Universidad de las Ciencias Informáticas

Facultad 1

Título: “Propuesta de Arquitectura del Sistema de Registro de Visitante en la UCI”

Trabajo de Diploma para optar por el titulo de Ingeniero en Ciencias Informáticas

Autor: Henry Yordan López Mastrapa

Tutor: Lic. Osmanys Alonso Guerra

Ciudad de La Habana, 10 de julio de 2008
“Año 50 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 1 y a la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente con este trabajo.

Para que así conste, firmo la presente a los ____ días del mes de julio del 2008

_____	_____
Henry Yordan López Mastrapa	Lic. Osmanys Alonso Guerra
[Autor]	[Tutor]

DATOS DEL CONTACTO

Tutor: Lic. Osmanys Alonso Guerra

Licenciado en Ciencias de la Computación.
Instructor Recién Graduado.
Dpto. de Programación Facultad 1

AGRADECIMIENTOS

A la Universidad de las Ciencias Informáticas por haberme formado como profesional.

En especial a mi mamá Gelsy por haberme dado todo lo que he querido en esta vida.

A mi papá Enrique por toda la educación que me ha dado.

A mi abuela Dorilda por quererme tanto.

A mi abuelo Pirimpollo, que aunque ya no soy su nieto preferido, se que me adora igual que yo a él.

A mis abuelos Romérico y Juana por brindarme todo el cariño y respeto.

A mis hermanos El Yasma, Luis Alberto, Orlando y Yanara (la perra).

En especial a mi amigo Yordanis, al cual le agradezco muchísimo.

A mi bebuchina Sissy, que aunque siempre estamos fajados, la quiero y le debo mucho por todo lo bueno que ha hecho por mi.

A María, Michel, Michelito y Duvany por todo estos años compartiendo con ellos.

A mis primos queridos Yahan, Lixan, Taty, Daniel, Laritza, Yadira, Leidys.

A mis tíos queridos Tata, Pancho, El gallo, Raúl, Gogui, Edilma, Aroldo, Luis, Grisel, Adela la peleona, Teté, Iris y Tony, Chocho, Piñita.

A Yamile y Juany.

A mi bisabuelita Carmita.

A toda mi familia en general.

A mis amigos de la farándula Edel, Frank, Willian, El Dave, Yisel, Yadira, Sandra, Yeli, Elsia, Lis, entre otros.

A mis compañeros de aula y de habitación.

A mis amigos que de una forma u otra ya no están a mi lado pero que en momentos determinados me ayudaron a compartir, Chang, Felix, Carlos, Reynaldo.

A Todos mis vecinos que me han apoyado.

A mi querida Olga.

A la familia de mi Hermano Luis.

A Pombo, Yudith.

A Odalis, Nené, Echenique, Luisito.

A mi tutor Osmanys por todo el apoyo que me ha brindado.

A Cobo, Jordenys, Raly por todo en lo que me ayudaron.

A mis amiguitas Annerys, Adith, Jeny, Milene.

En general a todos los que de una forma u otra me han apoyado en toda mi vida.

DEDICATORIA

Le dedico este trabajo por completo a mi mamá Gelsy que es lo que más quiero en este mundo, por toda su confianza en mí, por todo lo que me ha enseñado, por todas las cosas linda con que siempre me sorprende, por siempre hacerme feliz, por solo tenerme a mí y dedicarme todo su cariño, su amor, su ternura, por ser la madre que solo vive para su niñito lindo, por todo esto, gracias mamá, por traerme a este glorioso mundo.

RESUMEN

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Hay que esforzarse para lograr esto y se haga útil en la Universidad de las Ciencias Informáticas, logrando sus propias arquitecturas de desarrollo de los sistemas.

Puesto que existe un control de los visitantes en la Universidad que se realiza de forma manual y no existe un Sistema de Registro de Visitantes, el presente trabajo realiza una propuesta de arquitectura para el Sistema de Registro de Visitantes en la UCI basada en software libre. Durante el desarrollo de este trabajo se analizan las principales tecnologías y metodologías actuales para el desarrollo del mismo, obteniendo como resultado el diseño de la arquitectura para este sistema, logrando módulos flexibles y robustos que cumplan con las necesidades de la Universidad.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Desarrollar aplicaciones Web no ha sido una tarea fácil aunque cada día se obtienen mejores y actualizadas tecnologías que nos simplifican el manejo de varios aspectos del problema. Se necesita de una arquitectura de software que permita comprender el sistema, que sea capaz de organizar el desarrollo, fomentar la reutilización y hacerlo que evolucione de la forma más óptima.

La Universidad de las Ciencias Informáticas (UCI) desde sus inicios se ha dedicado, entre muchas cosas, al desarrollo de Aplicaciones Web para su propio beneficio y así facilitar la seguridad y flexibilidad de muchas de las acciones que en ella se llevan a cabo diariamente en diferentes sectores que integran la Universidad. La seguridad y protección ha sido un elemento clave para conservar los medios que cada vez son más los existente en ella y necesitan de este servicio. Cada día son muchos los visitantes que llegan a la Universidad y no hay un control con gran seguridad para que estos visitantes sean registrados, ya que se hacen de forma manual en estos momentos.

De lo anteriormente expuesto se puede plantear la siguiente **situación problémica**:
No existe un sistema informatizado capaz de lograr incrementar la flexibilidad del registro de visitantes en la UCI y a la vez simplificar la infraestructura tecnológica utilizando servicios web.

Dada toda esta situación surge el siguiente **problema científico**:
¿Qué Arquitectura utilizar en el Sistema de Registro de Visitantes en la UCI sobre software libre para lograr módulos flexibles y robustos?

Teniendo en cuenta el problema planteado se define como **objeto de estudio:**
Arquitectura de aplicaciones informáticas.

Por lo que se especifica el siguiente **campo de acción:**
Arquitectura de aplicaciones web.

Dadas las condiciones se plantea lograr como **objetivo general:**
Definir la arquitectura del Sistema de Registro de Visitante, teniendo en cuenta su desarrollo en software libre.

Según los objetivos planteados para la investigación y los posibles resultados se definen las siguientes **tareas** para lograr su óptimo cumplimiento:

- Realizar un estudio del estado del arte referente al Sistema de Registro de Visitantes.
- Investigar acerca de las herramientas en software libre que podrán ser usadas.
- Definir el marco de trabajo más apropiado para el desarrollo del sistema.
- Definir la nueva arquitectura ajustándose a las indicaciones propuestas por la dirección de Informatización.

El presente trabajo se estructura en tres capítulos:

Capítulo 1. Fundamentación teórica: En este capítulo se dan a conocer los principales concepto de la Arquitectura, se hace un estudio del estado del arte referente a la del sistema propuesto y de las tecnologías, herramientas y metodologías actuales con el fin de proponer las más adecuadas para la solución del problema.

Capítulo 2. Definición de la arquitectura: En este capítulo se realiza la definición de la arquitectura propuesta mediante las tecnologías, metodologías y herramientas adecuadas para el mejor desarrollo del sistema.

Capitulo 3. Descripción de la Arquitectura: En este capítulo se realiza la descripción de la arquitectura propuesta, donde se muestran las vistas arquitectónicas para el mejor entendimiento de la solución.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

En este capitulo se analizan los aspectos teóricos para realizar un estudio sobre las arquitecturas para desarrollar aplicaciones Web sobre software libre, se partirá de los principales conceptos a tener en cuenta para lograr su desarrollo y su entendimiento. Se analizarán y se caracterizarán las herramientas y tecnologías que puedan ser utilizadas en la confección del software.

1.1 Arquitectura de software (AS)

En los inicios de la informática, la programación se consideraba un arte, debido a la dificultad que entrañaba para la mayoría de los habitantes. Como consecuencia del surgimiento y desarrollo se abre un nuevo avance a nivel mundial: la producción de software, software que fueron ganando en complejidad y dando solución debido a la necesidad de resolver problemas más complicados que cada día aumentaban y requerían de una inmediata respuesta. En los años 60 ya se acariciaba el concepto de arquitectura de software en los círculos de investigación ,por ejemplo en 1968, Edsger Dijkstra, de la Universidad Tecnológica de Eindhoven en Holanda propuso que se estableciera una estructuración correcta de los sistemas de software antes de lanzarse a programar, pero con el tiempo se han ido desarrollando metodologías para conseguir esos propósitos. Y a todas estas técnicas se les ha dado en llamar Arquitectura de Software.

Entre las muchas definiciones de Arquitectura de software que se han dado durante los años es digno mencionar los siguientes:

Perry y Wolf en 1992: Una arquitectura de software es un conjunto de elementos arquitectónicos que tienen una determinada forma. Las propiedades restringen la elección de los elementos de la arquitectura mientras que la lógica captura la motivación de la elección de los elementos y la forma.[**¡Error! No se encuentra el origen de la referencia.**]

Garlan y Shaw en 1996: Una arquitectura de software incluye la descripción de elementos a partir de los cuales se construyen los sistemas de software, interacciones entre esos elementos, patrones que guían la composición y restricciones sobre esos patrones. En general, un sistema de software

particular se define en términos de una colección de componentes e interacciones entre dichos componentes. Tal sistema puede ser utilizado como un elemento en sistemas más grandes.[1]

Bass, Clements y Kazman en 1998: Una arquitectura de software de un programa o sistema de computación es la estructura o estructuras del sistema, el cual comprende componentes, las propiedades visibles externas de dichos componentes y las relaciones entre ellos. [3]

Kaiser en 2005: La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. Definición “oficial” de ARQUITECTURA DE SOFTWARE acordada y brindada en el documento de IEEE Std 1471-2000.[4]

La AS tiene una gran importancia debido a que es donde se representa la arquitectura en el desarrollo del software, sin tener en cuenta el tipo de software que vamos a desarrollar, la cual representa las bases óptimas sobre las que se desarrollará el sistema. Brinda soluciones mediante estructuras capaces de soportar todo tipo de problema que pueda presentarse en el desarrollo del trabajo. Asegura que los requerimientos importantes puedan ser implementados y evaluados. Define la interacción y organización entre los diferentes componentes que intervienen. Promueve la reutilización de componentes existentes como librerías de clases, sistemas legados y de aplicaciones de terceros.

La arquitectura de software provee una descripción de alto nivel de los subsistemas que comprenden la arquitectura del sistema. Está atada a comprender cómo las mayores operaciones del negocio son soportadas. Ayuda a la planificación de recursos y la asignación de tareas, debido a que el trabajo de desarrollo puede ser particionado a través de los subsistemas y los esfuerzos de desarrollo individual pueden proceder en paralelo. Cuando los equipos de desarrollo están geográficamente dispersos puede minimizar el número de interacciones requeridas.

1.1.1 Usabilidad y arquitectura del software

A menudo hay que ir más lejos y no basta con tener en cuenta la presentación y la funcionalidad. Sobre todo en sistemas complejos, como pueden ser los entornos distribuidos, los transaccionales, los multicanal y aquéllos en los que puede haber miles de usuarios conectados simultáneamente, hay que

tener en cuenta la usabilidad desde el inicio del diseño del sistema, es decir, desde lo que se denomina momento de Arquitectura de Software.

Es bien sabido por los ingenieros del software que cuanto más tarde se detectan los problemas, más cuesta arreglarlos. Se refiere a cosas como que el usuario pueda visualizar el progreso de sus peticiones, que pueda deshacer acciones, que pueda disponer de un entono multilingüe, que pueda cancelar una operación que lleva mucho tiempo en espera, que pueda reutilizar información que ha introducido anteriormente, y muchas otras cosas.

Si se analiza estos escenarios de interacción, veremos que la causa de que no se puedan implementar es que no se tuvo en cuenta al usuario al inicio del diseño del sistema, es decir, en la Arquitectura del Software.

La usabilidad es muy necesaria que se tenga presente desde el inicio del diseño de un sistema, o sea, desde el momento de la arquitectura de software, pero para que esto funcione de forma efectiva los arquitectos y expertos en usabilidad deben tener un lenguaje común. Por esto los escenarios de usabilidad y los patrones arquitectónicos pueden servir de gran ayuda.

1.1.2 Los productos resultantes de la Arquitectura de Software

El objetivo principal de la Arquitectura del Software es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. Para conseguirlo, la Arquitectura del Software construye abstracciones, materializándolas en forma de diagramas (blueprints) comentados.

No hay estándares en cuanto a la forma y lenguaje a utilizar en estos diagramas. De todas formas, existe un consenso en cuanto a la necesidad de organizar dichas abstracciones en forma de vistas, tal y como se hace al diseñar un edificio. La cantidad y tipos de vistas difieren en función de cada tendencia arquitectónica.

Quizás uno de los modelos más conocidos es el “4+1” de Philippe Kruchten, vinculado al Rational Unified Process (RUP), que define cuatro vistas diferentes:

- **Vista lógica**: describe el modelo de objetos.

- **Vista de proceso**: muestra la concurrencia y sincronía de los procesos.

- **Vista física (despliegue)**: muestra la ubicación del software en el hardware.

- **Vista de desarrollo** (**implementación**): describe la organización del entorno de desarrollo.

- Existe una quinta vista que consiste en una selección de casos de uso o de escenarios que los arquitectos pueden elaborar a partir de las cuatro vistas anteriores.

1.1.3 Modelos o vistas

Toda arquitectura de software debe describir diversos aspectos del software. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Es importante destacar que cada uno de ellos constituye una descripción parcial de una misma arquitectura y es deseable que exista cierto solapamiento entre ellos. Esto es así porque todas las vistas deben ser coherentes entre sí, evidente dado que describen la misma cosa.

Cada paradigma de desarrollo exige diferente número y tipo de vistas o modelos para describir una arquitectura. No obstante, existen al menos tres vistas absolutamente fundamentales en cualquier arquitectura:

- La visión **estática**: describe qué componentes tiene la arquitectura.

- La visión **funcional**: describe qué hace cada componente.

- La visión **dinámica**: describe cómo se comportan los componentes a lo largo del tiempo y como interactúan entre sí.

