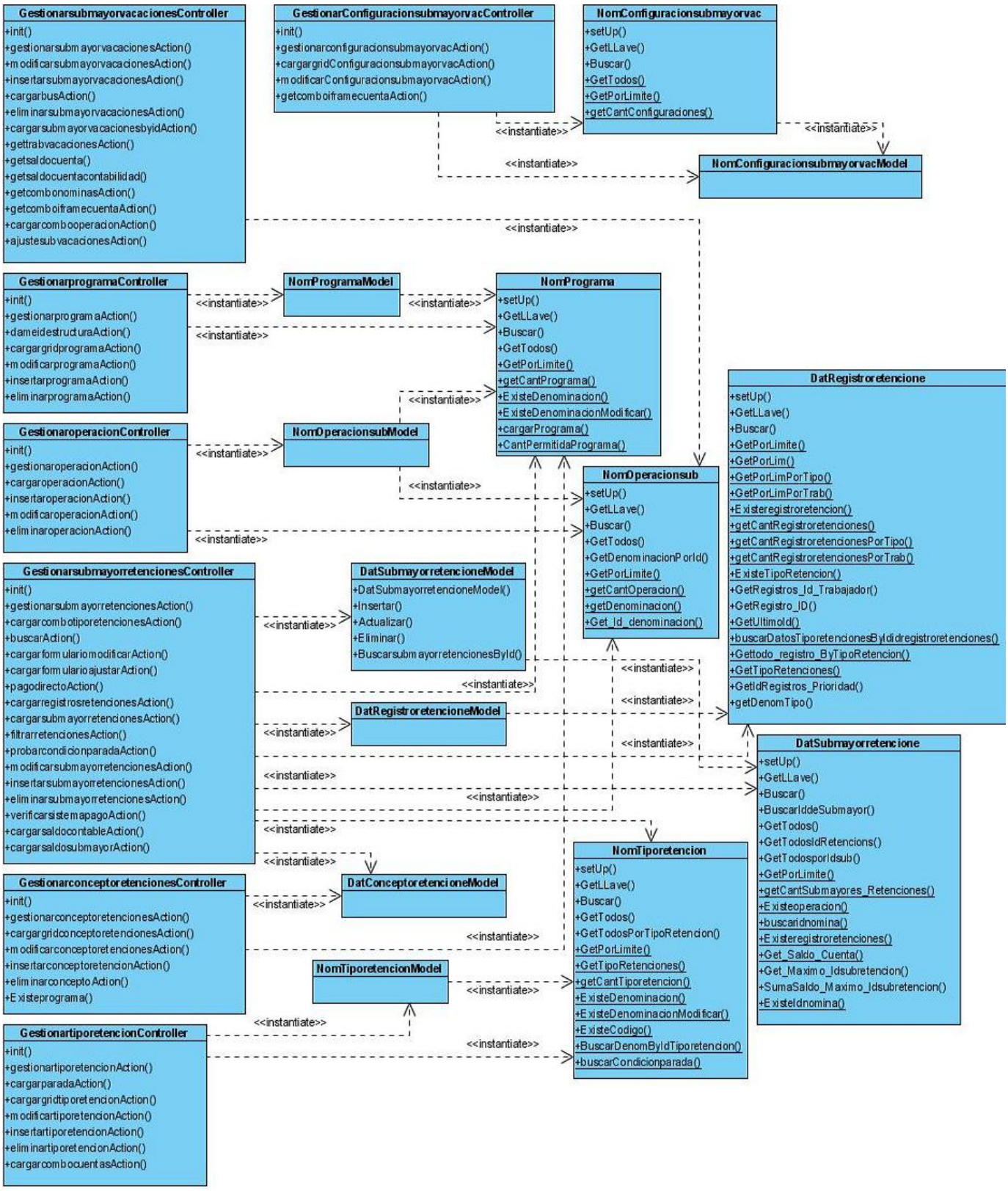


Figura 17 Diagrama de clases del diseño del componente Submayores.

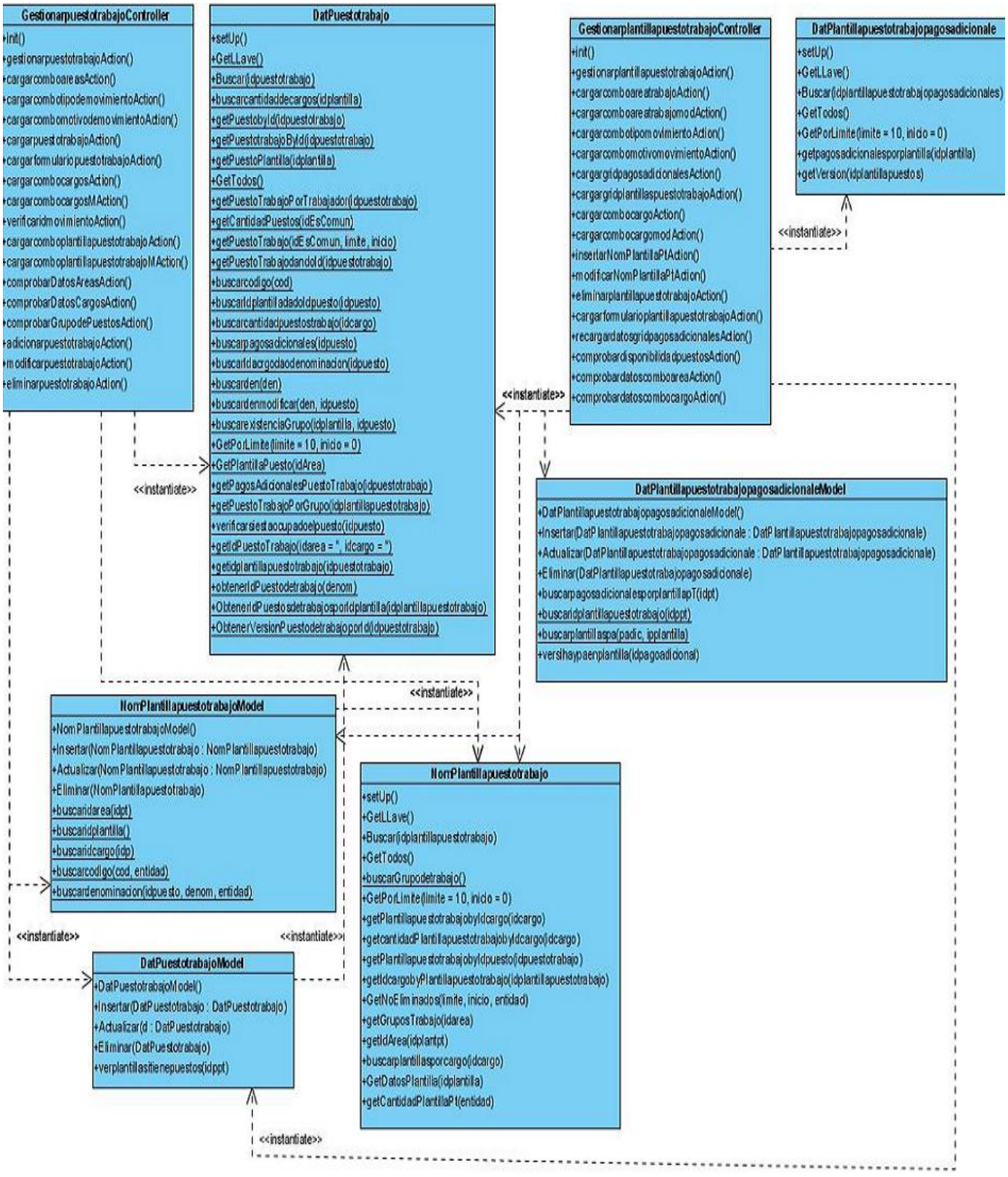


Figura 18 Diagrama de clases del diseño del componente Puesto de Trabajo.

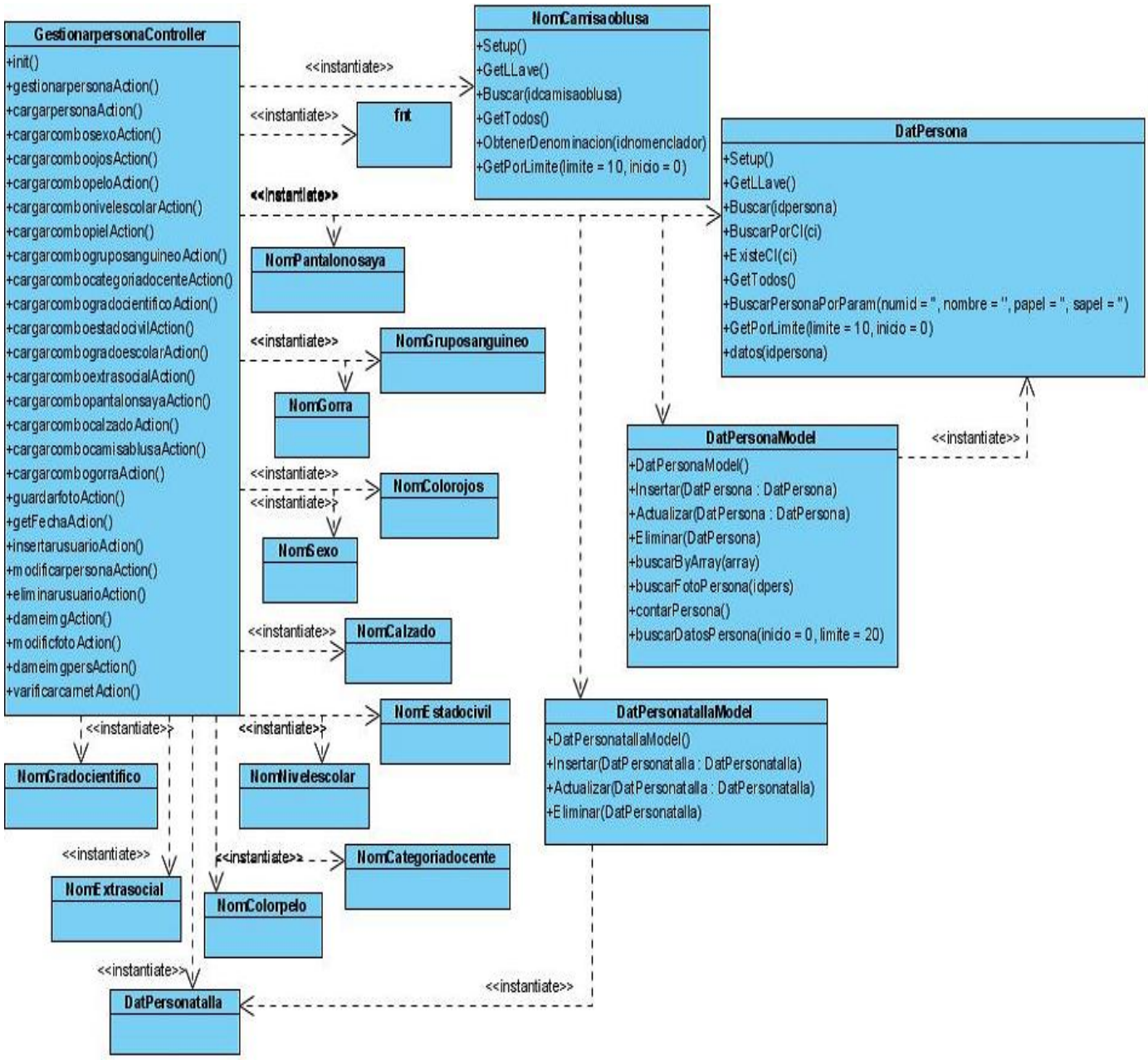


Figura 11 Diagrama de clases del diseño del componente Persona.

2.13 Integración Horizontal

Los ocho componentes principales por los cuales está compuesto el subsistema Capital Humano se integran horizontalmente en cada una de las capas: Presentación, Negocio y Datos. En la capa de Presentación se integran mediante el Portal UCID, en la de negocio la integración es mediante el framework IoC. Y en el caso de la capa de Datos la integración se lleva a cabo a través de sus respectivas interfaces.

2.14 Integración Vertical

La arquitectura usa la AOP, o sea, Programación Orientada a Aspectos, la cual ayuda a modificar, dinámicamente, el modelo estático, pues incluye el código requerido para cumplir los requerimientos secundarios sin tener que modificar el modelo estático original.

Algunos de los aspectos tratados son:

- Validación
- Excepciones
- Transacciones
- Reglas
- Trazas
- Concurrencia
- Caché
- Conceptos Globales
- Seguridad

La integración vertical se realiza mediante Doctrine en la capa de datos, Zend Framework y Zend EXTjs en la lógica del negocio, y EXTjs en la capa de presentación.

Validación

Se encarga de validar los campos que serán introducidos en inserciones a la base de datos, de manera que siempre se compruebe que estén correctos los mismos antes de introducirlos en las tablas. A través del mismo se disparan las excepciones del subsistema.

Excepciones

El manejador de excepciones permite realizar el tratamiento necesario cuando ocurra una excepción en el framework, evitando así que se tenga que incluir en cada clase el lanzamiento de excepciones.

Traza

Es un registro de determinadas acciones tanto de los usuarios como del sistema, la cual está compuesta por un lanzador de eventos que muestra una acción especificada al usuario y un manejador de trazas que es el encargado de registrar dichos historiales.

Concurrencia

Asegura que si hay varios usuarios haciendo peticiones a la vez en una interfaz que maneje las mismas tuplas de la base de datos, no ocurra que alguno de ellos haga

algún cambio sin tener bien actualizada la información. Esto lo realiza a través de un campo llamado versión que se incrementa cada vez que algún usuario hace uso de la interfaz, así cada vez que la interfaz hace la petición a la base de datos esta verifica la última versión que tiene con la que viene en el nuevo campo versión y si son diferentes el sistema envía un mensaje pidiendo que se actualicen los datos antes de hacer algún cambio.

Transacciones

Componente con el que cuenta el framework y mediante el cual serán salvados automáticamente los datos de modificaciones e inserciones. El mismo realizará la apertura, cierre y marcha atrás de las transacciones. Favorece el hecho de que las funcionalidades de modificación no necesiten crear instancias de la clase modelo ni ejecuten métodos en la misma, sino que solamente extraigan los datos de los campos de la presentación y con esto ya el framework hace la salva de dichos datos en la base de datos.

Esto le facilita al programador la implementación, puesto que anula la necesidad de crear consultas de inserción, producto a esto solo se debe programar la obtención de los campos de la presentación y el framework será el responsable de que los mismos sean guardados en la base de datos. Este componente es un aspecto del sistema y en su configuración se realiza un registro como aspecto para especificar si se quiere activar o no la salva automática.

2.15 Integración Sistema y Datos

En el subsistema los componentes se relacionan con diferentes tablas del modelo de datos, sobre las cuales van a interactuar y realizar diferentes transacciones. A continuación se muestra una tabla donde se encuentran, por componentes, las tablas con que interactúa y las principales transacciones que se realizan:

Tabla 8 Principales transacciones por componente.

Componente	**Tabla con la que interactúa**	**Principales transacciones realizadas**
Persona	dat_persona	Adicionar, Modificar,

		Eliminar
	dat_personatalla	Adicionar, Modificar, Eliminar
Puesto de Trabajo	dat_puestotrabajo	Adicionar, Modificar, Eliminar
	nom_plantillapuestotrabajo	Adicionar, Modificar, Eliminar
	dat_plantillapuestotrabajopagosadicionales	Adicionar, Modificar, Eliminar
	his_histplantillapuestotrabajopagosadicionales	Adicionar
Pagos Adicionales	nom_pagosadicionales	Adicionar, Modificar, Eliminar
	his_histpagosadicionales	Adicionar
Incidencias	dat_registroincidencias	Adicionar, Modificar, Eliminar
	nom_impuesto	Adicionar, Modificar, Eliminar
	nom_incidencias	Adicionar, Modificar, Eliminar
	dat_registroimpuesto	Adicionar, Modificar, Eliminar
Trabajador	dat_trabajador	Adicionar, Modificar, Eliminar
	dat_modelomovnomina	Adicionar, Modificar,

		Eliminar
	nom_motivomovnomina	Adicionar, Modificar, Eliminar
	dat_datoscontables	Adicionar, Modificar, Eliminar
	dat_registropagosadicionalestrabajador	Adicionar, Modificar, Eliminar
	his_histregistropagosadicionalestrabajador	Adicionar
Submayores	nom_tiporetencion	Adicionar, Modificar, Eliminar
	nom_programa	Adicionar, Modificar, Eliminar
	dat_conceptoretenciones	Adicionar, Modificar, Eliminar
	dat_submayorretenciones	Adicionar, Modificar, Eliminar
	dat_registroretenciones	Adicionar, Modificar, Eliminar
	his_histregistroretenciones	Adicionar
	dat_submayorvac	Adicionar, Modificar, Eliminar
	his_histsubmayorvac	Adicionar
	nom_configuracionsubmayorvac	Adicionar, Modificar, Eliminar

Nómina	dat_períodopago	Adicionar, Modificar, Eliminar
	dat_comprobanteoperaciones	Adicionar, Modificar, Eliminar
	dat_nomina	Adicionar, Modificar, Eliminar
	dat_docajuste	Adicionar, Modificar, Eliminar
	dat_procesamientonomina	Adicionar, Modificar, Eliminar
	nom_tipoajustenomina	Adicionar, Modificar, Eliminar
	dat_registroimpuestotipoajuste	Adicionar, Modificar, Eliminar
	dat_procesamientopagosadicionales	Adicionar, Modificar, Eliminar
	dat_procesamientoincidencia	Adicionar, Modificar, Eliminar
	dat_procesamientoretenciones	Adicionar, Modificar, Eliminar

2.16 Matriz de Integración

Tabla 9 Integración interna.

Entradas/Salid	Persona	Trabajador	Puesto de Tral	Pagos Adi	Incidencias	Submayores	Nómina	Cierre
Persona								
Trabajador	datpersona		DatPuestotrab NomPlantillap	NomPagos	NomIncide			
Puesto de Trab	datpersona	DatModelom NomMotivo NomTipomo		NomPagos				
Pagos Adiciona								
Incidencias							NomTiponom	
Submayores	datpersona	DatTrabajad					DatNomina	
Nómina	datpersona	DatTrabajad	DatPuestotrab NomPlantillap	NomPagos	DatRegistro	NomProgram DatRegistrore DatSubmayor DatSubmayor DatSubmayor		
Cierre							DatNomina	

2.17 Entradas y salidas de los componentes del subsistema

En la siguiente tabla se muestran las entradas y salidas por proceso, luego de hacer el análisis del diagrama de componentes.

Tabla 10 Entradas y salidas asociadas a componentes.

Componente	Entra	Sale
Pagos Adicionales	Elemento de Gasto	Pagos Adicionales
Puesto de Trabajo	Cargo, Área, Entidad	Puesto de Trabajo
Persona		Persona
Trabajador	Centro de Costo, Puesto de Trabajo, Pagos Adicionales	Trabajador, Movimiento de Nómina
Incidencias	Trabajador, Tipo de Nómina, Período de Pago, Cuentas, Elemento de	Registro de Incidencias

	Gasto	
Nómina	Provisión de Vacaciones, Registro de Retenciones, Movimiento de Nómina, Trabajador, Pago Adicional, Puesto de Trabajo, Registro de Incidencias, Cuentas, Elemento de Gasto	Retenciones Descontadas, Comprobante de Operaciones, Nóminas, Acumulado de Vacaciones
Submayores	Trabajador, Acumulado de Vacaciones, Retenciones Descontadas, Cuentas	Registro de Retenciones, Provisión de Vacaciones
Cierre	Nómina	

2.18 Soluciones específicas

Dentro de los componentes más complejos en la línea se encuentran: Trabajador, Nómina y Submayores.

El componente Trabajador permite mantener actualizado el registro de trabajadores y la actualización de la Plantilla de personal. Los movimientos de la Fuerza de trabajo pueden clasificarse en tres subprocesos fundamentales: Altas, Reubicaciones y Bajas.

El proceso de alta se resume en la selección de la persona a la que se desea realizar la formalización de la relación laboral y la asociación a ella de un puesto de trabajo que se encuentre vacante, también se registran los pagos adicionales asociados a dicha persona y un grupo de datos laborales necesarios para la elaboración del contrato de trabajo y el procesamiento de las nóminas.

Una vez formalizada la relación laboral del trabajador, el mismo puede ser objeto de movimientos de Reubicaciones o Bajas en dependencia de estudios organizacionales realizados en la entidad, o de aplicación de medidas disciplinarias, o cualquier otra causa que implique variaciones en los elementos de su salario o cambios de puestos de

trabajo o cargos. Los movimientos de reubicaciones pueden ser permanentes o temporales y en los dos casos deben especificarse las causas que motivaron dichos movimientos.

Una condición obligatoria para efectuar un movimiento de baja es que la persona haya sido registrada como trabajador, estos tipos de movimientos se producen por cualquiera de las causas especificadas en la legislación laboral vigente para este caso.

La Gestión del submayor de vacaciones se inicia con el proceso de su apertura para cada trabajador de nuevo ingreso a la Entidad laboral, en la gran mayoría de los casos el tiempo y los importes acumulados de las vacaciones son nulos, aunque en situaciones excepcionales se autoriza por la legislación vigente la transferencia de los días e importes acumulados de los trabajadores y que no fueron liquidados en sus Entidades de procedencia. Una vez que el trabajador causa baja de la entidad laboral, independientemente, de la causa que la genere tiene derecho a la liquidación de todos los importes acumulados hasta esa fecha y debe procederse al cierre del submayor de vacaciones.

La apertura del submayor de retenciones tiene lugar cuando los trabajadores vinculados laboralmente a un centro de trabajo contraen obligaciones de adeudos y son presentadas las notificaciones y documentos primarios que autorizan los mismos. La apertura de este submayor consiste en el registro para cada trabajador y cada tipo de retención de los saldos totales de la deuda, los importes a descontar en cada plazo y la cantidad de plazos en que se le realizarán los descuentos. A cada tipo de retención se le asigna un orden de prioridad en el que se le irán descontando los importes del salario.

Los submayores tanto el de vacaciones como el de retenciones tienen que estar actualizados y sus saldos deben corresponderse con las diferentes nóminas procesadas.

Una nómina se define como una lista conformada por el conjunto de trabajadores de una entidad determinada a los cuales se les va a remunerar por los servicios prestados. Es el instrumento que permite de una manera ordenada, realizar el pago de salarios a los trabajadores, así como proporcionar información contable y estadística, tanto para la empresa como para el organismo encargado de regular las relaciones laborales. Aunque en cada entidad puede variar la forma de calcular y contabilizar la nómina,

existen ciertos pasos comunes a todas, como la preparación de la nómina con los nombres y las remuneraciones de los trabajadores.

Uno de los procesos que se efectúa en el departamento de Capital Humano es el pago de los trabajadores en activo existentes en la entidad mediante la confección y cálculo de la nómina. Dentro del mismo se desarrollan los subprocesos siguientes:

- Apertura de nómina
- Procesar Nómina
- Cierre de Período.

Se realiza basado en el resumen de incidencia de cada trabajador en el período de pago a analizar. Inicialmente se crean el Mayor de trabajadores, los Submayores correspondientes (submayor de vacaciones y submayor de retenciones), el Fichero histórico devengado y el Registro de salario y tiempo de servicio. Luego se procede a realizar las nóminas correspondientes a Salarios, Subsidios, Vacaciones, entre otros, en el período que se establezca, con las informaciones obtenidas de la prenómina que contienen las incidencias procesadas por el área de personal; las retenciones de los trabajadores utilizando las notificaciones bancarias y los movimientos de nómina que permiten actualizar el maestro de trabajadores.

Cuando se realiza la nómina se tiene en cuenta si el trabajador tiene alguna retención al verificarlo con el submayor de retenciones, en caso de tener: se le descuenta y se actualiza este submayor al terminar el procesamiento de la nómina; si la nómina es de vacaciones se verifica la duración de las vacaciones que tomará el trabajador para sacar, proporcionalmente, el importe al tiempo, de acuerdo a lo que tiene acumulado en el submayor de vacaciones y esta cantidad es lo que se le pagará, actualizando este submayor al concluir el procesamiento. Si la nómina es de subsidio se debe tomar del fichero histórico devengado el acumulado del tiempo trabajado en los meses anteriores para el cálculo de la misma.

Al terminar se actualizan los submayores el de vacaciones con la provisión de vacaciones acumuladas en ese período y el de retenciones con las respectivas deducciones descontadas al trabajador; además de los submayores se actualiza el registro de salario y tiempo de servicio y se obtiene la distribución de moneda para el pedido del efectivo al banco.

La información de las nóminas se enviará mediante el Comprobante de Operaciones al subsistema de Contabilidad con el propósito de realizar la contabilización de las mismas y se obtendrá además la distribución de moneda para la solicitud del efectivo al banco.

2.19 Plan de Iteración

El plan de iteración organizado para la creación del subsistema Capital Humano estuvo sujeto a las restricciones de tiempo y a las exigencias en cuanto a estándares y calidad impuestas por el sistema de gestión integral al que pertenece. Toma como base la metodología de desarrollo implementada en el proyecto para, asignar las tareas conforme a las actividades predefinidas para cada rol y ordenarlas por fases específicas, además apoyado en la metodología se controlan los artefactos que deben ser generados a lo largo del ciclo de vida del software. Las dependencias originadas por los componentes, así como su nivel de complejidad fueron otros de los aspectos elementales a valorar para planificar el camino a seguir en esta iteración del desarrollo del producto.

Fue definido que una vez garantizado que todas las tareas predecesoras a las exclusivas del subsistema Capital Humano hayan sido concluidas, se comenzará la construcción de los componentes. De modo que una condición imprescindible para la realización de cada tarea era la finalización de todas las predecesoras.

El documento trae anexado imágenes del plan de iteración llevado a cabo en la línea Capital Humano, donde se pueden observar las horas dedicadas a trabajar tanto en el módulo, como en cada uno de sus componentes, el personal encargado del óptimo cumplimiento de la tarea, así como el porciento de la actividad cumplido, la duración de la misma y sus respectivas fechas de comienzo y culminación. Anexo 4 *Plan de Iteración.*.

2.20 Estructura de los componentes

Para lograr que un componente utilice un servicio de otro componente se creó un paquete Servicios que incluirá todas las clases y funcionalidades que están en los paquetes Controladora, Modelo y Validaciones. Cuando se desee hacer una petición a otro componente esta se hará a través del paquete Servicios, el cual analizará la

solicitud e irá a la clase que tiene implementada dicha funcionalidad y la entregará a Servicios que a su vez se la entregará a quien hizo la solicitud.

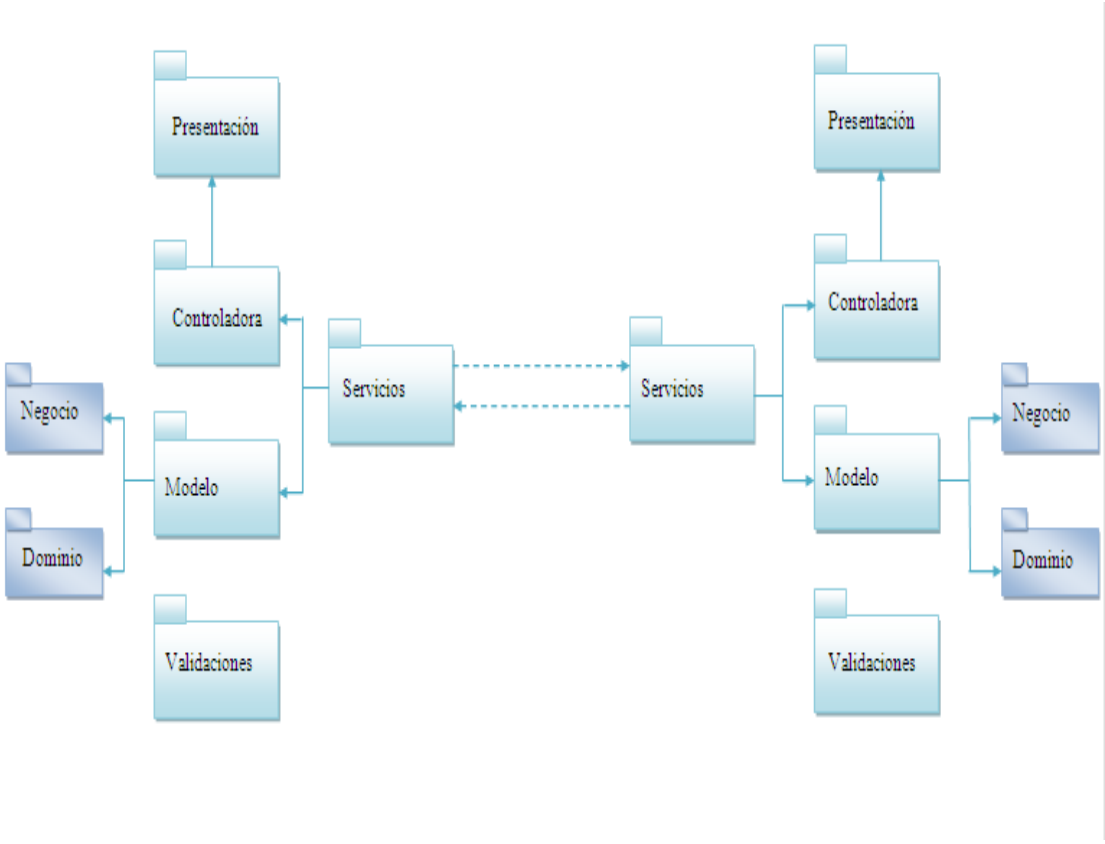


Figura 2012 Esquema estructural de componente.

2.21 Colaboración entre clases

Un usuario hace una solicitud mediante un evento del navegador en un formulario JS, este envía una notificación a la clase controladora, mediante JSON o Atributos. Esta setea entidades y se comunica con la clase gestora Business responsable de la petición, o se comunica con la Doctrine domain, entidad mapeada por el ORM Doctrine. La gestora o Business es la encargada de la lógica propia del negocio, cálculos y procesamiento de negocio, y también puede comunicarse con la Doctrine domain. Esta última se conecta mediante pdo (Doctrine Record) a la DBO (Capa interna de Doctrine basada en PDO nativo), y esta DBO es la que se conecta directamente a la base de datos.

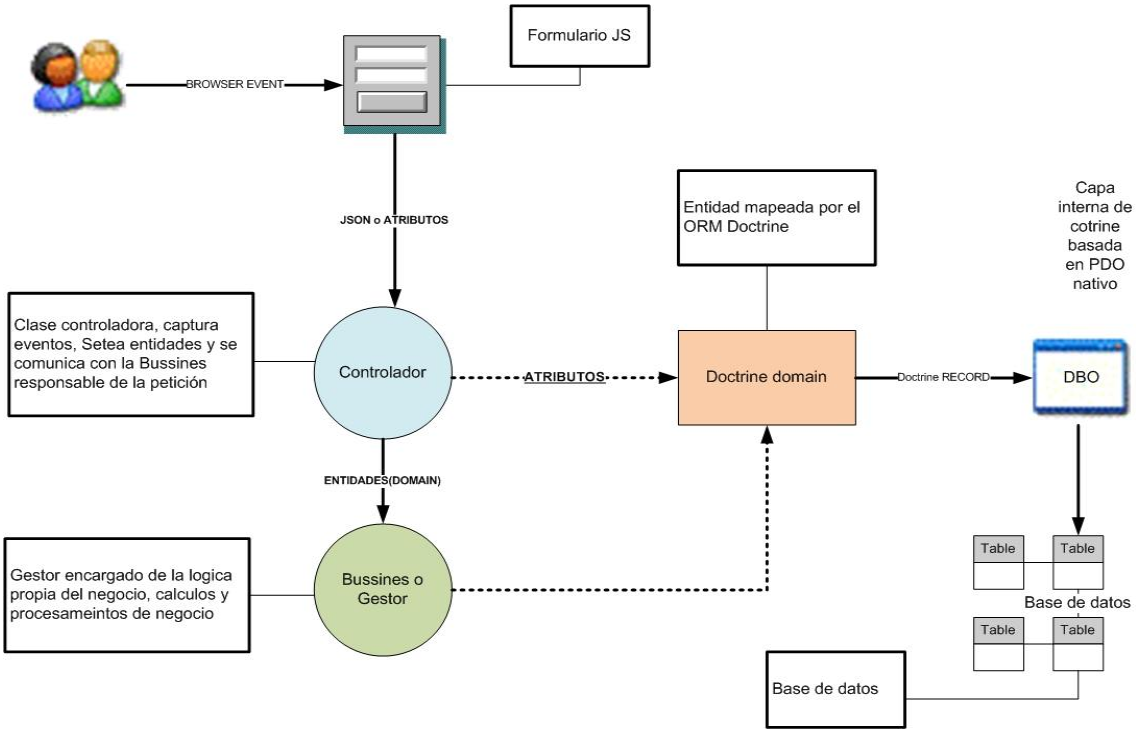


Figura 2113 Colaboración entre clases.