Las vistas o modelos de una arquitectura pueden expresarse mediante uno o varios lenguajes. El más obvio es el lenguaje natural, pero existen otros lenguajes tales como los diagramas de estado, los diagramas de flujo de datos, entre otros. Estos lenguajes son apropiados únicamente para un modelo o

vista. Afortunadamente existe cierto consenso en adoptar UML (*Unified Modeling Language*, lenguaje unificado de modelado) como lenguaje único para todos los modelos o vistas. Sin embargo, un lenguaje generalista corre el peligro de no ser capaz de describir determinadas restricciones de un sistema de información (o expresarlas de manera incomprensible).

Después de la aparición de la Arquitectura de Software surgen los lenguajes de descripción de arquitectura (ADL). Los ADL se utilizan para satisfacer requisitos descriptivos de alto nivel de abstracción. Con un ADL el arquitecto puede analizar sobre las propiedades del sistema con gran precisión. Algunas de esas propiedades pueden ser, por ejemplo, protocolos de interacción, anchos de bandas y latencia, localización de almacenamiento, entre otros. Estos fueron elementos importantes de la arquitectura que influyeron de manera eficaz en el surgimiento de la tecnología de componentes.

1.2 Experiencias anteriores vinculadas al problema

Durante la investigación se pudo constatar que actualmente en Cuba no existe la documentación de ningún sistema de registro de visitantes desarrollado para una institución. Sin embargo, en el ámbito internacional existen actualmente un sinnúmero de sistemas de registro de visitantes. Aunque la mayoría son soluciones a la medida, existen algunas compañías que se han encargado en desarrollar productos para el registro de visitantes.

A continuación se muestran algunos sistemas en el ámbito internacional:

Quanto Visitantes

Este sistema es un producto desarrollado por la compañía *Bertran Software*[5]. Entre las principales características con las que cuenta este sistema esta que presenta un Asistente de Registro de Visitas que agiliza el proceso de registro de visitas y mantiene información de visitas anteriores agilizando aún más las visitas frecuentes. Se pueden programar visitas para que al llegar el visitante se tenga toda la información de la visita. Se pueden generar credenciales en el momento con la fotografía del visitante. Cuenta con una bitácora de visitas que sustituye eficientemente a las bitácoras manuales ya que se pueden hacer poderosas búsquedas y reportes de las visitas. Crea sofisticados reportes al instante con su módulo *QuantoReports*, además de contar con la capacidad de utilizar reportes generados desde *Crystal Reports* de *Seagate Software*.

Esta es una herramienta poderosa para el registro de visitantes en una empresa. Sin embargo la principal debilidad que tiene este sistema es ser un software privativo y su comercialización es restringida.

Sistema BioVisitas

Este sistema fue desarrollado por la compañía argentina *Sistemas Tecnológicos S.A.* [6] y esta orientado no solo al registro de visitantes sino también al control de acceso de los mismos usando características biométricas como son las huellas digitales.

Entre las principales características y funciones del *Sistema BioVisitas* podemos enumerar las siguientes:

- Permite parametrizar el sistema en cuanto a los accesos controlados existentes.
- Definición de horarios de visitas y duración máxima de las mismas.
- Sistema de alarmas en el momento que finalice el plazo de la visita o bien si las mismas aun no egresaron del edificio en un horario determinado o al momento de finalización de las actividades en el mismo.
- Completos reportes de visitas diarios, con posibilidad de agrupamiento por sector.
- Exportación de reportes a MS-Excel o archivo de texto plano.

Los requerimientos de hardware para el uso de este sistema son mínimos ya que requiere de una PC Pentium II o superior compatible, 128 MB de RAM y 100 MB de espacio disponible de disco duro. Es un software potente y muy seguro pero requiere de herramientas propietarias para su funcionamiento como son el MS-Excel y el SQLServer; a esto se le añade que su comercialización y desarrollo es privativo y que debido a esto no podría ser modificado para adaptarlo a las necesidades de nuestra Universidad.

Sistema de Control de Visitantes

Este software fue desarrollado por la compañía mexicana *Beek Corporativo S.A*. [7] Este sistema tiene un modulo que permite organizar y llevar un control eficiente de los visitantes a las instalaciones, ligando esta información con el empleado y horarios correspondientes a la visita.

Entre las principales características con la que cuenta este sistema está:

- Funciona en forma autónoma o en red, lo cual permite que varias personas puedan estar utilizando el sistema en distintos puntos de acceso, administrando el flujo de información de los visitantes en forma centralizada.

- Emisión de etiqueta de control: Después de registrar al visitante, se podrá entregar una etiqueta con los datos relativos a su visita, lo que le permitirá identificarse durante su estancia en las instalaciones de la empresa.

- Registro de citas: Los empleados pueden capturar previamente las citas de las personas que esperan, facilitando el pre-registro de los visitantes en los puntos de acceso.

- Fotografías: El sistema permite capturar la imagen del visitante, vehículo y el equipo que ingresa a las instalaciones, facilitando su identificación.

- Reportes y Consultas: Puede obtener una amplia cantidad de informes generados por el sistema.

Este es sin duda alguna un sistema orientado al cliente en el cual se mejora la calidad en la atención del visitante, además se incrementa la productividad y la capacidad de atención de visitas. Este software al igual que los demás estudiados es privativo lo cual afecta su reutilización y adquisición para posterior estudio.

1.3 Tendencias, tecnologías y metodologías actuales

1.3.1 Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML), fue originalmente creado por Rational Software a pesar de que actualmente sea atendido por el Object Management Group (OMG). Es un lenguaje que permite visualizar, especificar, construir y documentar, modelos de sistemas de software, incluyendo su estructura y diseño; que capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Es un lenguaje de propósito general para el modelado visual y orientado a objetos, que permite una abstracción del sistema y sus componentes, al mismo tiempo posibilita establecer una serie de requerimientos y estructuras necesarias para plasmar en un sistema de software previo al proceso intensivo de escribir código.

Este lenguaje de modelado permite:

- Modelar sistemas utilizando técnicas orientadas a objetos (OO).
- Especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.
- Documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, entre otros.). Puede conectarse con lenguajes de programación (ingeniería directa e inversa).
- Cubrir todas las vistas necesarias para desarrollar y luego desplegar los sistemas, dado a que es un lenguaje muy expresivo.
- Cubrir las cuestiones relacionadas con el tamaño, propias de los sistemas complejos y críticos.
- Equilibrar expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

1.3.2 Metodologías de desarrollo de software

La rama de la **metodología**, dentro de la ingeniería de software, se encarga de elaborar estrategias de desarrollo de software que promuevan prácticas adoptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo.

No existe una metodología de software universal. Cada equipo de desarrollo escoge la metodología según las características de su proyecto, viendo cual es la que más se acomoda en su aplicación, por lo que es importante determinar el alcance del proyecto antes de escoger la metodología que se va a usar en el desarrollo del mismo.

A continuación algunas de las metodologías de desarrollo que existen, dando una breve explicación sobre las mismas.

Rational Unified Process (RUP)

RUP (Rational Unified Process, en inglés) es un proceso de ingeniería de software, es una guía de cómo usar UML de la forma más efectiva, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.[8]

RUP divide su ciclo de desarrollo en cuatro fases y agrupa sus actividades en 9 flujos de trabajo.

Fases:

Inicio: Esta fase tiene como objetivo determinar la visión del proyecto.

Elaboración: En esta etapa el objetivo es definir la arquitectura del sistema.

Construcción: En esta etapa el objetivo es llegar a obtener un producto listo para su utilización que está documentado y tiene un manual de usuario.

Transición: El objetivo es llegar a obtener el release ya listo para su instalación. Puede implicar reparación de errores.

Flujos de Trabajo:

Modelamiento del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.

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

Análisis y diseño: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

Implementación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

Prueba (Testeo): Busca los defectos a los largo del ciclo de vida.

Instalación: Produce *release* del producto y realiza actividades (empaque, instalación, asistencia a usuarios y demás) para entregar el software a los usuarios finales.

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

Administración de configuración y cambios: Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, entre otros.

Ambiente: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Cada flujo de trabajo tiene como resultado un modelo propuesto por RUP:

- Modelo de Casos de Uso del Negocio.
- Modelo de Objetos del Negocio.
- Modelo de Casos de Uso.
- Modelo de Diseño.
- Modelo de Despliegue.
- Modelo de Datos.
- Modelo de Implementación.
- Modelo de Pruebas.

El modelo de casos de uso, el modelo de diseño, el modelo de despliegue y el modelo de implementación son los modelos más importantes para describir la arquitectura de un sistema teniendo en cuenta que son los que proporcionan el desarrollo de las vistas de arquitectura.

El ciclo de vida de RUP tiene tres características fundamentales:

Guiado por casos de uso. Los casos de uso reflejan las necesidades de los futuros usuarios, las cuales se captan cuando se modela el negocio y se representa a través de los requerimientos. Los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

Centrado en la arquitectura. La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura (casos de uso arquitectónicamente significativo). El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

Iterativo e incremental. RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son miniproyectos.

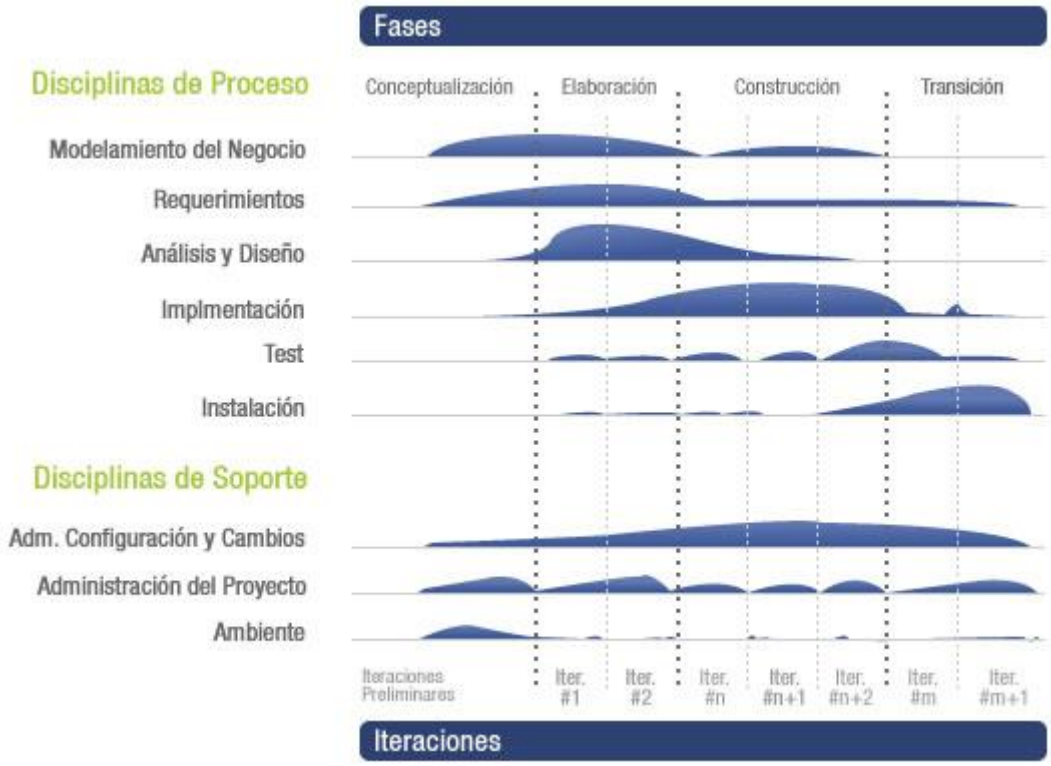

Figura 1. Fases e iteraciones de la Metodología RUP

Programación Extrema (XP)

XP (Extreme Programming, sus siglas en inglés) es una metodología de desarrollo de software, catalogada como una de las más exitosas en la actualidad, utilizadas para proyectos de corto plazo y corto equipo. Consiste en una programación rápida o extrema, donde el usuario final forma parte del equipo, siendo uno de los requisitos para llegar al éxito del proyecto.

Características de XP, la metodología se basa en:

Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.

Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

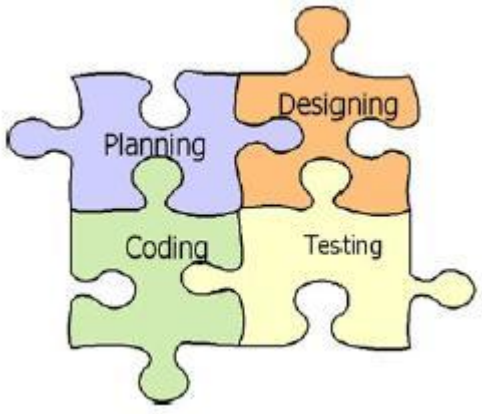

Figura 2. Extreme Programming

Microsoft Solution Framework (MSF)

Microsoft Solution Framework es una metodología de desarrollo de software para la planificación, desarrollo y gestión de proyectos tecnológico.

Se centra en el modelo de procesos y de equipo dejando las elecciones tecnológicas en un segundo plano.

Puede ser adaptada a proyectos de cualquier dimensión y usada para desarrollar soluciones sobre cualquier tecnología.

MSF tiene las siguientes características:

Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.

Escalable: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.

Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.

Tecnología Agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto:

Modelo de Arquitectura del Proyecto: Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.

Modelo de Equipo: Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles. [9]

Modelo de Proceso: Diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo. [9]

Modelo de Gestión del Riesgo: Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar. [9]

Modelo de Diseño del Proceso: Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores. [9]

Modelo de Aplicación: Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores.[9]

Figura 3. Metodología MSF

1.3.3 Estilos Arquitectónicos y Patrones

Desde sus inicios, en la arquitectura de software, se observó que en la práctica del diseño y la implementación ciertas regularidades de configuración aparecían una y otra vez como respuesta a similares demandas. El número de esas formas no parecía ser muy grande. Muy pronto se las llamó estilos, por analogía con el uso del término en arquitectura de edificios. Un estilo describe entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas.

Los estilos arquitectónicos son una generalización y abstracción de los patrones de diseño.

Estilo Arquitectónico caracteriza una familia de sistemas que están relacionados por compartir propiedades estructurales y funcionales.

También puede definirse como la descripción de los tipos componente y de los patrones de interacción entre ellos.

Notar que, a diferencia de los patrones de diseño, la definición apunta a describir sistemas completos y no partes de sistemas.

Un estilo arquitectónico es un conjunto coordinado de restricciones arquitectónicas que restringe los roles o rasgos de los elementos arquitectónicos y las relaciones permitidas entre esos elementos dentro de la arquitectura que se conforma a ese estilo.