A lo largo de este capítulo se puso de manifiesto que la arquitectura del negocio constituyó un paso de inicio importante en el desarrollo de la arquitectura de sistema del Subsistema Capital Humano. A través del análisis de los conceptos más importantes y sus principales relaciones, se pudo tener una idea inicial de cómo deberían interactuar los componentes para cumplir con las funcionalidades establecidas para dicho subsistema. Al concluir esta etapa de Arquitectura del Negocio se pudo tener la idea de lo que se debía definir para que los componentes pudiesen integrarse. Luego de haber realizado la formalización de los componentes y haber definido sus respectivas responsabilidades arquitectónicas se tuvo una vista global de cómo iba a funcionar el subsistema, qué módulos se iban a encargar de cada funcionalidad, la forma en que iban a asumir dicha responsabilidad y otros aspectos.

La aplicación de patrones estudiados en el primer capítulo ayudó a los componentes implementados a integrarse de una forma más fácil, cumpliendo con los requerimientos capturados en el negocio, contribuyendo a sentar las bases de una arquitectura sólida y además, fijando pautas para predecir el funcionamiento de componentes que pudiesen insertarse en futuras iteraciones.

El proceso de priorización de los componentes y la realización del plan de iteración, este último en conjunto con el Jefe de Línea, ayudaron a organizar el trabajo de implementación de los componentes y a planificar las tareas para dar cumplimiento a las funcionalidades previstas, según los recursos con los que contó la línea durante el proceso de desarrollo del subsistema.

Todos estos factores favorecieron que haya sido implementada la arquitectura del subsistema en tiempo real, siendo la Línea Capital Humano una de las primeras en hacer entregas a la Línea Calidad, por lo que fue reconocido el trabajo del equipo en general.

Capítulo III: Evaluación de la Arquitectura

Al concluir la definición de la arquitectura, es recomendable realizar valoraciones para tener la medida en que la solución cumple con parámetros de calidad que aseguren que lo que se construyó va a contribuir a un desarrollo de software viable, sin muchos problemas y soportado por una arquitectura robusta. Con este propósito, se llevo a cabo un proceso de evaluación de la arquitectura. El mismo refleja la importancia que esta tiene para el proceso de desarrollo del sistema. La alternativa seleccionada para evaluar la propuesta arquitectónica fue la aplicación del método ATAM[18].

3.2 Necesidad de evaluar una arquitectura

La arquitectura es el primer artefacto del ciclo de vida del desarrollo de un software, que incorpora importantes decisiones de diseño. Estas decisiones son fáciles de tomar, pero difíciles de cambiar una vez que el sistema se aplica.

Todos los diseños arquitectónicos implican desventajas en las cualidades del sistema, ya que, estas dependen en gran medida de las decisiones arquitectónicas; por lo que garantizar la calidad del sistema es a menudo a expensas de garantizar calidad en la arquitectura y esto es posible si se realiza una evaluación de la misma. (44)

3.3 ¿Cuándo hacer la evaluación de la arquitectura?

Por lo general la evaluación de la arquitectura ocurre después que ésta ha sido especificada, pero antes que empiece la implementación; esto se hace con el objetivo de validar el diseño propuesto. No obstante, uno de los aspectos más interesantes de la evaluación de arquitecturas es que se puede efectuar en cualquier etapa de la vida de una arquitectura.

Existen dos tipos de evaluaciones: la evaluación temprana y la evaluación tardía. En la primera no tiene por qué estar especificada completamente la arquitectura, es utilizada para examinar las decisiones arquitectónicas ya tomadas y decidir entre las opciones que están pendientes. La tardía es realizada tanto cuando la arquitectura está terminada, como cuando la implementación está completa.

La evaluación de la arquitectura de software debe realizarse cuando esta contiene suficientes elementos como para justificarla. Un buen momento para determinar cuándo realizar la evaluación es cuando el equipo de desarrollo comienza a tomar decisiones

[18] Architecture Trade-off Analysis Method

que dependen de la arquitectura y que el costo de no tenerlas en cuenta son mayores que el costo de realizar una evaluación.

(45)

3.4 Técnicas de evaluación

Existen dos tipos de mediciones: la medición cualitativa y la medición cuantitativa.La medición cualitativa se aplica para la comparación entre arquitecturas candidatas y tiene relación con la intención de saber la opción que se adapta mejor a cierto atributo de calidad. Este tipo de medición brinda respuestas afirmativas o negativas, sin mayor nivel de detalle. La medición cuantitativa busca la obtención de valores que permitan tomar decisiones en cuanto a los atributos de calidad de una arquitectura de software. El esquema general es la comparación con márgenes establecidos, como lo es el caso de los requerimientos de desempeño, para establecer el grado de cumplimiento de una arquitectura candidata, o tomar decisiones sobre ella. Este enfoque permite establecer comparaciones, pero se ve limitado en tanto no se conozcan los valores teóricos máximos y mínimos de las mediciones con las que se realiza la comparación.

Por lo regular, las técnicas de evaluación cualitativas son usadas cuando la arquitectura se encuentra en construcción, mientras que las técnicas de evaluación cuantitativas, se usan cuando la arquitectura ya ha sido implantada.

En el tipo de medición cualitativa se pueden utilizar las técnicas basadas en escenarios o en experiencia. Mientras que en el tipo de medición cuantitativas se pueden emplear simulaciones o modelos matemáticos.

(45)

3.5 Evaluación de la arquitectura a través del método ATAM

La evaluación se realizará atendiendo a cómo responden algunos elementos ante los principales atributos de calidad. Para dicha evaluación, por orientación de la línea Arquitectura se hizo uso del método ATAM, el cual se aborda más detalladamente a continuación.

3.5.1 Características de ATAM

Permite determinar el impacto de los atributos de calidad en la arquitectura, además provee un entendimiento interno de cómo interactúan los principales objetivos de calidad. El objetivo de la evaluación usando ATAM es entender las consecuencias de las decisiones arquitecturales que respectan a los requerimientos de los atributos de

calidad del sistema. Además permite comprobar si se pueden alcanzar los objetivos concebidos mediante la propuesta de arquitectura.

Los puntos sensibles son parámetros en la arquitectura en los cuales la respuesta medible de algunos atributos de calidad son altamente correlacionados (ej. se puede determinar que la cantidad total de datos trasmitidos en un sistema es altamente correlacional con la cantidad de datos que se transmiten por un canal de comunicación en especifico).

Un punto de intercambio (tradeoff) es descubierto en la arquitectura cuando un parámetro de construcción arquitectural es un anfitrión para más de un punto sensible, donde los puntos de calidad medibles son afectados, indistintamente, por cambios en el parámetro, (ej. si se aumenta la velocidad en el canal de comunicación mencionado anteriormente esto mejoraría en flujo de datos pero reduciría la confiabilidad, entonces la velocidad del canal es un punto de intercambio (tradeoff).

(44)

3.5.2 Propósito

Lo que se pretende hacer con ATAM, además de mejorar la documentación, es registrar los posibles *riesgos*, los *no riesgos*, los *puntos de sensibilidad* y los *puntos de desventajas* que se encuentran en el análisis de la arquitectura.

Riesgos: las decisiones de arquitectura que podrían crear problemas en el futuro para algunos atributos de calidad.

No riesgos: las decisiones arquitectónicas que sean adecuadas al atributo de calidad que afectan.

Desventaja: las decisiones de arquitectura que tienen un efecto en más de un atributo de calidad.

Puntos de sensibilidad: una propiedad de uno o más componentes, y/o las relaciones entre componentes, fundamental para el logro de un determinado requisito de atributo de calidad.

ATAM consiste en los siguientes nueve pasos:

1. *Presentar ATAM*: El equipo de evaluación presenta un panorama general de los pasos de ATAM, las técnicas utilizadas y de los resultados del proceso.

2. *Presentar el negocio*: El líder presenta brevemente el negocio y el contexto de la arquitectura.

3. *Presentar la arquitectura*: El arquitecto presenta un panorama de la arquitectura.

4. *Identificar los enfoques arquitectónicos*: El equipo de evaluación y el arquitecto detallar los planteamientos arquitectónicos descubiertos en el paso anterior.

5. *Generar el árbol de utilidad de atributos de calidad*: Un pequeño grupo de orientación técnica interesados identifica, prioriza, y refina los atributos de calidad más importantes en un formato de árbol de utilidad.

6. *Analizar los enfoques arquitectónicos*: El equipo de evaluación revisa los enfoques arquitectónicos sobre los atributos de calidad para identificar los riesgos, los no riesgos y las desventajas. Se utilizan las preguntas similares a las presentadas en el paso 5.

7. *Lluvia de ideas y dar prioridad a los escenarios*: un mayor y más diverso grupo de interesados crea escenarios que representan a sus diferentes intereses.

8. *Analizar los enfoques arquitectónicos*: El equipo de evaluación continúa detectando los riesgos, los no riesgos, y observando al mismo tiempo las ventajas de cada uno de los escenarios de impacto en la arquitectura.

9. *Presentar los resultados*: El equipo de evaluación recapitula los pasos de ATAM, los resultados, y recomendaciones.

(45)

3.6 Presentación del negocio.

Debido a que la línea se encuentra dividida en 3 módulos, se presentan a continuación los mapas de procesos de cada módulo junto con la descripción de los procesos involucrados en cada módulo.

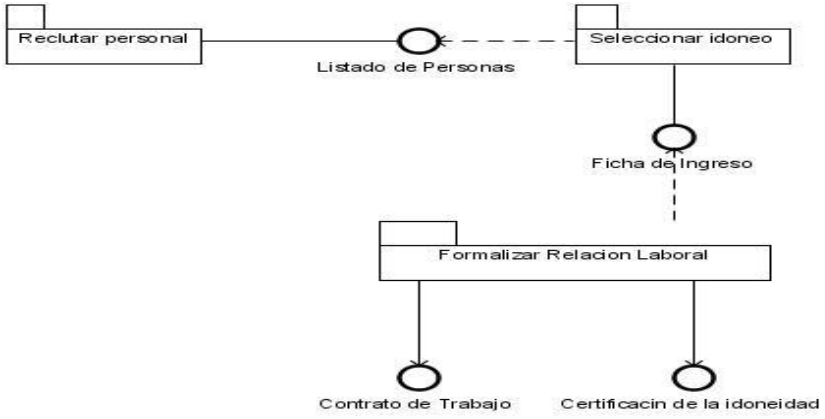


Figura 2214 Mapa de Procesos de Negocio del proceso Selección e Integración.

Descripción de Procesos de Negocio:

- Reclutar Personal.

En el proceso de *Reclutar Personal* se tienen en cuenta la prioridad de las posibles fuentes de reclutamiento que pueden ser: fuentes internas de la Entidad, otras entidades de la organización superior de dirección o del Organismo, Reserva laboral de las Direcciones de Trabajo, Personas sin vínculo laboral, Personal egresado de la educación técnico profesional y universitario, que son asignados a las Entidades según lo dispuesto por el Gobierno, logrando captar a todo el personal disponible para una plaza determinada.

- Seleccionar idóneos.

En el proceso de *Seleccionar Idóneos* se analiza entre los aspirantes quienes están aptos para desempeñar las ocupaciones o cargos existentes o para su incorporación a cursos. Se realizan técnica de selección como: entrevistas de selección, pruebas de conocimientos o pruebas psicométricas, se recepcionan las propuestas y citación de los candidatos a seleccionar, se aplican métodos de selección y se realizan análisis de los aspirantes seleccionando finalmente los idóneos para una plaza determinada.

- Formalizar Relación Laboral.

El proceso de *Formalizar Relación Laboral* permite a los aspirantes seleccionados incorporarse al colectivo laboral mediante la realización del contrato de trabajo, ambientarse en la labor a desarrollar y adquirir el dominio del trabajo y las habilidades para ejecutarlo adecuadamente.

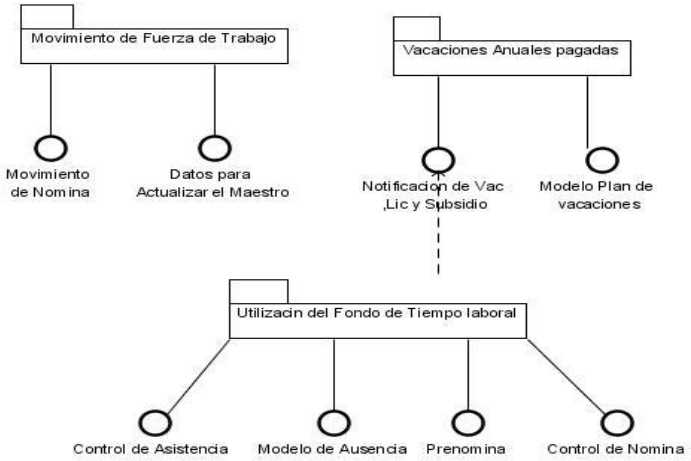


Figura 23 Mapa de procesos de negocio de Administración de Capital Humano.

Descripción de Procesos de Negocio:

- Movimiento de Fuerza de Trabajo.

El proceso de *Movimiento de Fuerza de Trabajo* es el que tiene que ver con todo el movimiento de personal. Los movimientos de Fuerza de trabajo se agrupan en movimientos de altas, reubicaciones y movimientos de bajas. Estos movimientos se oficializan a través de un documento (Mov de nómina) que se utilizan para informar todo el movimiento de personal que se realice y produzca modificaciones en su estructura salarial, cargo o área de trabajo, constituyendo el documento que respalda las anotaciones para mantener actualizados en Maestro de nóminas que sirve de fuente para la preparación y pago de las Nóminas.

- Vacaciones Anuales Pagadas.

El proceso de *Vacaciones Anuales Pagadas* comprende todo lo relacionado con la programación de las vacaciones, la acumulación de las vacaciones, el disfrute de las vacaciones, la posposición de las vacaciones y la liquidación de las mismas.

- Utilización del Fondo de Tiempo Laboral.

El proceso de Utilización del *Fondo de Tiempo Laboral* abarca una serie de actividades interrelacionadas entre si que permiten efectuar el pago de los trabajadores en dependencia del tiempo realmente laborado, así como realizar estudios sobre el aprovechamiento del tiempo laboral, el mismo comprende: control de asistencia, elaboración de la prenómina, análisis sobre la utilización del fondo de tiempo laboral y confección del modelo de control de ausencias y sus causales.

Mapa de Procesos

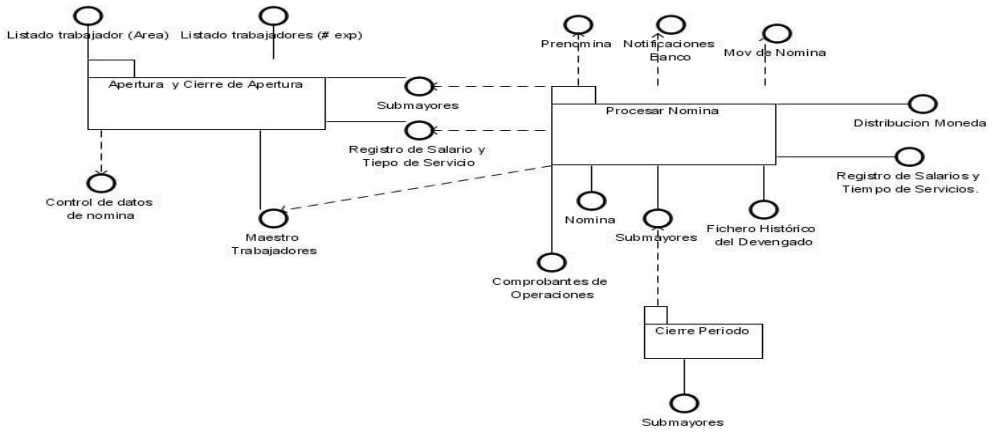


Figura 24 Mapa de Procesos de Negocio del Proceso Remuneración y Nómina.