Ejemplos de estilos arquitectónicos:

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

Cliente-servidor

Esta arquitectura se encuentra dentro de la clasificación de estilo de llamada y retorno. El cliente y el servidor generalmente están localizados en diferentes sistemas, sin embargo pueden encontrarse en el mismo sistema. El cliente es la entidad que hace la petición por un servicio. El servidor es la entidad que provee el servicio correspondiente a la petición. El servicio debe procurar el resultado, el cual es retornado al cliente. Los clientes pueden ser clasificados como clientes “flacos” y/o “gordos”. Lo clientes gordos típicamente contienen, además de la lógica de presentación, gran parte de la lógica de negocio de la aplicación. Lo clientes flacos manejan usualmente sólo la lógica de presentación lo que adiciona grandes ventajas como permitir que futuros cambios de negocio en la aplicación no afecten al cliente.

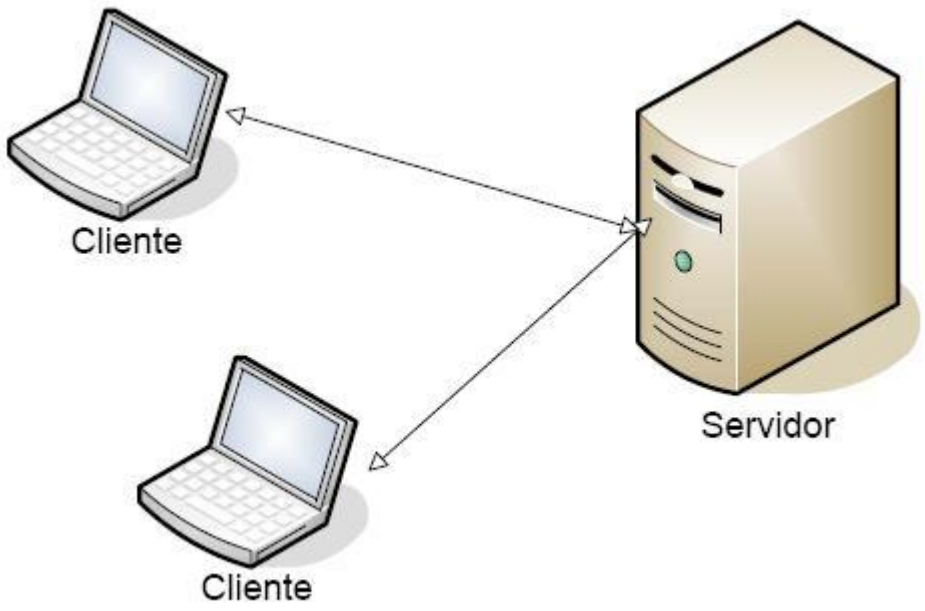

Figura 4. Arquitectura cliente-servidor

Arquitectura orientada a servicios (SOA)

En la actualidad el impacto en las tecnologías de la información (IT) y los procesos de negocios que ha tenido el surgimiento de SOA son ascendentes. Este modelo de arquitectura permite que las organizaciones combinen fluidamente sus activos de tecnologías de información existentes para

desarrollar nuevas aplicaciones, procesos y modelos de negocios internos o externos a la empresa, de una forma más económica rápida y sencilla. Actualmente en el mundo existen grandes empresas que han utilizado SOA para soluciones de mercado con magníficos resultados. Son los casos de Microsoft, IBM, SUN Microsystems, BEA Systems, Software AG, entre otras. SOA como tal empieza a sonar en el mercado en el año 2000 y son muchas las definiciones echas hasta ahora para este modelo. A continuación alguna de ellas:

Según W3C: "Conjunto de componentes que pueden ser invocados, cuyas descripciones de interfaces se pueden publicar y descubrir".

Según CBDI: "Estilo resultante de políticas, prácticas y marcos de trabajo que permiten que la funcionalidad de una aplicación se pueda proveer y consumir como conjuntos de servicios, con una granularidad relevante para el consumidor…"
El objetivo de SOA es aportar para lograr un alto rendimiento desarrollando servicios que puedan ser reutilizados por la organización, estas capacidades permiten que las tecnologías respondan rápidamente a los cambios de los negocios, los escenarios competitivos, las alianzas y las necesidades de los consumidores. SOA permite la creación de aplicaciones compuestas que trabajan a través de una única interfaz, facilitando la comprensión de distintos procesos de negocios. (Simoes)

Una SOA define las siguientes capas de software:

Aplicativa básica: Sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.

Exposición de funcionalidades: Donde las funcionalidades de la capa aplicativa son expuestas en forma de servicios (servicios web).

Integración de servicios: Facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración.

Composición de procesos: Define el proceso en términos del negocio y sus necesidades, y que varía en función del negocio; de entrega, donde los servicios son desplegados a los usuarios finales.

Además al contrario de las arquitecturas orientada a objetos, las SOAs están formadas por servicios de aplicación débilmente acoplados y altamente interoperables.

SOA presenta grandes ventajas en su adopción por parte de las empresas:

Mejora en los tiempos de realización de cambios en procesos.

- Facilidad para evolucionar a modelos de negocios basados en tercerización.

- Facilidad para abordar modelos de negocios basados en colaboración con otros entes (socios, proveedores).

- Poder para reemplazar elementos de la capa aplicativa SOA sin disrupción en el proceso de negocio.
- Proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

Arquitectura en capa

Garlan y Shaw definen el estilo en capas como una organización jerárquica, tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones dadas por la capa inferior. El estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. Permite realizar optimizaciones y refinamientos enfocando los cambios en un solo lugar. Proporciona amplia reutilización dada la división bien definida de responsabilidades. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas. Una especialización muy usada de la arquitectura en capas es la arquitectura de tres capas donde se observan muy bien delimitadas las responsabilidades de cada funcionalidad en la aplicación. Una arquitectura de tres capas, cada capa está muy bien delimitada de las demás. Una capa superior interactúa con una capa inferior mediante interfaces que definen las funcionalidades que la misma debe brindar. Las capas de la aplicación pueden residir tanto

en el mismo nodo físico como en nodos separados. Las tres capas son: presentación (interfaz de usuario, presentación), lógica de negocio (donde se encuentra modelado el negocio, lógica del negocio) y acceso a datos (persistencia, acceso a datos).

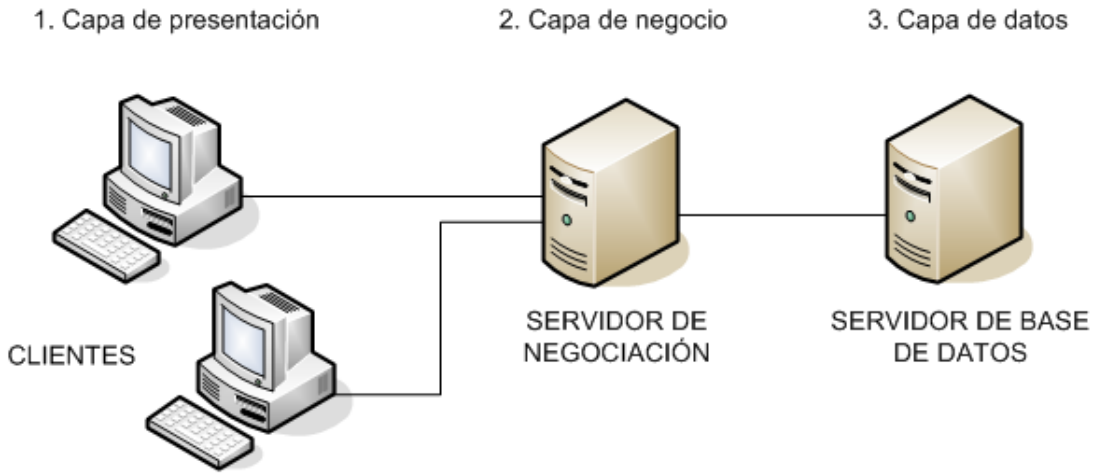

Figura 5. Arquitectura en tres Capas

MVC (Modelo Vista Controlador)

La arquitectura del patrón MVC es un paradigma de programación bien conocido para el desarrollo de aplicaciones con interfaz gráfica. El principal objetivo de este patrón es aislar tanto los datos de la aplicación como el estado (modelo) de la misma, del mecanismo utilizado para representar (vista) dicho estado, así como para modularizar esta vista y modelar la transición entre estados del modelo (controlador).

Las aplicaciones MVC se dividen en tres grandes áreas funcionales:

- **Vista:** La presentación de los datos.
- **Controlador:** El que atenderá las peticiones y componentes para la toma de decisiones de la aplicación.
- **Modelo:** La lógica del negocio o servicio y los datos asociados con la aplicación.

Es una arquitectura preparada para los cambios, que desacopla datos y lógica de negocio de la lógica de presentación, permitiendo la actualización y desarrollo independiente de cada uno de los citados componentes.

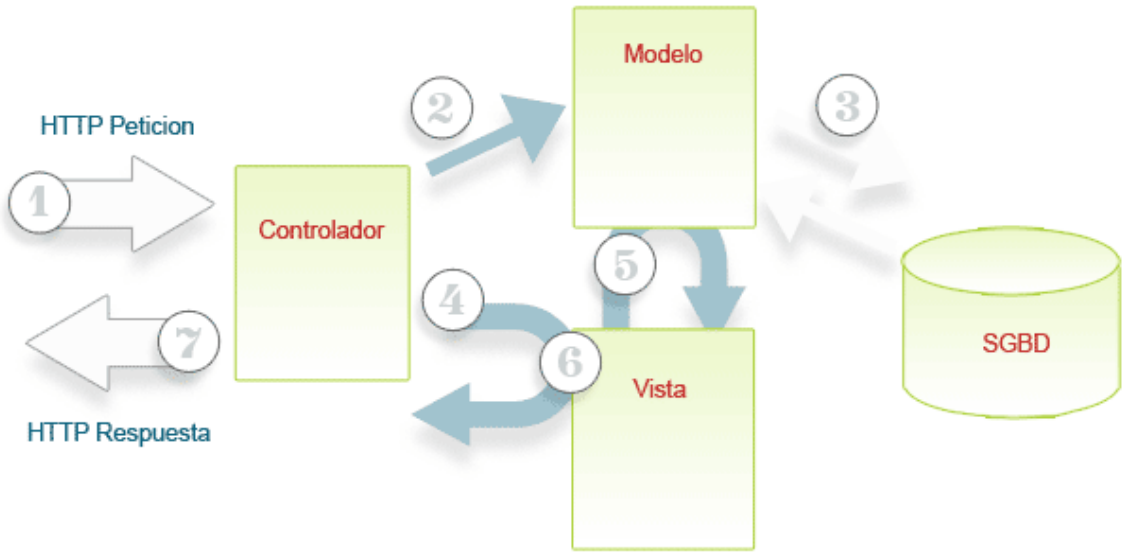

Figura 6. Modelo Vista Controlador Web(MVC-Web)

1.3.4 Lenguajes de Programación Web

En la actualidad, Internet es la vía de comunicación más usada mundialmente, por el sin número de ventajas y funcionalidades que ofrece a los usuarios a partir de las aplicaciones (principalmente web) que soporta.

Existen un conjunto de lenguajes de programación que proporcionan a las aplicaciones web gran interactividad, ya sea del lado del cliente o del servidor. Para software libre los más usados del lado del cliente (encargados de visualizar y comprobar la información en el navegador y los formularios) se encuentran HTML, Java Script y Ajax. Mientras que del lado del servidor (procesan la lógica del negocio) podemos encontrar PHP, JAVA, ASP y Perl.

1.3.4.1 Lenguajes de programación web del lado del cliente

HTML

El lenguaje HTML, cuyas siglas significan Lenguaje de Formato de Documento de Hipertexto (en inglés Hypertext Markup Language*), se* utiliza para crear documentos que muestren una estructura de

hipertexto. Un documento de hipertexto es aquel que contiene información cruzada con otros documentos, lo cual nos permite pasar de un documento al referenciado desde la misma aplicación con la que lo estamos visualizando.

HTML permite, además, crear documentos de tipo multimedia, es decir, que contengan información más allá de la simplemente textual, como pueden ser imágenes, video o sonido. El lenguaje HTML no es el único lenguaje existente para crear documentos hipertexto. Hay otros lenguajes anteriores o posteriores a HTML (SGML, XML), aunque para muchos HTML se ha convertido en el lenguaje estándar para la creación de contenido para Internet.

JavaScript

JavaScript es un lenguaje de programación interpretado, ampliamente utilizado en el mundo del desarrollo web por ser muy versátil y potente, tanto para la realización de pequeñas tareas como para la gestión de complejas aplicaciones, es ejecutado por el navegador que utilizamos para ver las páginas; lo que hace que podemos desarrollar aplicaciones de diversos tipos, desde generadores de HTML, comprobadores de formularios, páginas web dinámicas, intercambiar información entre páginas web en distintas ventanas, manipulación de gráficos, texto, programas que gestionan las capas de una página. Pueden desarrollarse incluso aplicaciones que permitan poder tener capas en una página como si fueran ventanas, y dar la sensación de estar trabajando con una aplicación con interfaz de ventanas. JavaScript comparte muchos elementos con otros lenguajes de alto nivel. Hay que tener en cuenta que este lenguaje es muy semejante a otros como C, Java o PHP, tanto en su formato como en su sintaxis, aunque por supuesto tienen sus propias características definitorias.

CSS

Las hojas de estilo en cascada (Cascading Style Sheets, CSS) representan un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores

Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento. Las ventajas de utilizar CSS (u otro lenguaje de estilo) son:

- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web remoto, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.

- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil, o ser leída por un sintetizador de voz.

1.3.4.2 Lenguajes de programación web del lado del servidor

PHP

Es el acrónimo de Hipertext Preprocesor, lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, de código abierto, ejecutado al lado del servidor y soportado por la mayoría de los servidores Web de hoy en día, por ejemplo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro Server, Caudium, Xitami, OmniHTTPd entre otros, con gran volumen de documentación y con una gran librería de funciones con un espíritu generoso ya que estas funciones son progresivamente construidas por colaboradores desinteresados en nuevas versiones del lenguaje. PHP nos permite embeber sus pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas programados íntegramente en un lenguaje distinto al HTML.

Este lenguaje de programación está preparado para realizar muchos tipos de aplicaciones Web con páginas dinámicas gracias a la extensa librería de funciones con la que está dotado; la cual cubre desde cálculos complejos hasta tratamiento de conexiones de red. Algunas de las más importantes capacidades de PHP es la compatibilidad con las bases de datos más comunes, como MySQL, Oracle, Informix, y ODBC. Esta tecnología aunque sea multiplataforma, ha sido concebida inicialmente para entornos UNIX y es en este sistema operativo donde se pueden aprovechar mejor sus prestaciones.

PHP5

PHP ("Hypertext Preprocessor") es sencillo, de sintaxis cómoda y similar a la de otros lenguajes como C o C++, es rápido a pesar de ser interpretado, es multiplataforma y dispone de una gran cantidad de librerías y se le pueden agregar extensiones fácilmente. Es un lenguaje basado en herramientas con licencia de software libre, es decir, no hay que pagar ni licencias, ni estamos limitados en su distribución y se puede ampliar con nuevas funcionalidades si se desea. Es ideal tanto para el que comienza a desarrollar aplicaciones Web como para el desarrollador experimentado. Se puede utilizar como módulo de Apache, lo que lo hace extremadamente veloz. Por estar completamente escrito en C, se ejecuta rápidamente utilizando poca memoria.

Puede ser compilado y ejecutado en diversas plataformas, incluyendo diferentes versiones. Como en todos los sistemas se utiliza el mismo código base, los scripts pueden ser ejecutados de manera independiente al sistema operativo. Se puede ejecutar bajo Apache, IIS, AOLServer, Roxen y THTTPD. Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL, y otros muchos.

PHP5 hace un cambio en el manejo de los objetos a diferencia de PHP4, o sea, PHP4 trata los objetos igual que otros tipos de datos básicos como los enteros o los arreglos, de modo que cuando se realizan operaciones sobre un objeto el mismo es copiado íntegramente, mientras que en PHP5 las variables que nombran objetos son en realidad referencias. Otro de los elementos significativos propio de PHP5 es que incluye modificadores de control de acceso para implementar el encapsulamiento.

JSP

JSP es un acrónimo de Java Server Pages. Fue creado por la compañía Sun Microsystems y utiliza como lenguaje de programación Java con paradigma orientado a objeto. Es multiplataforma lo que le da gran flexibilidad y seguridad. Es un lenguaje desarrollado para aplicaciones grandes. Presenta una estructura que permite separar la lógica de presentación en páginas JSP y el código o lógica del negocio en clases Java por lo que se considera un lenguaje avanzado para las paginas dinámicas en el servidor lo que permite mayor seguridad de los datos. Posee un grupo de marcos de trabajo que facilitan el trabajo. Las aplicaciones se pueden ejecutar en cualquier ambiente, independientemente del sistema operativo que se utilice, puesto a que es soportado por una máquina virtual que es la encargada de compilar todo el código.

1.3.5 Ambiente de desarrollo

El ambiente de desarrollo (Development Environment) es algo imprescindible en la producción de software. Es donde se definen el conjunto de herramientas y tecnologías (marcos de trabajo), versiones a usar y su integración, que intervienen en un proceso de desarrollo de software.

A continuación se presentan un conjunto de herramientas utilizadas en el desarrollo de este trabajo; como son: tres ambientes de desarrollo integrado (IDE) y sus plugins, un contenedor web, un control de versiones, dos gestores de base de datos, herramientas de modelado y los marcos de trabajo. Se exponen sus características y ventajas para poder tener un mayor conocimiento de sus prestaciones y valorar su utilidad.

Eclipse

Metafóricamente, Eclipse es como una tienda de software libre para herreros, donde no solamente se hacen productos, sino que además se hacen las herramientas para hacer los productos. Es un proyecto de desarrollo de software open-source (código abierto), que está dividido en tres partes: el proyecto Eclipse Proyect, Eclipse Tools, y Eclipse Technology Project.

El Eclipse Project está subdividido a su vez en tres sub-proyectos que son la propia plataforma, JDT (Java Development Tool) y PDE (Plugin Development Enviroment). Mediante este IDE se pueden crear

diversas aplicaciones como pueden ser, sitios web, programas Java, PHP, C++ y Enterprise Java Beans. Su principal aplicación es JDT que es la herramienta para crear aplicaciones en Java, además como su plataforma esta construida en base a plugins esto nos permite que otras aplicaciones pueden ser integradas a eclipse en forma de plugins y los mismos son reconocidos automáticamente por el IDE al iniciarse. Su interfaz de usuario esta compuesta de un conjunto de vistas, editores y perspectivas. Los editores permiten crear, modificar y salvar objetos, las vistas proveen información acerca de los objetos con los que se están trabajando y las perspectivas proveen distintas formas de organización del proyecto.

Eclipse es administrado y dirigido por un consorcio de compañías de desarrollo de software con un interés comercial en promover Eclipse como plataforma compartida para herramientas de desarrollo de software.

Ventajas

- Soporta la construcción de una variedad de herramientas para el desarrollo de aplicaciones.
- Es una herramienta de código abierto.
- Soporta el desarrollo de aplicaciones basadas en GUI y non-GUI.
- Corre en una gran cantidad de sistemas operativos incluyendo Windows y Linux.
- Soporta herramientas que manipulan diferentes tipos de archivos como por ejemplo Java, C, PHP, C++, EJB, HTML, GIF.
- Mediante JDT facilita la creación de aplicaciones programadas en Java.
- Provee a los desarrolladores, herramientas que facilitan la creación de plugins.

Desventajas

- Si bien Eclipse es multiplataforma, los plugins no tienen por qué serlo.

- Al ser una herramienta de código abierto, se desarrollan plugins que no tienen todas las funcionalidades que tienen en otras herramientas comerciales.

- Existen plugins que sólo corren en una plataforma, que aún no han sido desarrollados para más de una.

ZendStudio

ZendStudio divide sus funcionalidades en dos partes: la del cliente y la del servidor las cuales se instalan por separado. La parte del cliente contiene la interfaz de edición y la ayuda y además permite hacer depuraciones simples de scripts. Para disfrutar de toda la potencia de la herramienta de depuración se hace necesaria la parte del servidor que instala Apache y el módulo PHP o los configura para trabajar juntos en depuración en caso de que ya estén instalados.

El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. El editor, la parte del programa que permite escribir los scripts, es muy útil para la programación PHP. A la hora de escribir brinda gran ayuda puesto a que permite:

- Editar varios archivos y moverse fácilmente entre ellos.

- Marcar a qué elementos corresponden los inicios y cierres de las

- Etiquetas, paréntesis o llaves.

- Moverse al principio o al final de una función.

- Identificar automáticamente el código.

En la interfaz encontramos un explorador de archivos, una ventana de depuración, otra para mostrar el código de las páginas y los menús. Además coloca puntos de parada en los scripts y realiza las acciones típicas de depuración.

Dispone de una herramienta muy interesante de debug o depuración que nos permite ejecutar páginas y conocer en todo momento el contenido de las variables de la aplicación y las variables del entorno como las cookies, las recibidas por formulario o en la sesión.

Plugins

Un plugin es una aplicación que interactúa con otra para agregarle una funcionalidad específica y es ejecutada por la aplicación principal. En el caso particular de Eclipse no son más que un conjunto de clases que permiten hacerlo más extensible.

PDT

Es un plugin para Eclipse de código abierto el cual su objetivo es desarrollar y brindar soporte a los proyectos con tecnología PHP, es fácil de usar e intuitivo e integrable con Web Tools de Eclipse, posee asistente y autocompletado, presenta soporte para el debug incremental del código PHP, presenta una integración con el modulo del proyecto de eclipse lo cual posibilita inspeccionar el uso de las vistas del contorno del fichero y del proyecto, así como la nueva vista PHP Explorer. También contiene un extenso marco de trabajo y APIs que les permite con más facilidad a los desarrolladores extender este plugin para crear nuevas e interesantes herramientas orientadas al desarrollo PHP. Presenta una distribuido mediante un proyecto de código abierto con una extensa comunidad de desarrollador.

PDT nos permite utilizar Eclipse para integrar un gran número de herramientas con el objetivo de obtener un solo entorno de desarrollo, lo que nos posibilitará entregar servicios con gran calidad en menos tiempo.

Entre las diferentes características que ofrece PDT a Eclipse podemos enumerar las siguientes:

- Soporte de las versiones 4 y 5 de PHP indistintamente, ya sea bien de forma genérica a todos los proyectos que se generen o bien de forma individual a cada uno con previa especificación en las propiedades del proyecto en cuestión.

- Soporte completo del sistema de documentación PHPDoc, como característica clave la ayuda contextual a la hora de editar la documentación.

- Gestión y exploración de todas las clases generadas a lo largo de la edición del código o bien que se hayan importado de otra librería de PHP, con eso he de indicar que tales clases las toma Eclipse y pueden ser usadas en todo el proyecto como si fuera parte de la librería estándar de PHP.

- Informe de todos los fallos de sintaxis cometidos mientras se edita el código, aunque podemos modificar este comportamiento para que sea un poco más o menos estricto, pero este aspecto es mejor dejárselo al intérprete PHP.

- Por último comentar que PDT posee un formateado de código fuente, es decir, hay ocasiones en las cuales tenemos el código que no es nada legible, pues posee esta utilidad que nos facilitará un poco la vida formateando por nosotros el código para mayor legibilidad.

Subclipse

Es un plugin para Eclipse que adiciona integración para el control de versiones (Subversion, específicamente), permitiendo operaciones de sincronización, actualización, entre otras. Permite bloqueos a recursos para que otros usuarios no puedan modificarlo. Dispone de una vista de comparación entra el recurso local y remoto en caso que exista conflicto entre la versión del recurso local con el remoto. Muestra una vista del historial de versiones de los recursos con un conjunto de atributos de las acciones realizadas sobre el recurso.

Soporta conectarse a varios repositorios de control de versiones al mismo tiempo, permitiendo hacer operaciones sobre el repositorio directamente.

Es un plugin muy útil para el desarrollo colaborativo, en el que intervienen un conjunto de desarrolladores trabajando sobre el mismo proyecto, poniendo a disposición del equipo de desarrollo facilidades para el trabajo en equipo.

Hibernate Tools

Hibernate Tools es un conjunto de herramientas enteramente para trabajar con el marco de trabajo Hibernate, implementado como una suite integrada de plugins para Eclipse.

Este plugin tiene las siguientes características disponibles:

- **Editor de mapeos:** Es un editor para los archivos de mapeos XML de Hibernate, soportando auto completamiento y sintaxis resaltada. Soporta incluso auto completamiento semántico para nombres de clases, propiedades, tablas y columnas.

- **Consola:** La perspectiva de consola de Hibernate permite configurar conexiones a base de datos, provee visualización de clases y sus relaciones. Admite además ejecutar preguntas en formato del lenguaje de preguntas de Hibernate (HQL) interactivamente contra la base de datos y mostrar los resultados.

- **Ingeniería inversa:** Es su característica más importante, permitiendo generar las clases del modelo de dominio y los archivos de mapeos de Hibernate, los EJB3 entity beans anotados, documentación en formato HTML, a partir de una base de datos.

- **Asistentes:** Presenta asistentes como: generador de archivos de configuración rápidamente, configuración de la consola, entre otros.

- **Tareas de Ant:** El Hibernate Tools incluye una tarea de Ant que permite ejecutar la generación de esquemas, mapeos o código Java como parte de su construcción.

1.3.6 Contenedor Web

Los servidores de aplicaciones se crearon con el fin de gestionar las peticiones del usuario y devolverle a los mismos recursos a través de un protocolo de comunicación (HTTP, Hypertext Transfer Protocol, protocolo de transmisión del hipertexto, el protocolo que sirve para incursionar en los sitios de WWW en Internet). Sin embargo en el mundo de JEE surge un término muy común, contenedor Web, el cual además de interceptar solicitudes enviadas en los protocolos: HTTP, HTTPS, FTP, y otros, es

esencialmente un período de ejecución Java que proporciona una implementación del API Servlet7 Java y facilidades para las páginas servidoras de Java o JavaServer Pages (JSP). Además es responsable de inicializar, invocar, y gestionar el ciclo de vida de los servlets Java y las páginas JSP.

Apache

El servidor HTTP Apache es un software (libre) servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf eligió ese nombre porque quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, *a patchy server* (un servidor "parcheado").

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años, pero sigue siendo muy usado.

Apache Tomcat

Es un contenedor de Servlet usado en la implementación de referencia oficial para las tecnologías Java Servlet y JavaServer Pages. Es desarrollado en un ambiente participativo y abierto.

Apache Tomcat es usado en numerosas aplicaciones Web de gran escala y críticas en diversas industrias y organizaciones que se referencia en su sitio oficial.

La versión 5.x es implementada a partir de las especificaciones Servlet 2.4 y JSP 2.0 y cuenta con un mecanismo de recolección de basura perfeccionado, basado en su reducción. Posee una capa envolvente nativa para Windows y Unix para la integración de las plataformas.

1.3.7 Control de versiones

Control de versiones es la gestión de versiones (revisiones) de todos los elementos de configuración que forman la línea base de un producto o una configuración del mismo. Los sistemas para el control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado junto a las posibles especializaciones realizadas para algún cliente específico. Sin embargo, los mismos conceptos son aplicables a otros ámbitos y no sólo para código fuente sino para documentos, imágenes, archivos xml, xsd, dtd, jsp, html.

Un sistema de control de versiones debe proporcionar: Un mecanismo de almacenaje de cada uno de los recursos que deba gestionarse (archivos de texto, imágenes, documentación), posibilidad de modificar, mover, borrar cada uno de los elementos, un histórico de las acciones realizadas con cada elemento pudiendo volver a un estado anterior dentro de ese historial.

Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión, por ejemplo, Subversion.

Subversion

Es un controlador de versiones empleado en la administración de archivos utilizados en el desarrollo de software; es uno de los controladores más utilizados en proyectos de software libre.

Características:

- Fuerte integración con Apples; lo cual permite definir controles de acceso avanzados y navegación vía web para consultar el deposito de archivos.

- En las copias ligeras sobre ramificaciones este controlador de versiones independientemente del número de ramificaciones creadas mantiene un árbol diferencial de cambios, minimizando así el espacio consumido en el depósito.