Descripción de Procesos de Negocio:

- Apertura y Cierre de la Nómina.

En el proceso de *Apertura* es el que registra los datos de los trabajadores en activo existentes en la entidad al momento de comenzarse a trabajar con el Módulo y se crea el Maestro y los Submayores correspondientes. El *Cierre* ocurre cuando el total de las informaciones sobre los trabajadores hayan sido captadas y los saldos de los submayores y registros creados coincidan con los respectivos saldos que muestren los Submayores y el Mayor de la entidad.

- Procesar Nómina.

Este proceso comprende todo lo relacionado con le cálculo y confección de las nóminas correspondientes a Salarios, Subsidios y Vacaciones en el período que se establezca, con las informaciones obtenidas de la Prenómina con las incidencias, procesada por el área de Personal y de las retenciones de los trabajadores.

- Cierre de Período.

El proceso de *Cierre de Período* se comprueba que no quede ningún trabajador sin incorporar al Maestro y se verifica que no exista en los submayores ningún descuadre con el mayor de Contabilidad.

3.7 Descripción de la arquitectura.

El núcleo del marco de trabajo está desarrollado sobre la base de la plataforma LAPP(Linux-Apache-PostgreSQL-PHP), utilizando la versión de PHP 5.2.4 con la utilización de los frameworks; ZendFramework, Doctrine, ExtJS y ezComponent agregándole una extensión al ZendFramework (ZendExt) para darle solución a necesidades propias del negocio a desarrollar y del proceso del mismo, resolver potencialidades funcionales las cuales no se les pueden dar solución con los frameworks seleccionados, e incrementar la adaptabilidad a distintas núcleos de desarrollo incorporando un estilo híbrido basado en Capas y MVC.

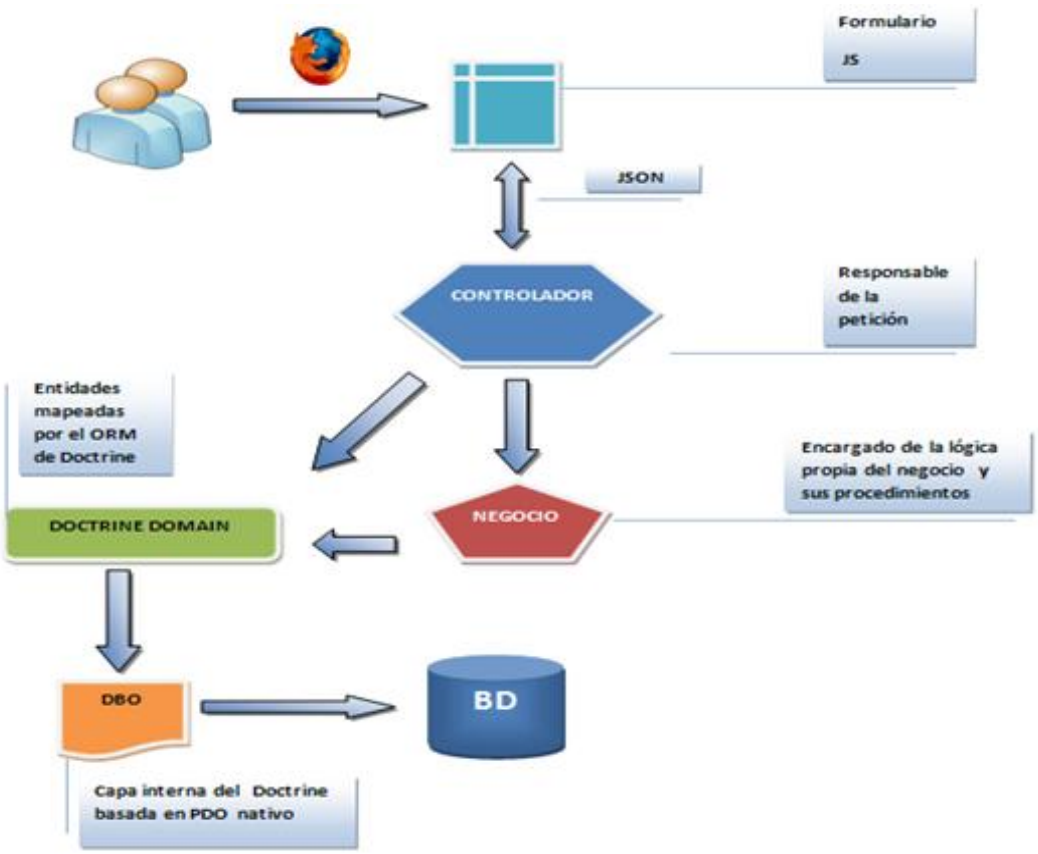


Figura 25 Estilo arquitectónico del marco de trabajo.

En la ilustración se presenta la interacción del núcleo del marco de trabajo, donde los aspectos arquitectónicos son respaldados por los componentes verticales del marco respondiendo además con las potencialidades funcionales de sistema como; *Caché, Conceptos globales, Validación, Trazas, Manejador de Excepciones,* entre otros. Se presenta un escenario simple dividido por capas (*Presentación, Negocio, Acceso a datos*), el cual interactúa con los componentes o subsistemas horizontales de la aplicación.

En este estilo o capa, para las interacciones del cliente con el servidor se define:

- ✓ Peticiones HTTP Utilizando la tecnología que brinda AJAX.
- ✓ Peticiones HTTP Utilizando el componente iframe de HTML.
- ✓ Peticiones HTTP de inclusión de archivos de código .js y .css.

Los formatos de intercambio definidos son:

JSON: para intercambio de datos entre componentes y demás elementos del servidor. Para archivos de multi-lenguaje

XML: para intercambio de datos que se almacenan archivos de configuración con formato XML.

Para el desarrollo de interfaces se trabaja en los siguientes aspectos:

- ✓ Nomenclatura para la definición de componentes.
- ✓ Definición mínima de uso para componentes del framework EXT.
- ✓ Validación de Componentes.
- ✓ Multilenguaje.
- ✓ Multitema.

En la Figura 26 se muestra otro de los aspectos desarrollados en esta capa, la integración de interfaces a partir de una lógica determinada utilizando elementos que puedan utilizarse en una interfaz correspondiente a dicha lógica, desde otra totalmente independiente que necesite del servicio. Es similar a utilizar un servicio Web, pero no a nivel de lógica sino a nivel de interfaces.

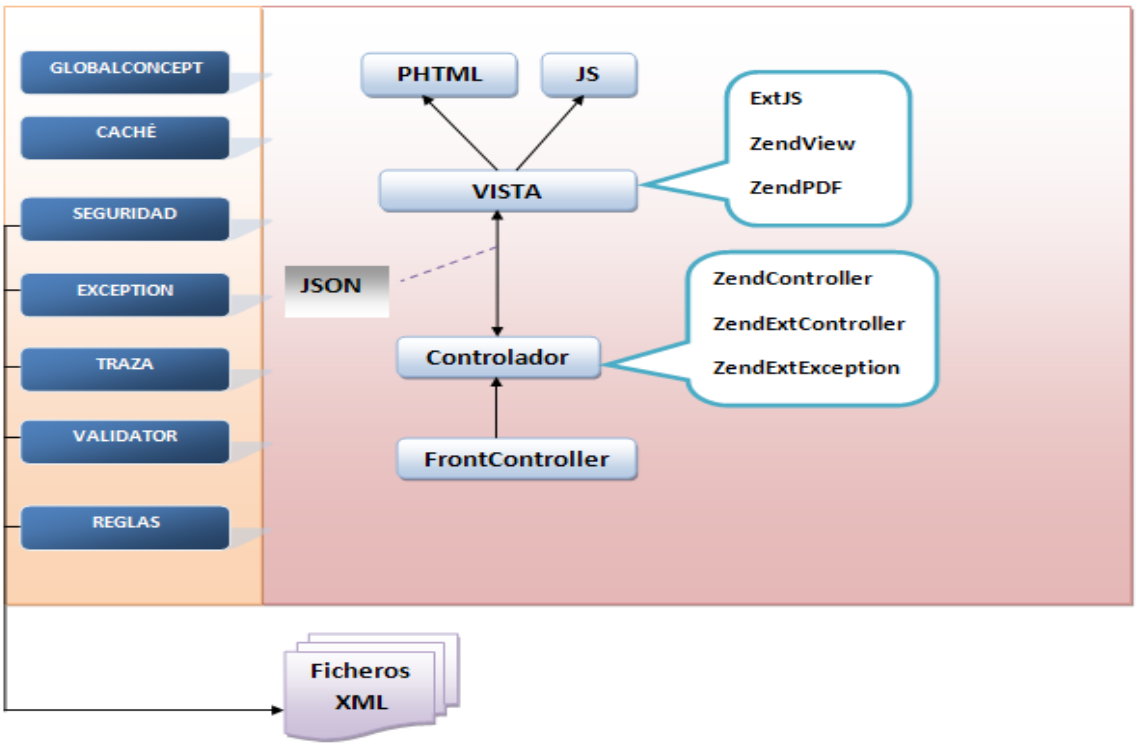


Figura 26 Escenario simple. Capa presentación.

El estilo de la Figura 27 donde el implementador define el negocio conjuntamente con sus procedimientos, se realiza instancias al dominio y se ejecutan las integraciones

a nivel de lógica de negocio. Se establecen los servicios tanto internos como externos del componente para realizar integraciones entre los distintos componentes del subsistema.

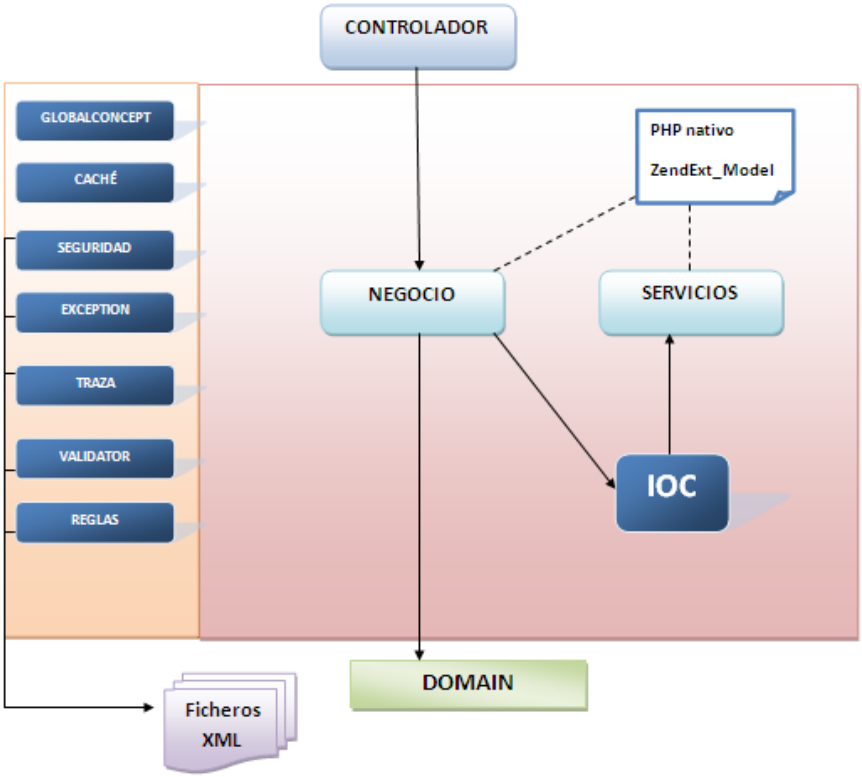


Figura 27 Escenario simple. Capa de negocio.

La Figura 28 presenta el estilo de acceso a datos Doctrine se encarga de la interacción del sistema con la base de datos mediante DBO (capa que utiliza Doctrine con PDO nativo) donde se exporta una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos.

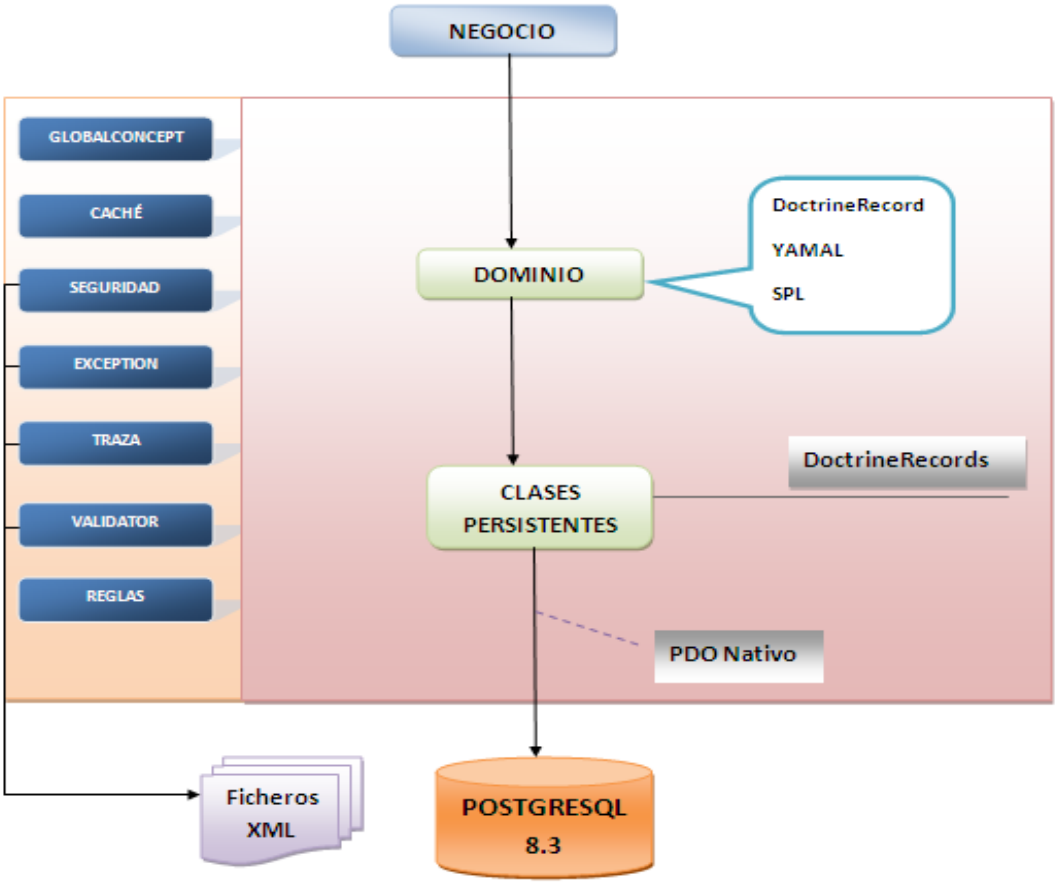


Figura 28 Escenario simple. Capa de acceso a datos.

3.8 Árbol de Utilidad

Tabla 11 Características de escenarios evaluados.

Característica	Escenario
Rendimiento	Un usuario accede a cualquier interfaz de la aplicación y esta debe estar disponible en un período de 0.1 a 0.2 segundos
	Si 100 000 usuarios concurrentes acceden a la interfaz principal de la aplicación, esta debe estar disponible para todos a la vez en un período de 0.5 a 1.0 segundos.
	Se solicita un recurso al sistema, que puede ser ligero (capacidad por debajo de 2 MB), medio (capacidad de 2 a 10 MB) o complejo (más de 10 MB de capacidad) en un servidor de 1 GB de

	memoria RAM y se recibe en un período de 0.1 a 0.7, de 0.8 a 2.0 y de 3.0 a 5.0 segundos respectivamente.
	Se intercambian datos con el sistema, ya sea mediante acciones de inserción, búsqueda o modificación, en un servidor de 1 GB de memoria RAM y se recibe la notificación de la acción realizada en un período de 0.1 a 0.7 segundos.
Mantenibilidad	Se modifican las características arquitectónicas del despliegue de la aplicación, el sistema deberá indicar las mismas cuando se realiza la próxima petición.