- Contempla y corrige con éxito la transparencia al eliminar y cambiar nombres de archivos.

- En las copias diferenciales de archivos binarios este controlador de versiones es capaz de mantener un control diferencial sobre cualquier archivo binario del depósito así reduciendo el consumo de espacio.

1.3.8 Gestores de bases datos

Un Sistema Gestor de base de datos (SGBD) es un tipo de software muy específico o conjuntos de ellos que permite manejar de manera clara, sencilla y ordenada altos volúmenes de datos. Actualmente existe una amplia gama de SGBD con características propias, no obstante todos deben tener en cuenta los siguientes aspectos de manera general: abstracción la información, la independencia de los datos, redundancia mínima, consistencia en los datos, seguridad, integridad, respaldo y recuperación, tiempo de respuesta y control de concurrencia.

Existen SGBD muy potentes tanto en la clasificación de software propietario como software libre. Como propietarios podemos encontrar Oracle y Microsoft SQL Server que lideran el mercado por sus altas prestaciones dado muchos años de experiencia mientras que como opción libre se encuentra, entre otros, MySQL, gestor muy utilizado en la web por su simplicidad de uso, y PostgreSQL que si bien no soporta alguno de los aspectos más avanzados si representa una muy opción muy confiable y comprometedora.

DB4O

Se trata de una base de datos que está específicamente diseñado para proporcionar persistencia a los programas desarrollados orientados a objetos. Persistencia de objetos es la capacidad de guardar los objetos en un sistema a fin de que existan incluso después de que la aplicación que los creó deje de usarlos. Este motor de base de datos presenta las siguientes características:

- **Consumo mínimo de recursos:** DB4O está diseñado para ser embebido en clientes u otros componentes de software, de manera totalmente invisible para el usuario final. Es por eso que viene como una librería fácil de incorporar, con un tamaño que ronda los 400 Kb. Como el motor corre en el mismo proceso de la aplicación, el usuario cuenta con control completo sobre la administración de memoria, y puede realizar procesos de profiling y debugging del desempeño sobre todo el sistema. Si la aplicación está corriendo, la base también lo está, sin excepción. Y aún más importante, DB4O es extremadamente flexible a la hora de actualizar una base existente con un modelo de objetos que ha cambiado. Siempre asume que no hay un administrador de base de datos y, por lo tanto, permite a la aplicación cambiar del modelo viejo al modelo nuevo de modo transparente.

- **Alto rendimiento:** El rendimiento de DB4O es equiparable al de los mejores sistemas de base de datos tradicionales.

- **Fácil implementación:** Sólo hay que agregar la librería única de DB4O (.jar o .dll) al entorno de desarrollo, abrir el archivo de base de datos y almacenar cualquier objeto (sin importar su complejidad) con una sola línea de código.

- **Portabilidad:** Corre de manera embebida y nativa en plataformas orientadas a objetos. Se pueden desarrollar aplicaciones para desplegar en varias plataformas (por ejemplo, en PDAs) o en combinaciones heterogéneas de clientes Windows y servidores Java.

- **Confiabilidad:** Finalmente, DB4O soporta todas las propiedades ACID. Múltiples usuarios simultáneos de una base DB4O son efectivamente aislados, y sus operaciones son serializadas de forma transparente por la librería. Las transacciones se terminan con los métodos commit() y rollback() de la clase ObjectContainer en caso de que el sistema se caiga durante una actualización de la base de datos, cuando el ObjectContainer de DB4O es reabierto, se completan de forma correcta todas las transacciones interrumpidas.

Postgre SQL

Es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. Fue el pionero en muchos de los conceptos existentes en

el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. Incluye características de la orientación a objetos, como la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

Ventajas:

- DBMS Objeto-Relacional. Aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son las consultas SQL declarativas, el control de concurrencia multi-versión, el soporte multi-usuario, las transacciones, optimización de consultas, la herencia, los arreglos entre otros.

- Cliente/Servidor. Usa una arquitectura proceso-por-usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

- Altamente Extensible. Soporta los tipos de datos base, así como: fechas, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits. Además operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.

- Soporte SQL Comprensivo. Soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

- Integridad Referencial. Es utilizada para garantizar la validez de los datos de la base de datos.

- Lenguajes Procedurales. Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Además tiene habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

- MVCC (Multi-Version Concurrency Control) Control de Concurrencia Multi-Versión. Es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios, es decir permite la lectura sin que sea bloqueada por los que escriben que están actualizando registros.

- Write Ahead Logging (WAL) Esta característica incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en caso de que no existan las condiciones para la conexión con la base de datos, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos desde el punto en que se quedó.

- Es un gestor magnífico bajo licencia Berkeley Software Distribution (BSD), que posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios Web que posean alrededor de 500.000 peticiones por día. Además por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM.

Desventajas:

- Consume bastantes recursos y carga con mucha facilidad el sistema.

- Velocidad de respuesta un poco deficiente al gestionar bases de datos relativamente pequeñas, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes.

Oracle

Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés Relational Data Base Management System), fabricado por Oracle Corporation. Se considera uno de los sistemas de bases de datos más completos. Es un sistema gestor de base de datos robusto, tiene muchas características que nos garantizan la seguridad e integridad de los datos; que las transacciones se ejecuten de forma correcta, sin causar inconsistencias; ayuda a administrar y almacenar grandes volúmenes de datos; estabilidad, escalabilidad y es multiplataforma.

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia de gestores de bases de datos comerciales y de la oferta de otros

con licencia Software Libre como PostgreSQL. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo Linux.

1.3.9 Herramientas de modelado

Los medios con los que siempre se ha realizado el intercambio de información de diseño e ideas usando la notación UML han sido populares: pizarras, cuadernos y trozos de papel. Pero UML se utiliza mejor a través de una herramienta de modelado, la cual puede ser usada para capturar, guardar, rechazar, integrar automáticamente información, y diseño de documentación. (Modelado de Sistemas con UML 2002).

Una característica que UML brinda para beneficiar a los modeladores es escoger una herramienta de modelado. Tiempos atrás, el modelador primero tenía que seleccionar una notación de metodología, y después estaba limitado a seleccionar una herramienta que la soportara. Ahora con UML como estándar, la elección de notación ya se ha hecho para el modelador. Y con todas las herramientas de modelado soportando UML, el modelador puede seleccionar la herramienta basada en las áreas claves de funcionalidad soportadas que permiten resolver los problemas y documentar las soluciones. (Modelado de Sistemas con UML 2002).

Como una buena caja de herramientas, una buena herramienta de modelado ofrece todas las herramientas necesarias para conseguir hacer eficientemente varios trabajos, sin dejarte nunca sin la herramienta correcta.

Las herramientas de modelado deberían soportar las siguientes funcionalidades:

- Soporte para toda la notación y semántica de UML.

- Soporte para una cantidad considerable de técnicas de modelado y diagramas para complementar UML, incluyendo tarjetas CRC, modelado de datos, diagramas de flujo, y diseño de pantallas de usuario. Posibilidad de reutilizar información obtenida por otras técnicas todavía usadas, como modelado tradicional de procesos.

- Facilitar la captura de información en un repositorio subyacente permitiendo la reutilización entre diagramas.

- Posibilidad de personalizar las propiedades de definición de elementos subyacentes de modelos UML.

- Permitir a varios equipos de analistas trabajar en los mismos datos a la vez.

- Posibilidad de capturar los requisitos, asociarlos con elementos de modelado que los satisfagan y localizar cómo han sido satisfechos los requisitos en cada uno de los pasos del desarrollo.

- Posibilitar la creación de informes y documentación personalizados en tus diseños, y la salida de estos informes en varios formatos, incluyendo HTML para la distribución en la Internet o Intranet local.

- Posibilidad para generar y realizar ingeniería inversa (por ejemplo C++, Java) para facilitar el análisis y diseño interactivo, para volver a usar código o librerías de clase existentes, y para documentar el código.

Visual Paradigm Suite

Visual Paradigm Suite es un conjunto de herramientas de modelado que permiten realizar el modelado dentro del proceso de desarrollo de software.

A continuación se describen las principales herramientas presentes dentro de esta suite:

Visual Paradigm for UML Enterprise Edition

Es una herramienta de modelado diseñada para un gran número de usuarios, incluyendo ingenieros de sistemas, analistas de sistemas, analistas de negocio, arquitectos de sistemas. Además se integra con IDEs (Eclipse, JBuilder, NetBeans, IntelliJ IDEA, JDeveloper and WebLogic Workshop) para soportar la fase de implementación de desarrollo de software. La transición desde el análisis al diseño y después a la implementación es cuidadosamente integrada dentro de la herramienta CASE, de esta manera se

reduce significativamente el esfuerzo en todas las etapas del ciclo de vida del desarrollo del software. Incluye además un conjunto de herramientas que soportan Object-Relational Mapping (ORM), como Hibernate, lo cual incluye generación completamente orientada a objetos, listo para usar librerías para obtener y modificar registros de base de datos para una gran variedad de gestores de base de datos, y la sincronización entre los diagramas de clases y los diagramas de entidad-relación (ERD).

Presenta además generación de código e ingeniería inversa del código. Permite generar los EJB y así mismo los descriptores de despliegue para varios servidores de aplicación. Distinto a muchas herramientas de modelado pueden extenderse sus diagramas hechos desde el Visio y de Rational Rose. Soporta la última versión de UML 2.0.

Visual Paradigm Smart Development Environment (SDE) Enterprise Edition

Visual Paradigm SDE ofrece integración de la herramienta de modelado UML con los IDEs (Visual Studio®, Eclipse/WebSphere®, Borland JBuilder®, NetBeans/Sun™ ONE, IntelliJ IDEA™, Oracle JDeveloper, BEA WebLogic Workshop™). Visual Paradigm SDE se incrusta él mismo dentro de tu IDE favorito para proveer ambiente de desarrollo y modelado unificado. Dando todas las prestaciones y servicios que brinda el Visual Paradigm para UML Enterprise Edition.

Visual Paradigm DB Visual Architect

DB Visual Architect es un simple y fácil de usar ambiente de Object-Relational Mapping (ORM), como Hibernate, el cual actúa como un puente entre el modelo de objetos, el modelo de datos y el modelo relacional. Este ayuda a los desarrolladores a construir aplicaciones de base de datos de alta calidad, mucho más rápido, mejores y barato. Disminuye el costo de desarrollo a un 75 % por automatizar el proceso de mapeo entre los objetos de java y las tablas de la base de datos visualmente y a través de diagramas. Los desarrolladores ahorran enormes cantidades de tiempo usando DB Visual Architect para manipular el código de SQL y JDBC transparentemente con (POJO), como DB Visual Architect está habilitado con ingeniería inversa de una base de datos relacional a un diagrama de clases y un diagrama entidad-relación (ERD) y viceversa (crear una base de datos relacional de modelos de relación de entidad (ERD) o de modelos de clases.

Rational Rose

Rational Rose es la herramienta CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. El navegador UML de Rational Rose nos permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de casos de uso, vista lógica, vista de componentes y vista de despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad.

Características Rational Rose:

- Mantiene la consistencia de los modelos del sistema software.
- Generación Documentación automáticamente.
- Ingeniería Inversa (crear modelo a partir código).
- Generación de Código a partir de los Modelos.
- Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos.
- Capacidad de análisis de calidad de código.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Capacidad de crear definiciones de tipo de documento XML para el uso en la aplicación.

- Integración con otras herramientas de desarrollo de Rational.

- Publicación web y generación de informes para optimizar la comunicación dentro del equipo.

ER/Studio

ER/Studio es una herramienta de modelado para diseñar bases de datos. Ayuda a descubrir, documentar y reutilizar los datos activos. Con un soporte de ida y vuelta de bases de datos, los arquitectos de datos tienen el poder para analizar a donde las fuentes de datos existentes tan bien como el diseño e implementación de bases de datos de alta calidad.

Esta herramienta ofrece las siguientes características: Ambiente de diseño orientado al modelo, soporte del ciclo de vida de bases de datos completas, administración del modelo empresarial, soporte para integración y almacenes de datos o data warehouse, y diseño de base datos con calidad.

1.3.10 Marcos de Trabajo

Un marco de trabajo es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un marco de trabajo puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

Symfony

Es un completo marco de trabajo desarrollado completamente con PHP5 diseñado para optimizar el desarrollo de las aplicaciones Web, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja, automatiza las tareas más comunes permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Compatible con la mayoría de los gestores de base de datos (MySQL, PostgreSQL, Oracle y SQL Server de Microsoft) y

con la mayoría de las plataformas, fácil de instalar, sigue la mayoría de las mejores practicas y patrones de diseño para la Web, sencillo de usar y suficientemente flexible para adaptarse a los casos mas complejos. Permite su integración con librerías desarrolladas por terceros, con código fácil de entender y leer.

Utilizado fundamentalmente para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

Características principales:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de convenir en vez de configurar, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa.
- Código fácil de leer que incluye comentarios y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

CakePHP

CakePHP es un marco de trabajo de desarrollo Web que permite la creación de aplicaciones de forma rápida. Este marco de trabajo implementa el conocido patrón de diseño MVC (acrónimo de Modelo-Vista-Controlador). Dentro de sus características está que es compatible con PHP4 y PHP5, además es expandible sin la modificación de ninguno de sus archivos, pues se pueden crear componentes reutilizables. CakePHP es un marco de trabajo gratis y de código abierto. Contiene un conjunto de librerías y clases que permite trabajar de una forma estructurada y rápida sin una gran pérdida de flexibilidad además, también hay que destacar su activa y colaborativa comunidad, que no se limita a su página Web. Otro de los aspectos que se tuvo en cuenta es que CakePHP está avalado por la Dirección de Informatización en cuanto al uso de marco de trabajo para el desarrollo de aplicaciones Web.

Hibernate

Hibernate es un potente marco de trabajo de mapeo objeto/relacional y servicio de consultas para Java. Es la solución ORM (Object-Relational Mapping) más popular en el mundo Java. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada podremos generar BBDD en cualquiera de los entornos soportados: Oracle, DB2, MySQL.

Sus principales características son:

- Permite expresar consultas utilizando SQL nativo o consultas basadas en criterios.

- Soporta todos los sistemas gestores de bases de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones J2EE y contenedores web, y por supuesto también puede utilizarse en aplicaciones de escritorio.

- Soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el marco de trabajo de colecciones de Java.

- Soporte para modelos de objetos con una granularidad muy fina Permite una gran variedad de mapeos para colecciones y objetos dependientes.

- Provee un sistema de caché de dos niveles y puede ser usado en un clúster. Permite inicialización perezosa (lazy) de objetos y colecciones.

- Proporciona el lenguaje HQL en cual provee una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.

- Presenta un potente mecanismo de transacciones de aplicación llegando incluso a permitir la interacciones largas (aquellas que requieren la interacción con el usuario durante su ejecución).

Soporta los diversos tipos de generación de identificadores que proporcionan los sistemas gestores de bases de datos así como generación independiente de la base de datos, incluyendo identificadores asignados por la aplicación o claves compuestas.

1.4 Conclusiones

En el presente capítulo se realiza un estudio de los sistemas existentes tanto a nivel nacional como internacional, los cuales presentan arquitecturas similares enfocadas al campo de acción, aportando un gran conocimiento y así colaborar de forma óptima con la arquitectura a escoger. Además se analizan las diferentes tendencias, tecnologías y metodologías actuales que podrían dar solución al problema planteado.

CAPÍTULO 2. DEFINICIÓN DE LA ARQUITECTURA

Introducción

En el presente capítulo se establece la descripción del sistema propuesto. Se justifican y detallan los lenguajes, herramientas, metodologías y tecnologías que serán utilizadas en para la elaboración del presente trabajo. Se especifican los patrones arquitectónicos y de diseño que permitirán el desarrollo del futuro sistema propuesto.

2.1 Descripción del Sistema Propuesto

Actualmente tanto en nuestro país como en la UCI no existe un sistema de registro de visitante que gestione los procesos fundamentales que se realizan en esta actividad. En este trabajo se desarrolla la propuesta de la arquitectura del Sistema de Registro de Visitantes para la UCI, el cual cuenta con dos módulos: el servicio Web “Gestor de Visitantes” y la aplicación Web “Registro de Visitantes”.

Servicio Web “Gestor de Visitantes”: este servicio Web tendrá como objetivo principal gestionar toda la información relacionadas con los visitantes en la Universidad y brindar todos los servicios necesarios a otros sistemas internos que lo necesiten como son el Sistema de Control de Acceso y el Registro de Visitantes. El Sistema de Control de Acceso recibe del servicio Web la información de las visitas que han sido registradas para permitir su acceso e inserta los accesos y salidas de dichos visitantes a la Universidad. Por su parte el Registro de Visitantes notifica las visitas a la UCI y emite reportes estadísticos sobre las visitas al centro, este sistema para realizar cada una de estas actividades consume los servicios que brinda “Gestor de Visitantes”.

De forma general cualquier sistema previamente autorizado podrá pedirle al servicio Web distintos reportes sobre las visitas, registrar una visita, insertar un acceso o salida de un visitante y gestionar los permisos de los usuarios, directivos, oficiales operativos y sistemas para realizar cada uno de estos procesos.

Aplicación Web “Registro de Visitantes”: mediante el Registro de Visitantes los directivos de la UCI serán capaces de notificar con antelación una visita a la Universidad insertando todos los datos correspondientes de la visita. Brinda un conjunto de reportes sobre los visitantes y las visitas. Además

este subsistema contiene una sección de administración la cual se encarga de gestionar los permisos de los directivos y oficiales operativos para solicitar reportes de información y notificar visitas al centro.

2.2 Línea Base de la Arquitectura

La Línea Base de la Arquitectura es un esqueleto del sistema con pocos músculos de software, pues no es más que la versión interna del sistema al final de la fase de elaboración. Tiene las versiones de todos los modelos que un sistema terminado contiene al final de la fase de construcción. Incluye el mismo esqueleto de subsistemas, componentes y nodos de un sistema definitivo, pero no existe toda la musculatura.

En la misma se exponen los estilos arquitectónicos de la aplicación, así como los principales elementos de la arquitectura. Se describen los principales patrones de arquitectura y patrones de diseño, unido a las tecnologías y herramientas de software que se utilizarán en el sistema a desarrollar.

Los usuarios de la Línea Base son:

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

- Los integrantes del equipo de desarrollo, quienes la usan como guía para la implementación del sistema.

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

Alcance

La Línea Base de la Arquitectura describe la estructura del sistema en un alto nivel de abstracción. Describe detalladamente el organigrama de la arquitectura según los estilos arquitectónicos que se utilizarán.

Se propone la utilización de patrones que resuelven problemas que se podrían presentar a lo largo del desarrollo del sistema y las principales tecnologías y herramientas que soportan los estilos y patrones especificados y que además deben cumplir con las restricciones del sistema.

2.3 Metodología, herramientas y lenguajes para el desarrollo del software

A continuación se especifica detalladamente las metodologías, herramientas y lenguajes que serán utilizados en la construcción de la aplicación propuesta, basándose en las tendencias del país y de la Universidad de las Ciencias Informáticas dentro del mercado del desarrollo en software libre. Además se dan a conocer elementos del por qué fueron escogidos.

2.3.1 Metodología

RUP es un proceso de desarrollo de software, o sea, conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Sin embargo, RUP es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto. RUP está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes de software interconectados a través de interfaces bien definidas y utiliza el UML.

Divide en fases el desarrollo del software (Inicio, Elaboración, Construcción, Transición), cada una de estas fases es desarrollada mediante un ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Se han agrupado las actividades en grupos lógicos, se va desarrollando el software por iteraciones e incrementos, es guiado por casos de uso, los que reflejan las necesidades de los futuros usuarios, centrado en la arquitectura, la cual muestra la visión común del sistema completo. Permite que en la medida que se vaya construyendo el software por ciclos se puedan detectar errores, una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo una de las metodologías más importantes y utilizada para alcanzar un grado de certificación en el desarrollo del software. El uso de esta metodología asegura que se produzca desde las primeras fases de desarrollo del software, un producto de calidad que cumpla con las características de funcionalidad, usabilidad y fiabilidad. Se usa RUP porque constituye

la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

2.3.2 Ambiente de desarrollo

Se sugiere utilizar ambientes de desarrollo para obtener aplicaciones robustas y flexibles ya que estos ambientes te ayudan a organizar, probar y escribir el código, aumentando la productividad y la calidad de la aplicación que se va a producir. Existen varios ambientes de desarrollo para PHP, pero en este caso se usa un ambiente de desarrollo gratuito que el Eclipse.

Se usa **Eclipse 3.2** (Callisto), porque presenta algunos plugins necesarios para la realización del trabajo como es el caso del pluging PDT 1.0 para el desarrollo en PHP, que permite conectarse al controlador de versiones que será utilizado como es el caso de Subversion 1.4.6.

Permite un ciclo de desarrollo más transparente y predecible. Tiene un muy buen soporte de refactorización mediante técnicas. Con esta versión se pueden utilizar todas las herramientas que un buen entorno de desarrollo integrado (IDE) ofrece. Posee una agradable interfaz, es confiable y muy utilizado por los programadores de hoy en día.

2.3.3 Control de versiones

Se usa el controlador de versiones **Subversion 1.4.6** siendo unos de los controladores más utilizados en proyectos de software libre, el cual sigue la historia de los archivos y directorios a través de copias y renombrados, la creación de ramas y etiquetas es una operación más eficiente; tiene costo de complejidad constante (O(1)) y no lineal (O(n)) como en CVS. Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos). Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto). Presenta alto rendimiento en la copia de trabajos enormes, el formato de copia que presenta esta versión permite que los clientes busquen rápidamente una copia de trabajo.

Posee soluciones a fallos, de modo que si ocurre algún problema se active un sistema de recuperación automático. Contiene nuevos interruptores y opciones de líneas de comandos para el cliente que lo hacen más fácil y cómodo que las otras versiones.

2.3.4 Contenedor web

Se utiliza **Apache 2.2.6** como servidor Web. El servidor Web simplemente se encarga de servir páginas estilo Web hacia quien lo solicita, que por lo general es un PC o dispositivo con un visor de Web. Apache es el servidor web por excelencia, con algo mas de un 60% de los servidores de internet confiando en él. Entre sus características mas sobresalientes están:

- **Fiabilidad**: Alrededor del 90% de los servidores con más alta disponibilidad funcionan con Apache.

- **Gratuidad**: Apache es totalmente gratuito, y se distribuye bajo la licencia, Apache Software License, que permite la modificación del código.

- **Extensibilidad**: Se pueden añadir módulos para aumentar las ya de por si amplias capacidades de Apache. Hay una extensa variedad de módulos, que permiten desde generar contenido dinámico (con PHP, Java, Perl, Python), monitorizar el rendimiento del servidor, atender peticiones encriptados por SSL, hasta crear servidores virtuales por IP o por nombre (varias direcciones web son manejadas en un mismo servidor) y limitar el ancho de banda para cada uno de ellos. Dichos módulos incluso pueden ser creados por cualquier persona con conocimientos de programación.

Apache presenta mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. Es multiplataforma. Elabora índices de directorios. Brinda beneficios de administración basada en la configuración de un único archivo, siendo todo esto de gran utilidad en el proyecto.

2.3.5 Lenguajes de programación web

EL lenguaje a utilizar del lado del servidor es **PHP5** escogido por ser un lenguaje de programación orientado a objetos. Cumple con las políticas de software libre y su código está disponible bajo la licencia GPL. Brinda excelente soporte de acceso a base de datos que para esta propuesta que es PostgreSQL. Su código es mucho más legible que el de PERL. Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (encriptación, envió de correo, acceso a

base de datos). Como en todos los sistemas se utiliza el mismo código base, los scripts pueden ser ejecutados de manera independiente al sistema operativo. Es multiplataforma. Es un lenguaje muy usado a nivel mundial con un gran éxito.

Los lenguajes a utilizar del lado del cliente son:

HTML, lenguaje de marcado predominante de gran utilidad que se encarga de la construcción de páginas web, de describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

Java Scripts, lenguaje de programación interpretado, es decir, que no requiere compilación, que se utilizada principalmente en páginas web siendo muy útil en el desarrollo de este software.

CSS, aquí se utiliza para dar estilo a documentos HTML. Los estilos a utilizar definen la forma de mostrar los elementos HTML. También permite controlar el estilo y el formato de múltiples páginas Web al mismo tiempo agilizando el trabajo relacionado con los estilos a lo que se desea mostrar. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento es algo fundamental que se ha tenido en cuenta para el este trabajo propuesto.

2.3.6 Gestor de base de datos

PostgreSQL 8.2, es un gestor de bases de datos que tiene prácticamente todo lo que poseen los gestores comerciales de estos tiempos. Además los tipos de datos base también soporta datos de tipo fecha, datos sobre redes (MAC, IP...), cadenas de bits, al mismo tiempo que permite la creación de tipos propios. Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.

Se utiliza este gestor, dado a que mejora el rendimiento en alrededor de un 20%. Las mejoras incluyen, ordenamientos en memoria y en disco más rápido, mejor escalabilidad en sistemas multi-procesador, mejor optimización de consultas sobre datos particionados, cargas masivas más rápidas y outer joins considerablemente acelerados. Permite a los administradores crear fácilmente una copia para recuperación inmediata (failover) de su cluster de bases de datos. La construcción de índices

puede ocurrir mientras las aplicaciones escriben en las tablas de la base de datos, permitiendo el afinamiento del rendimiento sin afectar la disponibilidad. Se puede definir un valor de retorno para los insert/update/delete, en los updates ahora se pueden utilizar múltiples columnas con una lista de valores haciendo más fácil el trabajo.

También decir que se utiliza este gestor porque es Open Source, bajo la licencia BSD, es multiplataforma, con un uso y distribución gratis. Para esta propuesta se tuvo en cuenta fundamentalmente que es el único gestor de base de datos aprobado por la Dirección de Informatización de la Universidad de las Ciencias Informáticas.

2.3.7 Herramienta de modelado

Como lenguaje para el modelado se utilizará **UML** (Lenguaje Unificado de Modelado), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, permite visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

Visual Paradigm UML 6.0, Se utilizará Visual Paradigm porque es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo.

Esta herramienta ofrece:

- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa 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 (Integrated Development Environment).
- Disponibilidad en múltiples plataformas.
- Ofrecen un mecanismo general para la organización de los modelos/subsistemas/capas agrupando elementos de modelado.
- Versión gratuita (licencia para Community Edition).

Esta herramienta, se integra con Eclipse, el cual es el ambiente de desarrollo de este proyecto.

2.3.8 Marco de Trabajo

Symfony 1.0.16, es desarrollado completamente con PHP5, lo cual su programación es orientada a objetos, beneficiando el desarrollo del sistema. Se puede ejecutar tanto en plataformas *nix (Unix, Linux) como en plataformas Windows. Es independiente del sistema gestor de bases de datos que en este caso es PostgreSQL. Posee una herramienta llamada Propel que se utiliza para el mapeo de objetos a bases de datos. Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos. La seguridad se hace muy sencilla. Se hace posible realizar cambios en la configuración (sin necesidad de reiniciar el servidor). El completo sistema de log permite a los administradores acceder hasta el último detalle de las actividades que realiza la aplicación. Los formularios incluyen validación automatizada y relleno automático de datos, lo que asegura la obtención de datos correctos. Todo esto facilita de forma óptima el trabajo.

2.4 Estilo de la Arquitectura

La arquitectura propuesta para desarrollar el servicio web Gestor de Visitantes es el estilo **arquitectura en capas**. Se propone implementar una arquitectura en capas porque esta es la que más se adapta al problema existente, en esta arquitectura a cada nivel se le confía una misión simple, lo que permite el diseño de una arquitectura escalable, o sea, se puede ampliar con facilidad en caso de que las necesidades del sistema aumenten. El desarrollo se puede llevar a cabo en varios niveles y en caso de

algún problema o cambio se ataca al nivel requerido sin tener que revisar entre código mezclado, reduce las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores, existiendo independencias entre las capas para cualquier detalle.