3.9 Análisis de los escenarios

Tabla 12 Análisis de escenarios.

Escenario:	Un usuario intercambia datos con el sistema.
Atributo:	Rendimiento, Reusabilidad
Ambiente:	En un servidor de 1 GB de memoria RAM en estado normal de explotación.
Estímulo:	Se realizan operaciones en el sistema ya sea mediante acciones de inserción, búsqueda, modificación o eliminación.
Respuesta:	Se recibe la notificación de la acción realizada en un período de 0.1 a 0.7 segundos.

Al hacer un análisis de la situación anterior, fue posible comprobar que al realizar operaciones en el sistema, ya sea de inserción, búsqueda, modificación o eliminación en un ambiente donde el servidor se encuentra en estado normal de explotación, es decir que no hay latencia en la conexión, no hay fallas en el sistema, contando con un

servidor de 1 GB de memoria RAM; se recibe la notificación de la acción realizada en un período de 0.1 a 0.7 segundos.

Cuando se interactúa con una vista se dispara una clase controladora o sea, un js que llama a un controlador, este controlador no se comunica directamente con la clase que implementa el negocio si no que carga e instancia el objeto de Doctrine. Este objeto carga de la base de datos desde donde va a las tablas por PDO. El Data Base Object a partir del PDO va a las tablas, carga la información construye el objeto y se lo devuelve al controlador que se comunica con la entidad y le pasa el objeto de dominio, se procesa la lógica en el gestor encargado de la lógica propia del negocio, cálculos y procesamiento del negocio. El controlador si tiene que persistir los datos vuelve a ir al Doctrine y le manda el objeto. El Doctrine repite todo el mecanismo y regresa y concluye la aplicación. Una vez que concluye el controlador re-dibuja el js.

El Framework a partir de cada tabla genera un objeto y toda la lógica asociada a modificar, eliminar, buscar y adicionar sin programar las clases. Solo se programa la lógica de negocio y el controlador que tiene. Aún cuando el rendimiento se ve afectado debido a que tiene que moverse por 6 componentes arquitectónicos es más productivo, porque se genera mucho código automático que aumenta la reutilización. Esta evita tener que crear una cadena de conexión, la transacción, evita por cada tabla tener que crear una clase de atributos y además crear los métodos de adicionar, modificar, eliminar y buscar.

La evaluación de la arquitectura dio como resultado que el rendimiento es aceptable y la solución es reutilizable. Se llegó a la conclusión de que exista el riesgo de que disminuya el rendimiento en caso de existir peticiones concurrentes al sistema, aunque este riesgo puede ser matizado por un aumento en la reutilización, atributos que se encuentran en el punto de sensibilidad. Además que para cualquier acción es necesario pasar por 6 componentes arquitectónicos, lo que constituye el punto de desventaja en este escenario.

El trabajo realizado en este capítulo ha reflejado que la aplicación del método ATAM para evaluar la arquitectura es de mucha utilidad, ya que permite una evaluación temprana de la misma. De modo que esta no tiene que estar totalmente definida para ser evaluada, unido a ello está que dichas evaluaciones parciales van a tributar a la evaluación de la arquitectura completa. El proceso evaluativo realizado permite anticipar

el comportamiento de componentes que serán añadidos posteriormente a la arquitectura, siempre y cuando estos cumplan con los mismos patrones.

CONCLUSIONES

Como resultado del desarrollo de la arquitectura de la línea Capital Humano se definieron los componentes arquitectónicos para dar solución a cada uno de los procesos identificados.

Para ello se aplicaron un conjunto de patrones que facilitaron el proceso de implementación, normas dictadas por el equipo central de arquitectura, así como el marco de trabajo establecido; lo que contribuyó a facilitar y estandarizar la solución. Se priorizaron los componentes según el nivel de complejidad con el propósito de realizar el plan de iteración, se identificaron los contratos internos y externos necesarios para integrar cada uno de los componentes así como la solución con otros subsistemas.

Por otro lado se documentaron los artefactos generados por el equipo de arquitectura de sistema de la línea y finalmente se evaluó la arquitectura implementada obteniendo un resultado aceptable. De esta forma se da cumplimiento al objetivo trazado.

RECOMENDACIONES

La arquitectura definida para la línea Capital Humano diseñó una solución configurable a entidades presupuestadas y empresariales, por lo que en base a la experiencia adquirida los autores recomiendan el presente trabajo para reutilizar la solución técnica y/o las ideas tratadas en el diseño, integrarse a otros Sistemas Integrales de Gestión, implementar mejoras a la solución propuesta así como aplicar métodos de evaluación a componentes más específicos.

REFERENCIAS BIBLIOGRÁFICAS

1. **Perry, Dewayne and Garlan, David.** *Introduction to the Special Issue on Software Architecture.* 1995.

2. **Clements, Paul.** *A Survey of Architecture Description Languages.* 1996.

3. **Kazman, Rick, Clements, Paul and Bass, Len.** *Software Architecture in Practice.* 1998.

4. **Gómez, Diego.** *Cómo documentar la arquitectura de software.* 2008.

5. **Reynoso, Carlos and Kiccillof, Nicolás.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2004.

6. *Introducción a la Arquitectura de Software.*

7. **Billy, Carlos.** *Introducción a la Arquitectura de Software.* 2004.

8. **Szyperski, Clemens Alden.** *Component software: Beyond Object-Oriented programming.* 1998.

9. **Casal, Julio.** *Desarrollo de Software basado en Componentes.* s.a.

10. **Shaw, Mary and Garlan, David.** *Software Architecture: Perspectives on an emerging discipline.* 1996.

11. Estilos Arquitectónicos.

12. Documento de Arquitectura.

13. **Bruce, Eckel.** *Thinking in Patterns with Java.* s.a.

14. **Larman, Craig.** *UML y Patrones: Introducción al Análisis y programación orientada a objeto.* s.a.

15. **Facultad de informática - Universidad Politécnica de Madrid.** *Patrones del "Gang of Four".*

16. [Online] [Cited: enero 19 , 2009.] http://www.cii-murcia.es/informas/ene05/articulos/Arquitectura_y_disenyo_de_sistemas_web_modernos.pdf.
.

17. [Online] http://www.desarrolloweb.com/articulos/449.php.

18. [Online] [Cited: 20 febrero, 2009.] http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML.

19. [Online] [Cited: enero 26, 2009.] http://www.lawebera.es/manuales/javascript/caracteristicas.php.

20. [Online] [Cited: enero 27, 2009.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

21. **Eguíluz Pérez, Javier.** *Introducción a JavaScript.* 2007.

22. [Online] [Cited: febrero 3, 2009.] http://www.zonamasters.com/2007/06/27/que-es-ajax/.

23. [Online] [Cited: febrero 17, 2009.] http://www.cristalab.com/tutoriales/tutorial-de-ajax-c162l/..

24. [Online] [Cited: febrero 4, 2009.] http://www.nociondigital.com/webmasters/php-tutorial-que-es-php-como-surgio-y-para-que-se-utiliza-detalle-191.html.

25. [Online] [Cited: febrero 7, 2009.] http://linux.ciberaula.com/articulo/linux_apache_intro/..

26. [Online] [Cited: marzo 3, 2009.] http://www.blog.packman.es/tag/wamp/.

27. [Online] [Cited: marzo 11, 2009.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

28. [Online] [Cited: marzo 11, 2009.] http://pixelco.us/blog/5-frameworks-para-crear-interfaces-web/.

29. [Online] [Cited: marzo 14, 2009.] http://www.joangarnet.com/blog/?p=415.

30. [Online] http://www.joangarnet.com/blog/?p=415.

31. [Online] [Cited: marzo 18, 2009.] http://www.guia-ubuntu.org/index.php?title=Subversion.

32. [Online] [Cited: marzo 18, 2009.] http://tortoisesvn.net/docs/nightly/TortoiseSVN_es/tsvn-intro-features.html..

33. [Online] [Cited: marzo 20, 2009.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/..

34. [Online] [Cited: marzo 19, 2009.] http://www.monografias.com/trabajos-pdf2/sistema-gestion-base-datos-postgresql/sistema-gestion-base-datos-postgresql.pdf.

35. [Online] [Cited: marzo 18, 2009.] http://www.vivalinux.com.ar/soft/postgresql-8.3.html.

36. [Online] [Cited: abril 2, 2009.] http://www.taringa.net/posts/info/1649223/Que-IDE-usas---JAVA.html.

37. [Online] [Cited: marzo 19, 2009.] http://en.calameo.com/books/0000446953a6e3deb1aaa.

38. [Online] [Cited: marzo 19, 2009.] http://www.taringa.net/posts/info/1649223/Que-IDE-usas---JAVA.html.

39. [Online] [Cited: marzo 20, 2009.] http://www.genbeta.com/web/aptana-ide-para-aplicaciones-ajax.

40. [Online] [Cited: marzo 21, 2009.] http://www.maestrosdelweb.com/editorial/zendstudio/.

41. [Online] [Cited: marzo 20, 2009.] http://foros.cristalab.com/nuevo-en-zend-studio-v6.0.1-para-eclipse-t57772/.

42. [Online] [Cited: abril 3, 2009.] http://www.ingenierosoftware.com/analisisydiseno/uml.php.

43. [Online] [Cited: abril 3, 2009.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_%5Bcuenta_de_Windows_14723_p/free.htm.

44. **CAMACHO, ERIKA, CARDESO, FABIO and NUÑEZ, GABRIEL.** *ARQUITECTURAS DE SOFTWARE: GUÍA DE ESTUDIO.* 2004.

45. **Céspedes Vega, Anisleydis, et al.** *Procedimiento para Evaluar Arquitecturas de Software.*

BIBLIOGRAFÍA

1. **Perry, Dewayne and Garlan, David.** *Introduction to the Special Issue on Software Architecture.* 1995.

2. **Clements, Paul.** *A Survey of Architecture Description Languages.* 1996.

3. **Kazman, Rick, Clements, Paul and Bass, Len.** *Software Architecture in Practice.* 1998.

4. **Gómez, Diego.** *Cómo documentar la arquitectura de software.* 2008.

5. **Reynoso, Carlos and Kiccillof, Nicolás.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2004.

6. *Introducción a la Arquitectura de Software.*

7. **Billy, Carlos.** *Introducción a la Arquitectura de Software.* 2004.

8. **Szyperski, Clemens Alden.** *Component software: Beyond Object-Oriented programming.* 1998.

9. **Casal, Julio.** *Desarrollo de Software basado en Componentes.* s.a.

10. **Shaw, Mary and Garlan, David.** *Software Architecture: Perspectives on an emerging discipline.* 1996.

11. Estilos Arquitectónicos.

12. Documento de Arquitectura.

13. **Bruce, Eckel.** *Thinking in Patterns with Java.* s.a.

14. **Larman, Craig.** *UML y Patrones: Introducción al Análisis y programación orientada a objeto.* s.a.

15. **Facultad de informática - Universidad Politécnica de Madrid.** *Patrones del "Gang of Four".*

16. [Online] [Cited: enero 19 , 2009.] http://www.cii-murcia.es/informas/ene05/articulos/Arquitectura_y_disenyo_de_sistemas_web_modernos.pdf..

17. [Online] http://www.desarrolloweb.com/articulos/449.php.

18. [Online] [Cited: 20 febrero, 2009.] http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML.

19. [Online] [Cited: enero 26, 2009.] http://www.lawebera.es/manuales/javascript/caracteristicas.php.

20. [Online] [Cited: enero 27, 2009.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

21. **Eguíluz Pérez, Javier.** *Introducción a JavaScript.* 2007.

22. [Online] [Cited: febrero 3, 2009.] http://www.zonamasters.com/2007/06/27/que-es-ajax/.

23. [Online] [Cited: febrero 17, 2009.] http://www.cristalab.com/tutoriales/tutorial-de-ajax-c162l/..

24. [Online] [Cited: febrero 4, 2009.] http://www.nociondigital.com/webmasters/php-tutorial-que-es-php-como-surgio-y-para-que-se-utiliza-detalle-191.html.

25. [Online] [Cited: febrero 7, 2009.] http://linux.ciberaula.com/articulo/linux_apache_intro/..

26. [Online] [Cited: marzo 3, 2009.] http://www.blog.packman.es/tag/wamp/.

27. [Online] [Cited: marzo 11, 2009.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

28. [Online] [Cited: marzo 11, 2009.] http://pixelco.us/blog/5-frameworks-para-crear-interfaces-web/.

29. [Online] [Cited: marzo 14, 2009.] http://www.joangarnet.com/blog/?p=415.

30. [Online] http://www.joangarnet.com/blog/?p=415.

31. [Online] [Cited: marzo 18, 2009.] http://www.guia-ubuntu.org/index.php?title=Subversion.

32. [Online] [Cited: marzo 18, 2009.] http://tortoisesvn.net/docs/nightly/TortoiseSVN_es/tsvn-intro-features.html..

33. [Online] [Cited: marzo 20, 2009.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/..

34. [Online] [Cited: marzo 19, 2009.] http://www.monografias.com/trabajos-pdf2/sistema-gestion-base-datos-postgresql/sistema-gestion-base-datos-postgresql.pdf.

35. [Online] [Cited: marzo 18, 2009.] http://www.vivalinux.com.ar/soft/postgresql-8.3.html.

36. [Online] [Cited: abril 2, 2009.] http://www.taringa.net/posts/info/1649223/Que-IDE-usas---JAVA.html.

37. [Online] [Cited: marzo 19, 2009.] http://en.calameo.com/books/0000446953a6e3deb1aaa.

38. [Online] [Cited: marzo 19, 2009.] http://www.taringa.net/posts/info/1649223/Que-IDE-usas---JAVA.html.

39. [Online] [Cited: marzo 20, 2009.] http://www.genbeta.com/web/aptana-ide-para-aplicaciones-ajax.

40. [Online] [Cited: marzo 21, 2009.] http://www.maestrosdelweb.com/editorial/zendstudio/.

41. [Online] [Cited: marzo 20, 2009.] http://foros.cristalab.com/nuevo-en-zend-studio-v6.0.1-para-eclipse-t57772/.

42. [Online] [Cited: abril 3, 2009.] http://www.ingenierosoftware.com/analisisydiseno/uml.php.

43. [Online] [Cited: abril 3, 2009.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_%5Bcuenta_de_Windows_14723_p/free.htm.

44. **CAMACHO, ERIKA, CARDESO, FABIO and NUÑEZ, GABRIEL.** *ARQUITECTURAS DE SOFTWARE: GUÍA DE ESTUDIO.* 2004.

45. **Céspedes Vega, Anisleydis, et al.** *Procedimiento para Evaluar Arquitecturas de Software.*

46. [Online] [Cited: febrero 3, 2009.] http://www.maestrosdelweb.com/editorial/zendstudio/.

47. [Online] [Cited: febrero 3, 2009.] http://www.scribd.com/doc/3930805/Patrones-de-Diseno.

48. [Online] [Cited: febrero 4, 2009.] http://java.ciberaula.com/articulo/diseno_patrones_j2ee/.

49. [Online] [Cited: febrero 17, 2009.] http://www.ajaxya.com.ar/temarios/descripcion.php?cod=8&punto=1.

50. **Eguíluz Pérez, Javier.** *Introduccion a AjAX.* 2007.

51. [Online] [Cited: marzo 2, 2009.] http://www.eici.ucm.cl/Academicos/R_Villarroel/descargas/ing_sw_1.

52. [Online] [Cited: marzo 3, 2009.] http://es.tech-faq.com/wamp.shtml&prev=hp..

53. [Online] [Cited: marzo 14, 2009.] http://onready.blogspot.com/2009/05/extjs-1-definicion.html..

54. [Online] [Cited: marzo 14, 2009.] http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/..

55. [Online] [Cited: febrero 15, 2009.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/..

56. [Online] [Cited: marzo 19, 2009.] http://www.publish.cl/postgresql.html.

57. [Online] [Cited: abril 3, 2009.] http://www.dosideas.com/wiki/IDE.