Este estilo permite ganar en organización del sistema, además soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales, proporciona una amplia reutilización, facilita la corrección de errores y se obtiene al final una solución altamente flexible y reutilizable como la que se quiere para el Sistema de Registro de Visitantes.
En la actualidad muchos sistemas están basados en arquitecturas de dos capas, denominadas generalmente nivel de aplicación y nivel de la base de datos. En este caso se propone 2 capas reduciendo las dependencias entre ellas, las capas serían: lógica de negocio, acceso a datos, la lógica del negocio se encargaría de todo lo relacionado con los servicios que presta y la de acceso a datos representa toda la información necesaria para brindar los servicios.

La aplicación web Registro de Visitantes está basada en un patrón clásico del diseño web conocido como arquitectura **Modelo-Vista-Controlador (MVC),** se basa en separar la lógica de negocio (el modelo), la presentación al usuario (la vista) y el control del flujo de la aplicación (el controlador), de manera que, de haber cambios en uno de estos componentes afecten en la menor medida posible al resto, por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

Aplicando MVC a la aplicación Web se desarrollan los siguientes componentes:

- **Modelo**: representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.

- **Vista**: transforma el modelo en una página web que permite al usuario interactuar con ella.

- **Controlador**: se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

2.5 Patrón de Diseño

El patrón de diseño que se utiliza es **Data Access Object (DAO, Objeto de Acceso a Datos)**, es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una base de datos o archivos.

EL patrón DAO es una solución al problema del diferencial de impedancia de una aplicación orientada a objetos y una base de datos relacional, empleando únicamente la interfaz de programación (API) nativa del manejador de base de datos, o algún otro sustituto como el ODBC, DBI, entre otros.

Se usa este patrón para:

- Abstraer y encapsular los accesos.
- Gestionar las conexiones a las fuentes de datos.
- Obtener los datos almacenados.

Ventajas que proporciona:

- Cualquier objeto no requiere conocimiento directo del destino final de la información que se manipula.
- Se baja en nivel de acoplamiento entre clases, reduciendo la complejidad de realizar cambios.
- Se aísla las conexiones a las fuentes de datos en una capa fácilmente identificable y mantenible.

2.6 Conclusiones

En este capítulo se hace una descripción del sistema a grandes rasgos para la solución de las problemáticas existentes y la justificación de todas las herramientas, lenguajes, metodología, arquitectura y patrones que se van a utilizar en la presente propuesta, logrando hacer una definición de la arquitectura.

CAPÍTULO 3. DESCRIPCIÓN DE LA ARQUITECTURA

Introducción

El conjunto de modelos que describen la Línea Base de la Arquitectura se denomina *Descripción de la Arquitectura*. El papel de la descripción de la arquitectura es guiar al equipo de desarrollo a través del ciclo de vida del sistema. Esta descripción puede adoptar diferentes formas, puede ser un conjunto de vistas, o puede ser una reescritura de estos de forma que sea más fácil leerlos. En este capítulo se especifican los requerimientos funcionales y no funcionales, las vistas fundamentales que componen a la arquitectura y diagramas que son arquitectónicamente significativos para el sistema.

3.1 Metas y restricciones arquitectónicas

A continuación se plantean las metas y restricciones con las que se deberá cumplir para la correcta puesta en funcionamiento del Sistema Gestión de Credenciales propuesto y que además son significativas para la arquitectura.

3.1.1 Requerimientos

Un requerimiento es una condición o capacidad necesaria para que un usuario resuelva un problema o alcance un objetivo. A continuación se describen los requerimientos funcionales y no funcionales del sistema.

Requerimientos funcionales

RF-1 Registrar Visitante

RF-1.1 Registrar las entradas de visitantes

RF-1.2 Registrar las salidas de visitantes

RF-1.3 Eliminar las entradas de visitantes

RF-1.4 Eliminar las salidas de visitantes

RF-2 Consultar Reportes de Visitas

RF-2.1 Mostrar reportes

RF-3. Administrar Sistema

RF-3.1 Brindar permisos de usuario

RF-1.2 Brindar servicios

RF-4. Controlar Acceso de Visitantes

RF-4.1 Verificar autorización

RF-4.2 Almacenar datos de visitantes

Requerimientos no funcionales

- **Requerimientos de apariencia o interfaz externa:** El sistema debe tener un ambiente amigable y entendible para los usuarios finales, de forma tal que no les sea muy complicado utilizar el software.

- **Requerimientos de usabilidad:** El sistema podrá ser usado por personas autorizadas, la navegabilidad debe no debe ser muy compleja, todas las funcionalidades deben ser rápidamente accesibles por el usuario.

- **Requerimientos de rendimiento:** El tiempo de respuesta de una petición al servidor deber ser rápido ya que el sistema operara con grandes cantidades de datos.

- **Requerimientos de soporte:** Se le debe dar mantenimiento periódicamente a los servidores de bases de datos controlando la integridad de la información.

- **Requerimientos de portabilidad:** Debe tener facilidad para adaptarlo a diferentes ambientes. Independencia de la plataforma.

- **Requerimientos de seguridad y privacidad**

    - Identificar al usuario antes de que pueda realizar cualquier acción sobre la configuración del sistema.

    - Garantizar que la información sea vista solo por personas autorizadas.

    - Verificación sobre acciones irreversibles (eliminaciones).

    - Garantía de que el sistema funcione correctamente aun cuando no haya conectividad.

- **Requerimientos de confiabilidad:** Se debe chequear la integridad de los datos.

- **Requerimientos Legales:**
  Debe cumplir con:

    - Decreto Ley – 186 / 98
      2da Sección, Artículos 5, 8, 11, Inciso D.

    - Reglamento del Decreto Ley 186 (Resolución No. 2 / 2001 del Ministerio del Interior).
      Artículo 1. Capítulo IV (Artículos 3, 4, 90)

    - El Plan de Seguridad y Protección del Centro.

- **Requerimientos de Ayuda y documentación en línea:** Se debe brindar un sistema de ayuda que explique las diferentes funcionalidades con que cuenta el sistema, además los manuales de usuario.

- **Requerimientos en el diseño y la implementación:** Para el análisis y el diseño del sistema debe ser utilizada la metodología RUP, usando el lenguaje de modelado UML y como herramienta para llevarlo a cabo el Visual Paradigm.

- **Requerimientos de software:** Se debe disponer en el servidor con Linux. Se utilizará como lenguaje de programación: PHP5, Java Scripts, HTML, CSS y como gestor de Base de Datos: PostgreSQL.

- **Requerimientos de hardware:** Requiere estar instalada en una PC Pentium, 256 Mb de RAM (como mínimo), Micro 2.0 o superior, 280 Mb de disco duro, una tarjeta de red de 100Mbps y el procesador de 133 MHz o superior.

3.2 Vistas Arquitectónicas

En RUP, la arquitectura se compone de 4+1 vistas fundamentalmente: Vista de Casos de Uso, Vista Lógica, Vista de Procesos, Vista de Implementación y Vista de Despliegue. Estas vistas son la parte esencial de lo que se obtuvo en los flujos de Requerimientos, Análisis y Diseño e Implementación, que son las etapas que más tributan a la arquitectura. La arquitectura se representa a través de 4+1 vista, dicho de modo que se entienda que 4 de estas vistas están regidas por una rectora: la Vista de Casos de Uso. Esto responde, además, a una de las características de RUP: Guiado por Casos de Usos.

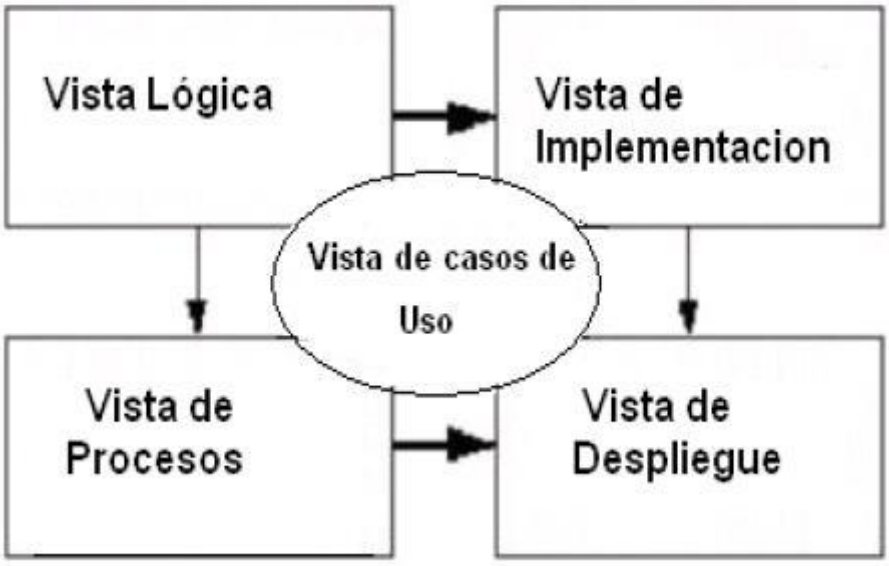

Figura 7. Modelo de "4+1" vistas

3.2.1 Vista de Casos de Uso

La vista de casos de uso representa un subconjunto del artefacto Modelo de Casos de Uso y lista los casos de uso o escenarios del modelo de casos de uso más significativos, con las funcionalidades centrales del sistema. En esta vista se representa el diagrama de cases de usos arquitectónicamente significativos para el sistema y una pequeña explicación de cada uno.

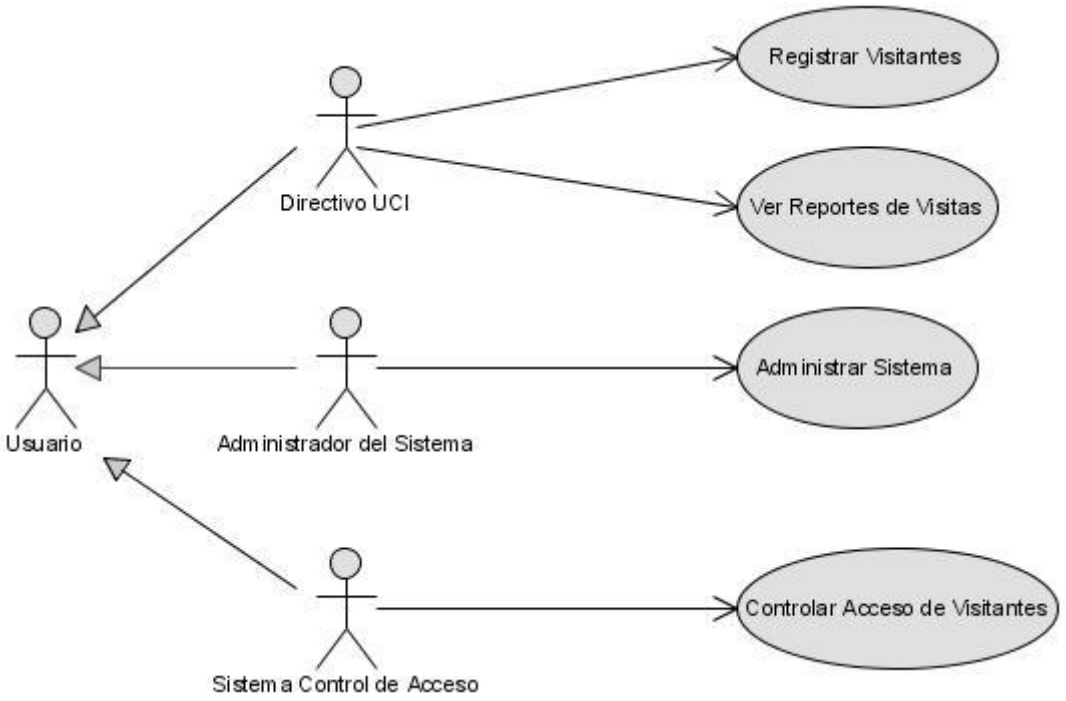

Figura 8. Vista de Casos de Uso Arquitectónicamente Significativos

Resumen de cada caso de uso

Registrar Visitantes: Los Directivos UCI son los encargados de iniciar este caso de uso conectándose a la aplicación web Registro de Visitantes autenticándose, en la cual registra la visita mostrando que ha sido registrada y luego cierra la aplicación, finalizando así el caso de uso.

Ver reportes de visitas: Los Directivos UCI son los encargados de iniciar este caso de uso conectándose a la aplicación web Registro de Visitantes autenticándose, en la cual pude ver los reportes de las visitas efectuadas y no efectuadas, luego cierra la aplicación, finalizando así el caso de uso.

Administrar Sistema: El Administrador del Sistema es el encargado de iniciar el caso de uso conectándose a la aplicación web Registro de Visitantes autenticándose, en la cual brinda los permisos de usuarios y administra los servicios del servicio web, luego cierra la aplicación, finalizando así el caso de uso.

Controlar Acceso de Visitantes: El Sistema Control de Acceso es el encargado de iniciar este caso de uso conectándose al servicio web el cual le brinda los servicios para controlar el acceso de los visitantes, al efectuarse los servicios finaliza el caso de uso.

Todos los servicios brindados en cada caso de uso los facilita el servicio web (Gestor de Visitantes).

3.2.2 Vista Lógica

La vista lógica representa un subconjunto del artefacto Modelo de Diseño, representando los elementos de diseño más importantes para la arquitectura del sistema. Se describen las clases más importantes, su organización en paquetes y subsistemas.

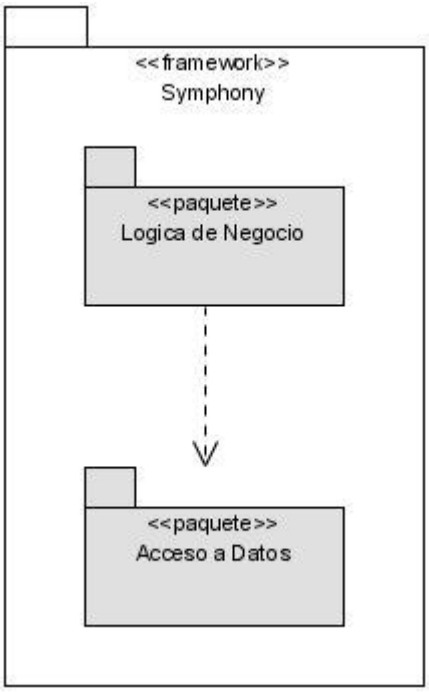

Figura 9. Vista Lógica del módulo "Gestor de Visitantes"

Paquete Lógica del Negocio: dentro de este paquete se encuentran las clases que llevan a cabo toda la lógica del negocio.