58. [Online] [Cited: abril 4, 2009.] http://www.gtdownload.com/es/libre-descargue/Desarrollo-categoria/Herramientas-De-la-Ayuda-subcategoria/Visual-Paradigm-for-UML-Standard-Edition-for-Mac-OS-X-detalles-de-la-vision.html.

59. **Kazman, Rick, Klein, Mark and Clements, Paul.** *ATAM: Method for Architecture Evaluation.* 2009.

60. *Arquitectura de Software: Documentación.* 2006.

61. **Microsoft.** The Role of an Architect. *The Architecture Journal.*

62. **Jesús, García.** Patrones de Diseño.

63. **Reynoso, Carlos.** *Métodos Heterodoxos en Desarrollo de Software.* Buenos Aires : s.n., 2004.

64. Desarrollo de Aplicaciones basado en Componentes y Frameworks.

65. Análisis y Diseño de Software.

66. **MÖHRKE, CARSTEN.** Exploración del Entorno de Desarrollo Zend Studio 4.

67. [Online] marzo 25, 2009. http://pixelco.us/blog/5-frameworks-para-crear-interfaces-web/.

68. **Romo Portilla, Luis Yovany and Moreno, Dorance.** *Arquitectura de Software: Una disciplina emergente y creciente.*

69. **Mejia Alvarez, Pedro.** Patrones de Diseño.

70. **Microsoft.** Arquitectura de Infraestructura. *The Architecture Journal.*

71. **Kruchten, Philippe.** *Planos Arquitectónicos: El Modelo de "4+1" Vistas de la Arquitectura de Software.* 1995.

72. **ERP Cuba.** *Documento Especificación de la Arquitectura de Sistema e Integración ERP Cuba.* 2008.

73. —. *Estándar de diseño de interfaces para las aplicaciones de gestión.* 2008.

74. —. *Ayuda al usuario del caso de estudio en el nuevo marco de trabajo.* 2008.

ANEXOS

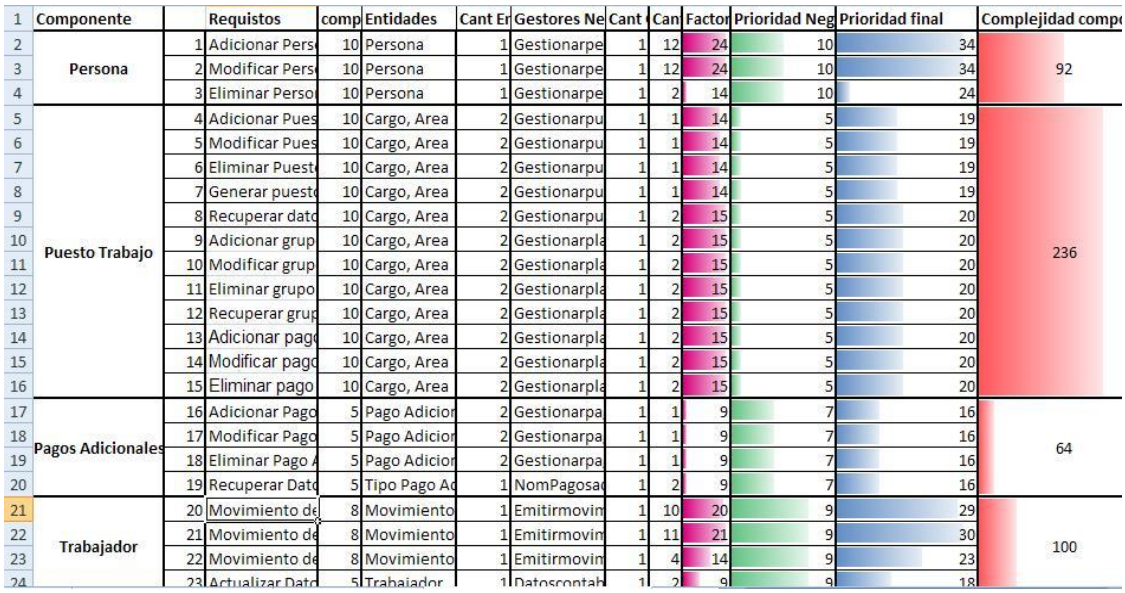
Anexo 1 Priorización de los requisitos.

No	Requisitos	CD	CC	CCP	CN	SARF
1	Adicionar Persona	4	1	12	8	25
2	Modificar Persona	0	1	12	8	21
3	Eliminar Persona	0	1	2	2	5
4	Adicionar Puesto de trabajo	6	1	1	6	14
5	Modificar Puesto de Trabajo	0	1	1	6	8
6	Eliminar Puesto de Trabajo	0	1	1	3	5
7	Generar puestos de trabajo	0	1	1	7	9
8	Recuperar datos de Puesto de Trabajo	0	1	2	1	4
9	Adicionar grupo puesto de trabajo	7	3	2	6	18
10	Modificar grupo puesto de trabajo	0	3	2	6	11
11	Eliminar grupo puesto de trabajo	0	3	2	3	8
12	Recuperar grupo datos del puesto de trabajo	0	3	2	1	6
13	Adicionar pago adicional a grupo de trabajo	4	4	2	1	11
14	Modificar pago adicional de un grupo de trabajo	0	4	2	1	7
15	Eliminar pago adicional al grupo de trabajo	0	4	2	1	7
16	Adicionar Pago Adicional	3	1	1	4	9
17	Modificar Pago Adicional	0	1	1	4	6
18	Eliminar Pago Adicional	0	1	1	1	3
19	Recuperar Datos de Tipo de Pago Adicional	0	1	2	1	4
20	Movimiento de Alta	7	3	10	9	29
21	Movimiento de Reubicación	0	3	11	9	23
22	Movimiento de Bajas	0	3	4	4	11
23	Actualizar Datos Contables Trabajador	0	1	2	5	8
24	Adicionar Asociación de Incidencia-Impuesto	2	2	1	5	10

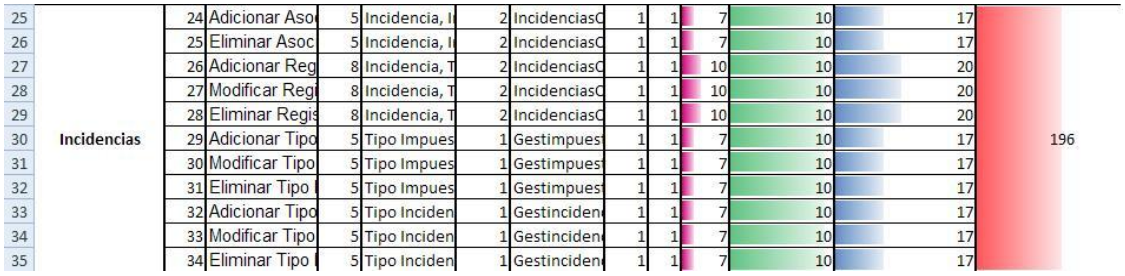
25	Eliminar Asociación de Incidencia-Impuesto	2	2	1	4	9
26	Adicionar Registro de Incidencia	4	3	1	4	12
27	Modificar Registro de Incidencia	0	3	1	4	8
28	Eliminar Registro de Incidencia	1	3	1	1	6
29	Adicionar Tipo Impuesto	4	1	1	4	10
30	Modificar Tipo Impuesto	0	1	1	4	6
31	Eliminar Tipo Impuesto	0	1	1	1	3
32	Adicionar Tipo Incidencias	3	1	1	5	10
33	Modificar Tipo Incidencias	0	1	1	5	7
34	Eliminar Tipo Incidencias	0	1	1	1	3
35	Crear Nómina	3	2	2	10	17
36	Agregar trabajadores a la nómina	1	2	2	8	13
37	Adicionar Períodos de Pago	3	1	1	4	9
38	Modificar Períodos de Pago	0	1	1	4	6
39	Eliminar Períodos de Pago	0	1	1	1	3
40	Crear Documento de Ajuste	3	3	1	6	13
41	Modificar Documento de Ajuste	0	3	1	6	10
42	Eliminar Documento de Ajuste	0	1	1	2	4
43	Adicionar Tipos de Ajuste	4	1	1	4	10
44	Modificar Tipos de Ajuste	0	1	1	4	6
45	Eliminar Tipos de Ajuste	0	1	1	1	3
46	Procesar Documento de Ajuste	0	3	1	10	14
47	Adicionar Tipo de Nómina	3	1	1	4	9
48	Modificar Tipo de Nómina	0	1	1	4	6
49	Eliminar Tipo de Nómina	0	1	1	1	3
50	Emitir Comprobante Operaciones	1	2	1	4	8
51	Adicionar Tipo de Retención	3	1	1	4	9
52	Modificar Tipo de Retención	0	1	1	1	3
53	Eliminar Tipo de Retención	0	1	1	1	3
54	Adicionar Tipo de Programa	4	1	1	3	9
55	Modificar Tipo de Programa	0	1	1	3	5
56	Eliminar Tipo de Programa	0	1	1	1	3
57	Apertura SubMayor de Vacaciones	4	1	1	6	1

						2
58	Cierre SubMayor de Vacaciones	0	1	1	6	8
59	Actualizar SubMayor de Vacaciones	0	2	1	6	9
60	Detalles SubMayor de Vacaciones	0	1	1	4	6
61	Conciliación con Contabilidad de SubMayor de Vacaciones	1	2	1	6	10
62	Apertura SubMayor de Retenciones	4	3	2	8	17
63	Modificar Registro Retenciones-Trabajador	0	3	2	6	11
64	Cierre Submayor de Retenciones	0	1	1	6	8
65	Detalles SubMayor de Retenciones de un Trabajador	0	1	1	4	6
66	Actualizar SubMayor de Retenciones	0	2	2	1	5
67	Definir porciento de Vacaciones	0	1	1	1	3
68	Definir cuenta de Vacaciones	0	1	1	1	3
69	Cierre contable	0	0	0	9	9
70	Procesar Nómina	1	6	0	10	17

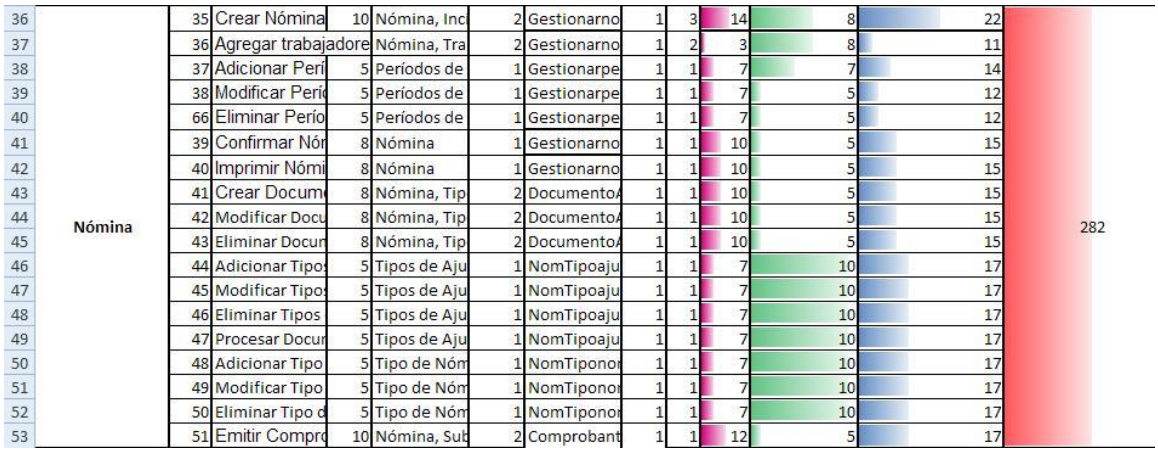
Anexo 2 Priorización de los componentes



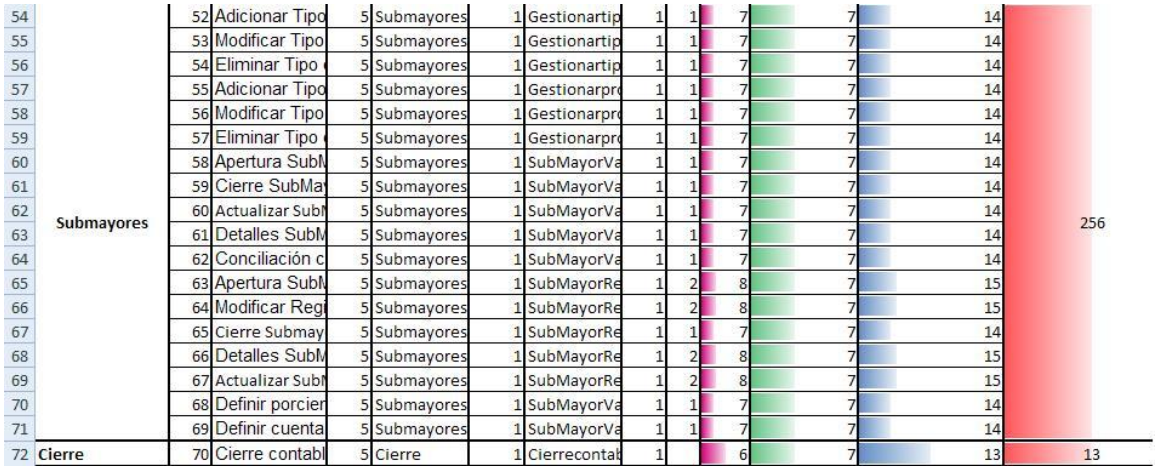
	Componente		Requistos	comp	Entidades	Cant Er	Gestores Ne	Cant	Can	Factor	Prioridad Neg	Prioridad final	Complejidad compo
2	Persona	1	Adicionar Pers	10	Persona	1	Gestionarpe	1	12	24	10	34	92
3		2	Modificar Pers	10	Persona	1	Gestionarpe	1	12	24	10	34	
4		3	Eliminar Perso	10	Persona	1	Gestionarpe	1	2	14	10	24	
5	Puesto Trabajo	4	Adicionar Pues	10	Cargo, Area	2	Gestionarpu	1	1	14	5	19	236
6		5	Modificar Pues	10	Cargo, Area	2	Gestionarpu	1	1	14	5	19	
7		6	Eliminar Puest	10	Cargo, Area	2	Gestionarpu	1	1	14	5	19	
8		7	Generar puest	10	Cargo, Area	2	Gestionarpu	1	1	14	5	19	
9		8	Recuperar dato	10	Cargo, Area	2	Gestionarpu	1	2	15	5	20	
10		9	Adicionar grup	10	Cargo, Area	2	Gestionarpla	1	2	15	5	20	
11		10	Modificar grup	10	Cargo, Area	2	Gestionarpla	1	2	15	5	20	
12		11	Eliminar grupo	10	Cargo, Area	2	Gestionarpla	1	2	15	5	20	
13		12	Recuperar grup	10	Cargo, Area	2	Gestionarpla	1	2	15	5	20	
14		13	Adicionar pago	10	Cargo, Area	2	Gestionarpla	1	2	15	5	20	
15		14	Modificar pago	10	Cargo, Area	2	Gestionarpla	1	2	15	5	20	
16		15	Eliminar pago	10	Cargo, Area	2	Gestionarpla	1	2	15	5	20	
17	Pagos Adicionales	16	Adicionar Pago	5	Pago Adicion	2	Gestionarpa	1	1	9	7	16	64
18		17	Modificar Pago	5	Pago Adicion	2	Gestionarpa	1	1	9	7	16	
19		18	Eliminar Pago A	5	Pago Adicion	2	Gestionarpa	1	1	9	7	16	
20		19	Recuperar Dato	5	Tipo Pago Ad	1	NomPagosad	1	2	9	7	16	
21	Trabajador	20	Movimiento de	8	Movimiento	1	Emitirmovim	1	10	20	9	29	100
22		21	Movimiento de	8	Movimiento	1	Emitirmovim	1	11	21	9	30	
23		22	Movimiento de	8	Movimiento	1	Emitirmovim	1	4	14	9	23	
24		23	Actualizar Dato	5	Trabajador	1	Datoscontab	1	2	9	9	18	



25		24	Adicionar Aso	5	Incidencia, I	2	IncidenciasC	1	1	7	10	17		
26		25	Eliminar Asoc	5	Incidencia, I	2	IncidenciasC	1	1	7	10	17		
27		26	Adicionar Reg	8	Incidencia, T	2	IncidenciasC	1	1	10	10	20		
28		27	Modificar Regi	8	Incidencia, T	2	IncidenciasC	1	1	10	10	20		
29		28	Eliminar Regis	8	Incidencia, T	2	IncidenciasC	1	1	10	10	20		
30	**Incidencias**	29	Adicionar Tipo	5	Tipo Impues	1	Gestimpuest	1	1	7	10	17		196
31		30	Modificar Tipo	5	Tipo Impues	1	Gestimpuest	1	1	7	10	17		
32		31	Eliminar Tipo	5	Tipo Impues	1	Gestimpuest	1	1	7	10	17		
33		32	Adicionar Tipo	5	Tipo Inciden	1	Gestinciden	1	1	7	10	17		
34		33	Modificar Tipo	5	Tipo Inciden	1	Gestinciden	1	1	7	10	17		
35		34	Eliminar Tipo	5	Tipo Inciden	1	Gestinciden	1	1	7	10	17		

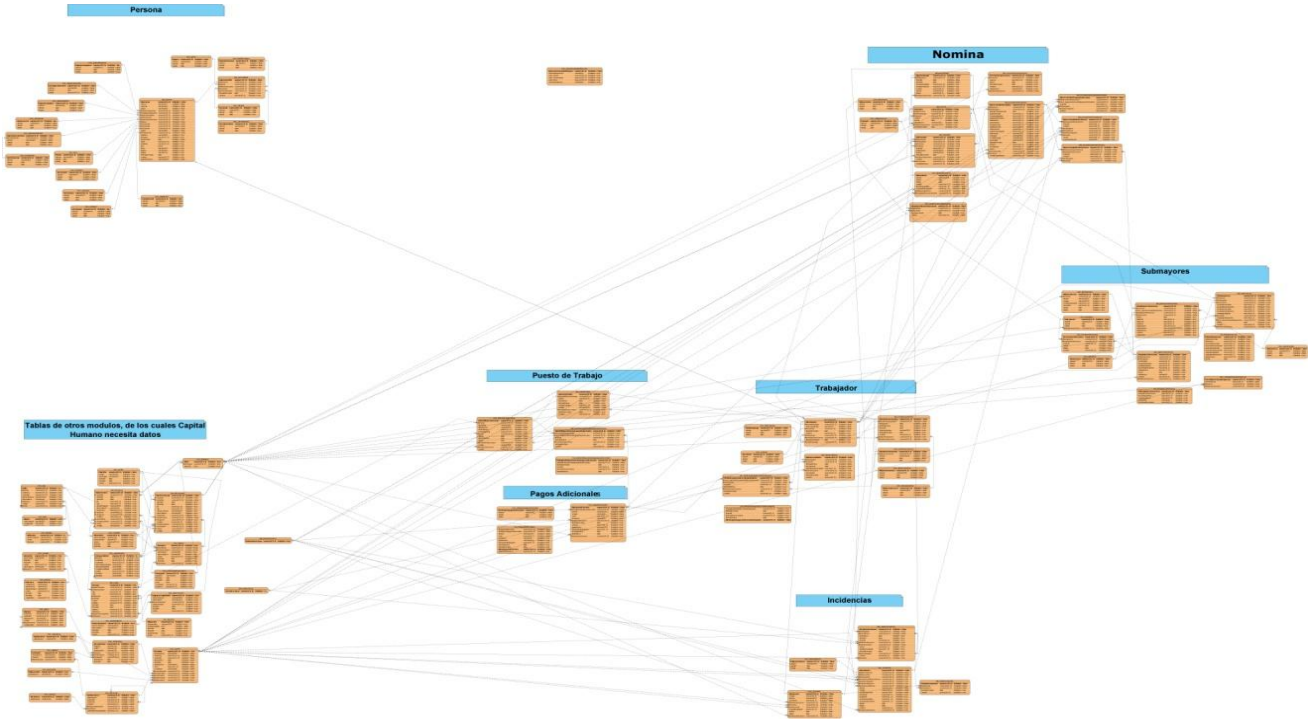


36		35	Crear Nómina	10	Nómina, Inc	2	Gestionarno	1	3	14	8	22		
37		36	Agregar trabajadore		Nómina, Tra	2	Gestionarno	1	2	3	8	11		
38		37	Adicionar Perí	5	Períodos de	1	Gestionarpe	1	1	7	7	14		
39		38	Modificar Perí	5	Períodos de	1	Gestionarpe	1	1	7	5	12		
40		66	Eliminar Perío	5	Períodos de	1	Gestionarpe	1	1	7	5	12		
41		39	Confirmar Nór	8	Nómina	1	Gestionarno	1	1	10	5	15		
42		40	Imprimir Nómi	8	Nómina	1	Gestionarno	1	1	10	5	15		
43		41	Crear Docum	8	Nómina, Tip	2	DocumentoA	1	1	10	5	15		
44	**Nómina**	42	Modificar Docu	8	Nómina, Tip	2	DocumentoA	1	1	10	5	15		282
45		43	Eliminar Docun	8	Nómina, Tip	2	DocumentoA	1	1	10	5	15		
46		44	Adicionar Tipos	5	Tipos de Aju	1	NomTipoaju	1	1	7	10	17		
47		45	Modificar Tipos	5	Tipos de Aju	1	NomTipoaju	1	1	7	10	17		
48		46	Eliminar Tipos	5	Tipos de Aju	1	NomTipoaju	1	1	7	10	17		
49		47	Procesar Docur	5	Tipos de Aju	1	NomTipoaju	1	1	7	10	17		
50		48	Adicionar Tipo	5	Tipo de Nóm	1	NomTiponor	1	1	7	10	17		
51		49	Modificar Tipo	5	Tipo de Nóm	1	NomTiponor	1	1	7	10	17		
52		50	Eliminar Tipo d	5	Tipo de Nóm	1	NomTiponor	1	1	7	10	17		
53		51	Emitir Compro	10	Nómina, Sub	2	Comprobant	1	1	12	5	17		



54		52	Adicionar Tipo	5	Submayores	1	Gestionartip	1	1	7	7	14		
55		53	Modificar Tipo	5	Submayores	1	Gestionartip	1	1	7	7	14		
56		54	Eliminar Tipo	5	Submayores	1	Gestionartip	1	1	7	7	14		
57		55	Adicionar Tipo	5	Submayores	1	Gestionarpr	1	1	7	7	14		
58		56	Modificar Tipo	5	Submayores	1	Gestionarpr	1	1	7	7	14		
59		57	Eliminar Tipo	5	Submayores	1	Gestionarpr	1	1	7	7	14		
60		58	Apertura SubM	5	Submayores	1	SubMayorVa	1	1	7	7	14		
61		59	Cierre SubMay	5	Submayores	1	SubMayorVa	1	1	7	7	14		
62		60	Actualizar SubM	5	Submayores	1	SubMayorVa	1	1	7	7	14		
63	**Submayores**	61	Detalles SubM	5	Submayores	1	SubMayorVa	1	1	7	7	14		256
64		62	Conciliación c	5	Submayores	1	SubMayorVa	1	1	7	7	14		
65		63	Apertura SubM	5	Submayores	1	SubMayorRe	1	2	8	7	15		
66		64	Modificar Regi	5	Submayores	1	SubMayorRe	1	2	8	7	15		
67		65	Cierre Submay	5	Submayores	1	SubMayorRe	1	1	7	7	14		
68		66	Detalles SubM	5	Submayores	1	SubMayorRe	1	2	8	7	15		
69		67	Actualizar SubM	5	Submayores	1	SubMayorRe	1	2	8	7	15		
70		68	Definir porcier	5	Submayores	1	SubMayorVa	1	1	7	7	14		
71		69	Definir cuenta	5	Submayores	1	SubMayorVa	1	1	7	7	14		
72	**Cierre**	70	Cierre contabl	5	Cierre	1	Cierrecontab	1		6	7	13		13

Anexo 3 Modelo de Datos.


Persona
Nomina
Submayores
Puesto de Trabajo
Trabajador
Tablas de otros modulos, de los cuales Capital Humano necesita datos
Pagos Adicionales
Incidencias

Anexo 4 Plan de Iteración.

tarea	Comienzo	Fin	% completado	Nombres de los recursos	Duración	Trabajo
CAPITAL HUMANO	**mar 7/1/08**	**vie 2/27/09**	**97%**		**160.94 días?**	**76,455.08 horas**
Inicio	**mar 7/1/08**	**mar 9/2/08**	**100%**		**47 días?**	**14,382 horas**
Elaboración	**vie 9/5/08**	**lun 10/6/08**	**100%**		**17.03 días?**	**5,806.5 horas**
Taller de Análisis	vie 9/5/08	lun 10/6/08	100%	Arquitecto de Sistema,Arquitecto	305.5 horas	3,055 horas
Diagrama de componentes	vie 9/5/08	lun 10/6/08	100%	Arquitecto de Datos,Arquitecto,A	16.97 días?	916.5 horas
Agrupación de requisitos por componentes	vie 9/5/08	lun 10/6/08	100%	Arquitecto de Datos,Arquitecto,A	16.97 días?	1,527.5 horas
Prioridad de los componentes	vie 9/5/08	lun 10/6/08	100%	Arquitecto	16.97 días?	305.5 horas
Aprobación de artefactos	lun 10/6/08	lun 10/6/08	100%	Jefe de Línea,Arquitecto	0.06 días	2 horas
Construcción	**mié 10/8/08**	**lun 1/19/09**	**99%**		**65.42 días?**	**51,682.83 horas**
Componentes	**mié 10/8/08**	**lun 1/19/09**	**99%**		**65.42 días?**	**50,780.83 horas**
Persona	**mié 10/8/08**	**jue 12/4/08**	**100%**		**31.64 días?**	**990.02 horas**
Entrega a calidad del componente Persona	jue 12/4/08	jue 12/4/08	100%	Javier Heredia Ruiz,Rubén R	0 días	0 horas
Pagos adicionales	**sáb 11/1/08**	**lun 1/12/09**	**99%**		**46.06 días?**	**5,763 horas**
Entrega a calidad del componente Pagos adicionales	vie 12/5/08	vie 12/5/08	100%	Javier Heredia Ruiz	0 días	0 horas
Puesto de Trabajo	**lun 11/10/08**	**lun 1/12/09**	**100%**		**41.81 días?**	**5,790.98 horas**
Entrega a calidad del componente Puesto de Trabajo	lun 1/12/09	lun 1/12/09	100%	Javier Heredia Ruiz	0 días	0 horas
Trabajador	**lun 11/10/08**	**vie 1/9/09**	**100%**		**40.91 días?**	**13,072.28 horas**
Entrega a calidad de Trabajador	lun 12/22/08	lun 12/22/08	100%	Javier Heredia Ruiz	0 días	0 horas
Incidencias	**mié 10/8/08**	**vie 12/12/08**	**100%**		**38.06 días?**	**3,643.12 horas**
Entrega a calidad de incidencias	lun 12/22/08	lun 12/22/08	100%	Javier Heredia Ruiz	0 días	0 horas
Nómina	**lun 11/17/08**	**lun 1/19/09**	**99%**	**Osvaldo Díaz ,Wendy Gracia**	**43.64 días?**	**16,904.93 horas**
Entrega a calidad de Nómina	jue 1/8/09	jue 1/8/09	100%	Javier Heredia Ruiz	0 días	0 horas
Submayores	**lun 11/24/08**	**lun 1/12/09**	**100%**		**34.08 días?**	**4,111.52 horas**
Entrega a calidad de Submayores	vie 12/12/08	vie 12/12/08	100%	Javier Heredia Ruiz,Rubén R	0 días	0 horas
Recuperaciones de Capital Humano	**lun 12/22/08**	**mié 1/14/09**	**100%**		**17.89 días?**	**484 horas**
Entrega de Recuperaciones de CH	sáb 1/17/09	sáb 1/17/09	100%	Rubén Rodríguez Torres,Javier H	0.39 días?	21 horas
Soporte a la Arquitectura de datos de los componentes de la línea	mié 10/8/08	sáb 12/27/08	100%	Alexei Olivero Márquez	49.11 días?	884 horas
Soporte de desarrollo (No conformidades) de la línea CH	vie 12/26/08	sáb 12/27/08	100%	Yainelis Carbonell Toral	1 día?	18 horas
Cierre	**mié 10/8/08**	**vie 2/27/09**	**37%**		**93.8 días?**	**4,583.75 horas**
Terminación CH Versión 1	vie 2/27/09	vie 2/27/09	0%	Rubén Rodríguez Torres	0 días	0 horas

tarea	Comienzo	Fin	Nombres de los recursos
Componentes	**mié 10/8/08**	**lun 1/19/09**	
Persona	#########	**jue 12/4/08**	
Diseño de Modelo de Datos	lun 11/24/08	mar 12/2/08	Yoenry Vanega Hechavarria
Creación de Modelo de Datos	lun 11/24/08	mar 12/2/08	Yoenry Vanega Hechavarria
Programación del Modelo de Datos	lun 11/24/08	mar 12/2/08	Yoenry Vanega Hechavarria
Diseño de Lógica de Negocio	lun 11/24/08	mar 12/2/08	Yoenry Vanega Hechavarria
Diseño de Interfaz de Usuario	lun 11/24/08	mar 12/2/08	Alexis Lorenzo Herrera
Validación de Interfaz de Usuario	lun 11/24/08	mar 12/2/08	Alexis Lorenzo Herrera
Reunión de Implementación	mié 10/8/08	jue 10/9/08	Yoenry Vanega Hechavarria
Terminar de migrar sistema cuadro	vie 11/21/08	vie 11/28/08	Yosney Hernandez,Alexis Lorenzo[30%],Yoenry Vanega[20%],Javier Limonta[25
Ingeniería inversa	lun 11/24/08	mar 12/2/08	Javier Heredia Ruiz,Crescencio Campos Hung
Integración de Persona con IoC Interno	lun 12/1/08	lun 12/1/08	Yoenry Vanega Hechavarria
Integración de Persona con IoC Externo	mar 12/2/08	mar 12/2/08	Yoenry Vanega Hechavarria
Integración de Persona con Aspect Template	mar 12/2/08	jue 12/4/08	Yoenry Vanega Hechavarria
Integración de Persona con Concurrencia	mar 12/2/08	jue 12/4/08	Yoenry Vanega Hechavarria
Integración de Persona con Validación del servidor	mar 12/2/08	mar 12/2/08	Yoenry Vanega Hechavarria
Integración de Persona con Excepciones	lun 12/1/08	lun 12/1/08	Yoenry Vanega Hechavarria
Integración de Persona con Portal	lun 12/1/08	lun 12/1/08	Yoenry Vanega Hechavarria
Integración de Persona con Transacciones	mar 12/2/08	jue 12/4/08	Yoenry Vanega Hechavarria
Integración de Persona con Multientidad	mar 12/2/08	jue 12/4/08	Yoenry Vanega Hechavarria
Pruebas internas a Persona	mar 12/2/08	mar 12/2/08	Javier Heredia Ruiz
Diseño de casos de prueba de Persona	lun 11/24/08	mié 12/3/08	Javier Heredia Ruiz
Entrega a calidad del componente Persona	jue 12/4/08	jue 12/4/08	Javier Heredia Ruiz,Rubén Rodríguez Torres

Nombre de tarea	Fin	Comienzo	Nombres de los recursos
Pagos adicionales	**lun 1/12/09**	#########	
Diseño de Modelo de Datos	mar 11/25/08	lun 11/10/08	Alexei Olivero Márquez,Hector Prada Nicot
Creación de Modelo de Datos	mar 11/25/08	lun 11/10/08	Alexei Olivero Márquez,Hector Prada Nicot
Programación del Modelo de Datos	mar 11/25/(	lun 11/10/08	Alexei Olivero Márquez,Hector Prada Nicot
Diseño de Lógica de Negocio	mar 11/25/08	lun 11/10/08	Alexei Olivero Márquez,Hector Prada Nicot
Diseño de Interfaz de Usuario	mar 11/25/08	lun 11/10/08	Alexei Olivero Márquez,Hector Prada Nicot
Validación de Interfaz de Usuario	mar 11/25/08	lun 11/10/08	Alexei Olivero Márquez,Hector Prada Nicot
Reunión de Implementación	**mié 12/3/08**	**lun 11/10/08**	**Alexei Olivero Márquez,Hector Prada Nicot**
Análisis	mié 12/3/08	lun 11/10/08	Yusmara Buchillón,Susana Bermúdez,Crescencio Campos
Implementación	**lun 12/8/08**	**sáb 11/1/08**	
Gestionar tipo de pago adicional(Nomenclador)	lun 12/8/08	sáb 11/1/08	Cesar Lage,José L. Céspedes,Carlos Vingut,Carlos M. Pedro
Integración de Pagos adicionales con el Nomenclador de CP	mar 12/2/08	lun 12/1/08	Hector Prada Nicot

Nombre de tarea	Fin	Comienzo	Nombres de los recursos
Puesto de Trabajo	**lun 1/12/09**	#########	
Diseño de Modelo de Datos	mar 11/25/08	lun 11/10/08	Danier Estévez Laborí,Lázaro Perez Bajuelo,Alexei Olivero Mé
Creación de Modelo de Datos	mar 11/25/08	lun 11/10/08	Danier Estévez Laborí,Lázaro Perez Bajuelo,Alexei Olivero Mé
Programación del Modelo de Datos	mar 11/25/08	lun 11/10/08	Danier Estévez Laborí,Lázaro Perez Bajuelo,Alexei Olivero Mé
Diseño de Lógica de Negocio	mar 11/25/08	lun 11/10/08	Danier Estévez Laborí,Lázaro Perez Bajuelo,Alexei Olivero Mé
Diseño de Interfaz de Usuario	mar 11/25/08	lun 11/10/08	Danier Estévez Laborí,Lázaro Perez Bajuelo,Alexei Olivero Mé
Validación de Interfaz de Usuario	mar 11/25/08	lun 11/10/08	Danier Estévez Laborí,Lázaro Perez Bajuelo,Alexei Olivero Mé
Reunión de Implementación	**lun 11/24/08**	**lun 11/10/08**	**Alexei Olivero,Lázaro Perez,Danier Estévez**
Análisis	lun 11/24/08	lun 11/10/08	Yusmara Buchillón Hernández,Crescencio Campos Hung
Implementación	**mié 12/10/08**	**lun 11/17/08**	**Danier Estévezí[50%],Lázaro Perez[50%],Daniellis Palau**
Gestionar Puestos de trabajo	**jue 12/4/08**	##########	
Implementar escenario adicionar	jue 12/4/08	mar 11/18/08	Lázaro Perez Bajuelo
Implementar escenario modificar	jue 12/4/08	mar 11/18/08	Lázaro Perez Bajuelo
Implementar escenario eliminar	jue 12/4/08	mar 11/18/08	Lázaro Perez Bajuelo
Terminación de la implementación realizada	jue 12/4/08	mar 11/18/08	Rosendo Leonardo Hernández
Gestionar Grupo de puesto de trabajo	jue 12/4/08	mar 12/2/08	Danier Estévez Laborí,Lázaro Perez Bajuelo
Asociar puesto de trabajo con pago adicional	mié 12/10/08	lun 11/17/08	Danier Estévez Laborí,Lázaro Perez Bajuelo
Integración de Puesto de Trabajo con EC (área)	vie 12/5/08	mié 12/3/08	Danier Estévez Laborí,Lázaro Perez Bajuelo
Integración de Puesto de Trabajo con EC (cargos)	vie 12/5/08	mié 12/3/08	Danier Estévez Laborí,Lázaro Perez Bajuelo
Integración de Puesto de Trabajo con IoC Externo	vie 12/5/08	mié 12/3/08	Danier Estévez Laborí,Lázaro Perez Bajuelo
Integración de Puesto de Trabajo con IoC Interno	vie 12/5/08	mié 12/3/08	Danier Estévez Laborí,Lázaro Perez Bajuelo

tarea	Fin	Comienzo	Nombres de los recursos
Puesto de Trabajo	**lun 1/12/09**	**lun 11/10/08**	
Entrega a calidad del componente Puesto de Trabajo	lun 1/12/09	lun 1/12/09	Javier Heredia Ruiz
Trabajador	**vie 1/9/09**	**lun 11/10/08**	
Diseño de Modelo de Datos	lun 12/1/08	mar 11/11/08	Oilede Perez Ramirez,Yonisley Garcia Lo
Creación de Modelo de Datos	lun 12/1/08	mar 11/11/08	Oilede Perez Ramirez,Yonisley Garcia Lo
Programación del Modelo de Datos	lun 12/1/08	mar 11/11/08	Oilede Perez Ramirez,Yonisley Garcia Lo
Diseño de Lógica de Negocio	lun 12/1/08	mar 11/11/08	Oilede Perez Ramirez,Yonisley Garcia Lo
Diseño de Interfaz de Usuario	lun 12/1/08	mar 11/11/08	Oilede Perez Ramirez,Yonisley Garcia Lo
Validación de Interfaz de Usuario	lun 12/1/08	mar 11/11/08	Oilede Perez Ramirez,Yonisley Garcia Lo
Reunión de Implementación	**jue 12/4/08**	**lun 11/10/08**	**Oilede Perez Ramirez,Yonisley Garc**
Análisis	vie 11/28/08	lun 11/10/08	Yulexy Vega Pérez ,Crescencio Campos
Análisis	jue 12/4/08	mar 12/2/08	Crescencio Campos Hung
Implementación	**jue 12/18/08**	**lun 11/10/08**	
Emitir Movimiento de nóminas	mar 12/9/08	##########	Oilede Perez,Yosniel Ramos,Yonisl
Gestionar tipo de movimiento de nómina	jue 12/18/08	lun 11/10/08	Yosniel Ramos,Yonisley Garcia,Oile
Gestionar motivo de movimiento de nómina	jue 12/4/08	mar 12/2/08	Carlos Miguel Pedroso Caraballo
Registro de pago adicional	jue 12/18/08	lun 11/10/08	Carlos M. Pedroso,Hector Prada,Ke
Integración de Trabajador	**vie 1/9/09**	**mar 12/2/08**	
Integración de Trabajador con Persona	jue 12/4/08	mar 12/2/08	Carlos Miguel Pedroso Caraballo
Integración de Trabajador con Incidencias	##########	mar 12/2/08	Carlos Miguel Pedroso Caraballo
Integración de Trabajador con Pagos adicionales	##########	jue 12/4/08	Carlos Miguel Pedroso Caraballo
Integración de Trabajador con el Nomenclador de CP	##########	mar 12/2/08	Carlos Miguel Pedroso Caraballo
Integración de Trabajador con EC (área)	##########	mar 12/2/08	Carlos Miguel Pedroso Caraballo
Integración de Trabajador con EC (cargos)	##########	mar 12/2/08	Carlos Miguel Pedroso Caraballo
Integración de Trabajador con componente Uso del Nomenclador de Cuentas	##########	mié 12/3/08	Carlos Miguel Pedroso Caraballo
Integración de Trabajador con IoC Externo	##########	mar 12/2/08	Carlos Miguel Pedroso Caraballo

Nombre de tarea	Fin	Comienzo	Nombres de los recursos
Incidencias	**vie 12/12/08**	**mié 10/8/08**	
Diseño de Modelo de Datos	lun 12/1/08	mar 11/11/08	Alexei Olivero Márquez
Creación de Modelo de Datos	lun 12/1/08	mar 11/11/08	Alexei Olivero Márquez
Programación del Modelo de Datos	lun 12/1/08	mar 11/11/08	Alexei Olivero Márquez
Diseño de Lógica de Negocio	lun 12/1/08	mar 11/11/08	Alexei Olivero Márquez
Diseño de Interfaz de Usuario	lun 12/1/08	mar 11/11/08	Alexei Olivero Márquez
Validación de Interfaz de Usuario	lun 12/1/08	mar 11/11/08	Alexei Olivero Márquez
Reunión de Implementación	**vie 12/5/08**	**lun 11/10/08**	**Alexei Olivero Márquez**
Análisis	jue 11/13/08	lun 11/10/08	Olga Yarisbel Rojas Grass,Crescencio C
Análisis	vie 12/5/08	lun 11/24/08	Olga Yarisbel Rojas Grass,Crescencio C
Implementación	**lun 12/8/08**	**jue 11/20/08**	
Gestionar tipo de impuesto y contribuciones	##########	jue 11/20/08	Sergio Hernandez
Gestionar Tipo de incidencia	lun 12/8/08	lun 11/24/08	Sergio Hernandez
Gestionar los Reportes de impuestos	lun 12/8/08	lun 11/24/08	Sergio Hernandez
Gestionar los Reportes de las incidencias	lun 12/8/08	lun 11/24/08	Sergio Hernandez
Integración incidencias	**mié 12/10/08**	**mar 12/2/08**	
Integración de Incidencias con el Nomenclador de CP	##########	mar 12/2/08	Sergio Hernandez ,Kenner Sánchéz
Integración de Incidencias con componente Uso del Nomenclador de Cuentas	##########	mié 12/3/08	Sergio Hernandez ,Kenner Sánchéz
Integración de Incidencias con IoC Externo	jue 12/4/08	mié 12/3/08	Sergio Hernandez ,Kenner Sánchéz

Nombre de tarea	Fin	Comienzo	Nombres de los recursos
Nómina	**lun 1/19/09**	**lun 11/17/08**	**Osvaldo Díaz ,Wendy Gracia**
Diseño de Modelo de Datos	lun 12/8/08	lun 11/17/08	Alexei Olivero Márquez
Creación de Modelo de Datos	lun 12/8/08	lun 11/17/08	Alexei Olivero Márquez
Programación del Modelo de Datos	lun 12/8/08	lun 11/17/08	Alexei Olivero Márquez
Diseño de Lógica de Negocio	lun 12/8/08	lun 11/17/08	Alexei Olivero Márquez
Diseño de Interfaz de Usuario	lun 12/8/08	lun 11/17/08	Alexei Olivero Márquez
Validación de Interfaz de Usuario	lun 12/8/08	lun 11/17/08	Alexei Olivero Márquez
Reunión de Implementación	**mar 1/13/09**	**lun 11/17/08**	**Alexei Olivero Márquez**
Análisis 1	vie 12/5/08	lun 11/17/08	Fidel Jimenez Sanzano,Crescencio Camp
Análisis 2	lun 1/12/09	lun 11/24/08	Fidel Jimenez Sanzano,Crescencio Camp
Análisis 3	lun 1/12/09	lun 11/17/08	Fidel Jimenez Sanzano,Crescencio Camp
Análisis 4	mar 1/13/09	vie 11/28/08	Crescencio Campos Hung
Implementación	**lun 1/19/09**	**vie 11/28/08**	
Gestionar tipo de nómina	vie 1/16/09	vie 11/28/08	Javier Limonta,Yoenry Vanega Hech
Gestionar Tipo de ajuste	lun 1/19/09	vie 11/28/08	Javier Limonta,Yoenry Vanega Hech
Gestionar período de pago	vie 1/16/09	vie 11/28/08	Javier Limonta,Yoenry Vanega Hech
Documentos de Ajuste	lun 1/19/09	vie 11/28/08	Javier Limonta,Yoenry Vanega Hech
Procesar Nómina	vie 1/16/09	vie 11/28/08	Javier Limonta,Yoenry Vanega Hech
Revisar Nómina	vie 1/16/09	vie 12/5/08	Javier Limonta,Yoenry Vanega Hech
Procesar nómina de ajuste	mié 1/14/09	vie 11/28/08	Javier Limonta[103%],Yoenry[103%]
Generar comprobante	mar 1/13/09	vie 11/28/08	Javier Limonta,Yoenry Vanega Hech
Mostrar Comprobante de operaciones	mar 1/13/09	vie 11/28/08	Javier Limonta[87%],Yoenry[87%]
Cierre de período	vie 1/16/09	vie 11/28/08	Javier Limonta[33%],Yoenry[33%]
Configurar nómina	sáb 1/10/09	#########	Javier Limonta,Yoenry Vanega Hech
Integración de Nómina con Incidencias	lun 1/12/09	vie 1/9/09	Javier Limonta,Yosniel Ramos Ofar
Integración de Nómina con Trabajador	lun 1/12/09	vie 1/9/09	Javier Limonta,Yosniel Ramos Ofar
Integración de Nómina con componente Uso del Nomenclador de Cuentas	jue 12/4/08	mié 12/3/08	Javier Limonta,Yosniel Ramos Ofar
Integración de Nómina con IoC Externo	lun 1/12/09	vie 1/9/09	Javier Limonta,Yosniel Ramos Ofar
Integración de Nómina con IoC Interno	lun 1/12/09	vie 1/9/09	Javier Limonta,Yosniel Ramos Ofar

Uriser Barbón [54%],Yosniel Ramos Ofarrill[54%],Yainelis Carbonell Toral[54%]

Nombre de tarea	Fin	Comienzo	Nombres de los recursos
Submayores	**lun 1/12/09**	**lun 11/24/08**	
Diseño de Modelo de Datos	sáb 12/13/08	vie 11/28/08	Alexei Olivero Márquez
Creación de Modelo de Datos	sáb 12/13/08	vie 11/28/08	Alexei Olivero Márquez
Programación del Modelo de Datos	sáb 12/13/08	vie 11/28/08	Alexei Olivero Márquez
Diseño de Lógica de Negocio	sáb 12/13/08	vie 11/28/08	Alexei Olivero Márquez
Diseño de Interfaz de Usuario	sáb 12/13/08	vie 11/28/08	Alexei Olivero Márquez
Validación de Interfaz de Usuario	sáb 12/13/08	vie 11/28/08	Alexei Olivero Márquez
Reunión de Implementación	**vie 12/5/08**	**vie 11/28/08**	**Alexei Olivero Márquez**
Análisis	vie 12/5/08	vie 11/28/08	Crescencio Campos Hung [353%]
Implementación	**mar 12/9/08**	**lun 11/24/08**	
Tipo de programa	mar 12/9/08	lun 11/24/08	Uriser Barbón [101%],Yainelis Carb
Tipo de retención	lun 12/8/08	lun 11/24/08	Yainelis Carbonell[54%],Yosniell[54
Gestionar Registro de retenciones	lun 12/8/08	lun 11/24/08	Yosniel[54%],Uriser[54%],Yainelis[5
Apertura y cierre SubMayor de Vacaciones	mar 12/9/08	lun 11/24/08	Uriser Barbón [54%],Yosniel Ramos
Actualizar Submayor de vacaciones	mar 12/9/08	lun 11/24/08	Uriser Barbón [54%],Yosniel Ramos
Actualizar Submayor de retenciones	jue 12/4/08	mar 12/2/08	Uriser Barbón [54%],Yosniel Ramos
Detalles del Submayor de vacaciones	lun 12/8/08	lun 11/24/08	Uriser Barbón [54%],Yosniel Ramos
Detalles del Submayor de retenciones	lun 12/8/08	lun 11/24/08	Uriser Barbón [54%],Yosniel Ramos
Conciliación con contabilidad del Submayor de vacaciones	lun 12/8/08	lun 11/24/08	Uriser Barbón [54%],Yosniel Ramos
Conciliación con contabilidad del submayor de retenciones	jue 12/4/08	mar 12/2/08	Uriser Barbón [54%],Yosniel Ramos
Configuracion del submayor de vacaciones	jue 12/4/08	mar 12/2/08	Uriser Barbón [54%],Yosniel Ramos
Integración de Submayores con Trabajador	lun 1/12/09	vie 1/9/09	Uriser Barbón [54%],Yosniel Ramos
Integración de Submayores con Pago adicional	lun 1/12/09	vie 1/9/09	Uriser Barbón [54%],Yosniel Ramos
Integración de Submayores con Puesto de trabajo	lun 1/12/09	vie 1/9/09	Uriser Barbón [54%],Yosniel Ramos
Integración de Submayores con Nómina	lun 1/12/09	vie 1/9/09	Uriser Barbón [54%],Yosniel Ramos
Integración de Submayores con componente Uso del Nomenclador de Cuentas	lun 1/12/09	vie 1/9/09	Uriser Barbón [54%],Yosniel Ramos