Las clases que contiene este paquete son las siguientes:

WebServicieCF.php: Clase controlador frontal del Gestor de Visitantes, es donde se ofrece un punto de entrada único para toda la aplicación. Maneja la seguridad del sistema. Carga la configuración de la aplicación. Se encarga del manejo de los servicios pedidos por otros sistemas clientes mostrándole una respuesta.

Action.class.php: aquí se encuentra todas las acciones que se llevan a cabo en el módulo Gestor de Visitantes.

Paquete Acceso a Datos: este paquete contiene las clases que van a trabajar con el manejo de los datos.

Las clases que contiene este paquete son las siguientes:

AccesoDatos.php: clase que maneja todas las acciones relacionadas con los datos.

Las otras clases que aparecen en este paquete son generadas por la herramienta Propel cuando mapea la base de datos, dependiendo de las tablas de la base de datos.

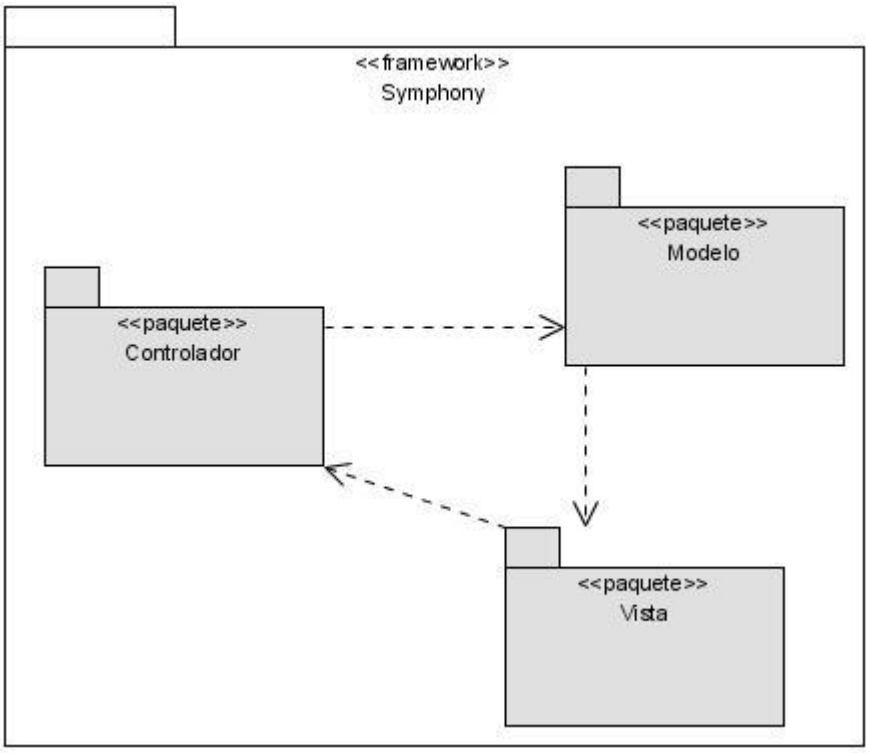

Figura 10. Vista Lógica del módulo "Registro de Visitantes"

Paquete Controlador: dentro de este paquete se encuentran las clases que llevan a cabo toda la lógica del negocio.

Las clases que contiene este paquete son las siguientes:

WebServicieCF.php: Clase controlador frontal de la aplicación Registro de Visitantes, es donde se ofrece un punto de entrada único para toda la aplicación. Maneja la seguridad del sistema. Carga la configuración de la aplicación. Se encarga de los pedidos por los usuarios clientes mostrándole una respuesta.

Action.class.php: aquí se encuentra todas las acciones que se llevan a cabo en el módulo Registro de Visitantes.

Paquete Modelo: este paquete contiene las clases que van a trabajar con el manejo de los datos.

Las clases que contiene este paquete son las siguientes:

ManipuladorDatos.php: clase que maneja todas las acciones relacionadas con los datos, la cual se conecta al servicio web Gestor de Visitantes.

Paquete Vista: dentro de este paquete se encuentran las clases que se encargan de producir las páginas que se muestran como resultado de las acciones.

Las clases que contiene este paquete son las plantillas para cada acción, una clase layout.php que contiene lo común que tiene cada plantilla para cada acción y una clase final que va a estar conformada por la plantilla para cada acción junto a la clase layout.php, la cual es el resultado final para mostrarle al usuario.

3.2.3 Vista de Implementación

El propósito de la vista de implementación es el de capturar las decisiones arquitectónicas hechas por el implementador. Típicamente la vista de implementación contiene:[10]

- Un listado de todos los subsistemas del modelo de implementación.
- Un diagrama de componentes que ilustre la organización de los subsistemas y los organice en capas y jerarquías.
- Ilustraciones de las dependencias de importaciones (paquetes cuyas clases están referencias dentro de otro paquete dado) entre subsistemas.

La vista de implementación es útil para:

- Calcular el trabajo de implementación de los individuos y equipos.
- Calcular la cantidad de código a ser desarrollado, modificado, o eliminado.
- Planificar el reuso a gran escala.
- Considerar las estrategias de liberación.

Esta vista muestra gráficamente todas las clases más importantes, su organización en paquetes y subsistemas, como se puede apreciar a continuación en los diferentes módulos.

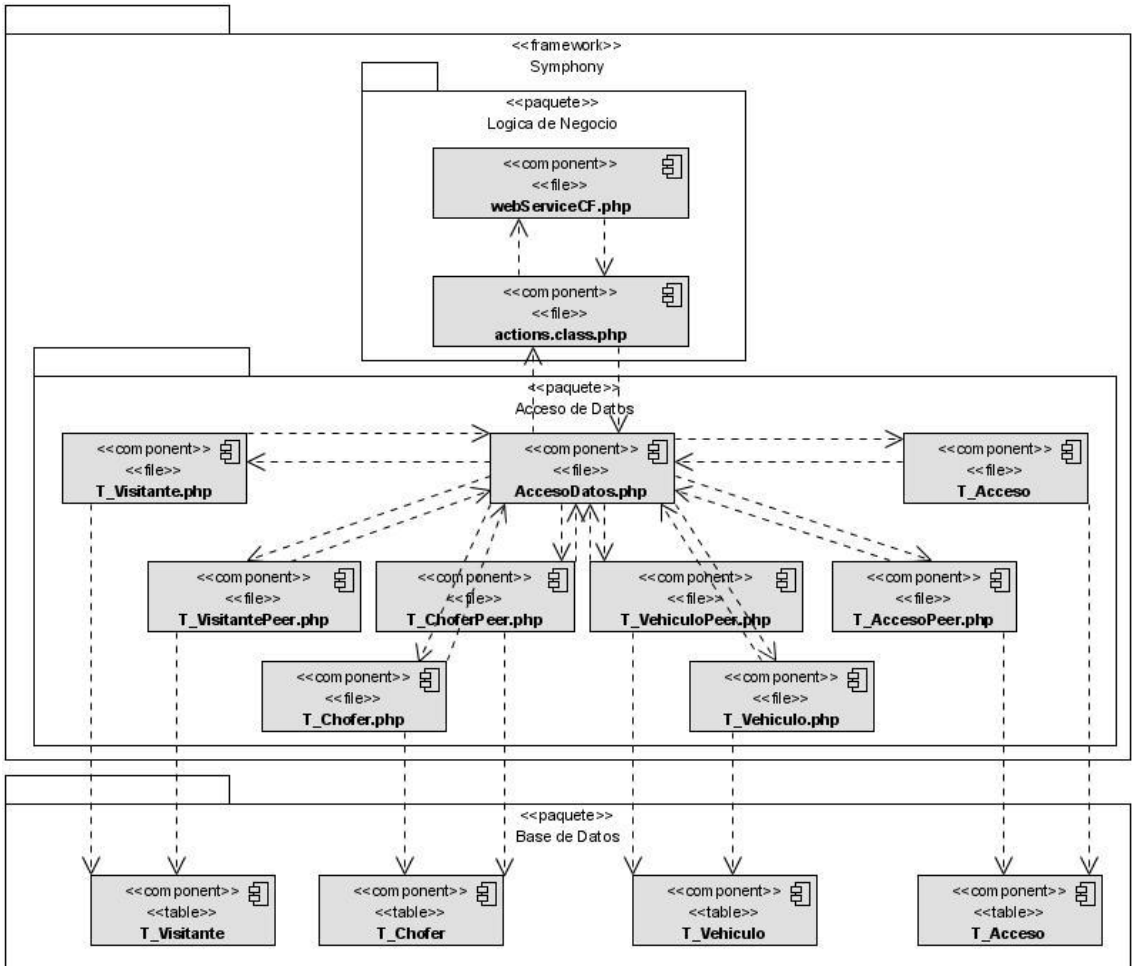

Figura 11. Vista de Implementación del módulo "Gestor de Visitantes"

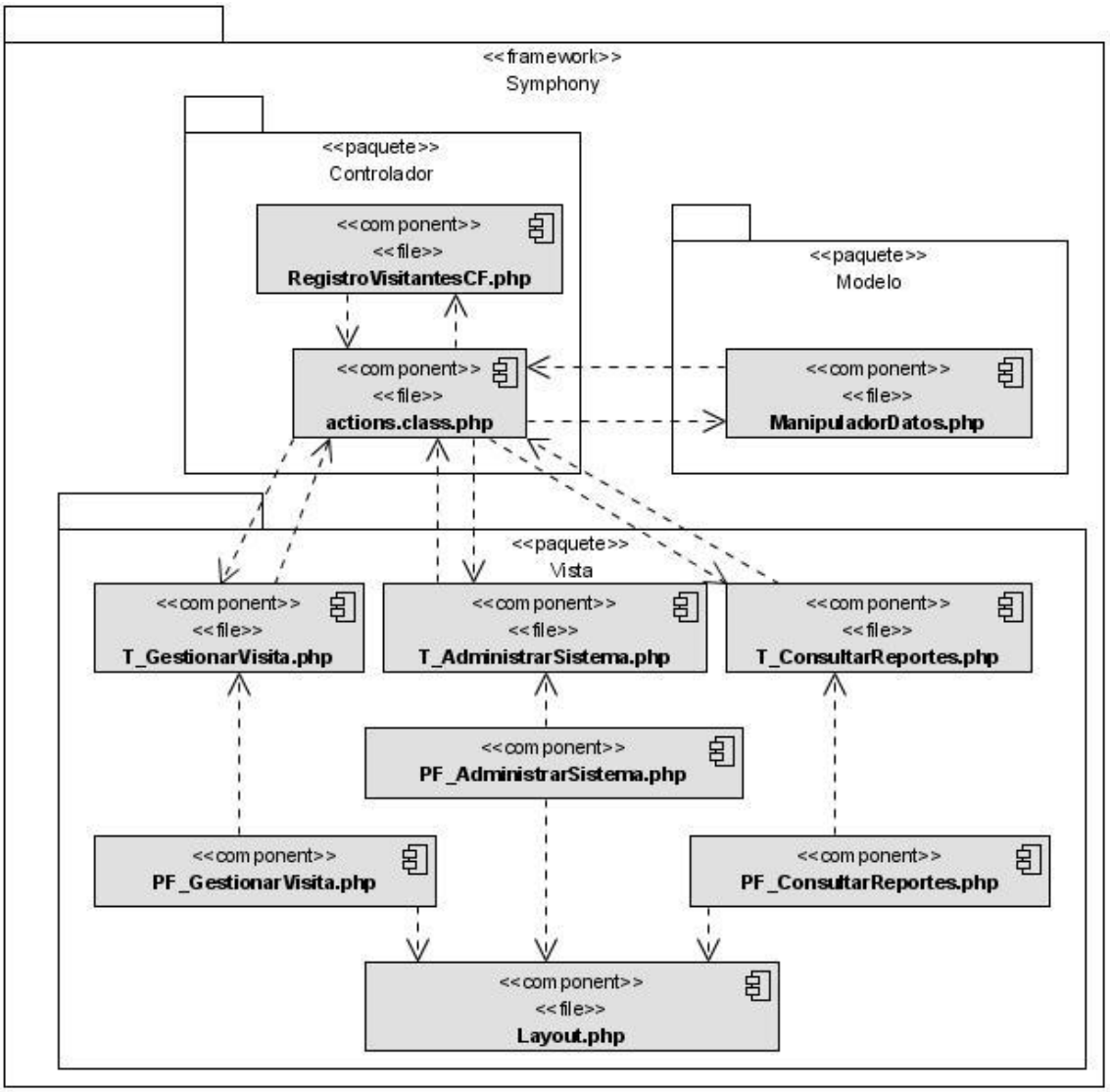

Figura 12. Vista de Implementación del módulo "Registro de Visitantes"

3.2.4 Vista de Despliegue

Mediante la vista de despliegue se obtiene una base para la comprensión de la distribución física de un sistema a través de nodos. Suele usarse cuando el sistema esta distribuido, y hay una traza directa del modelo de implementación puesto que cada componente físico debe estar almacenado en un nodo.

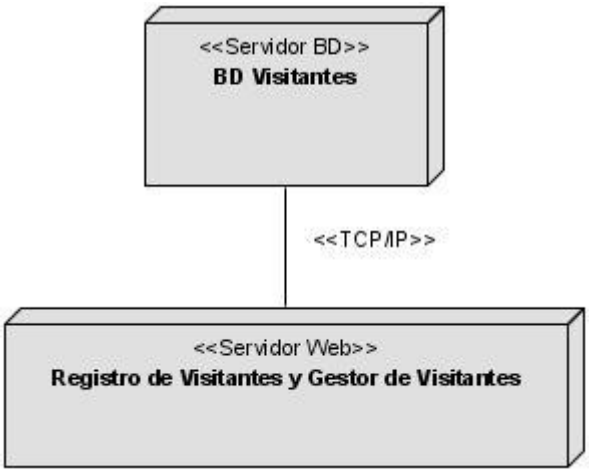

Figura 13. Vista de Despliegue del Sistema

Esta vista esta compuesta por dos nodos:

Servidor de Base de Datos: en este nodo se despliega la base de datos Visitantes, la cual contiene todo los datos necesario para el sistema.

Servidor Web: en este nodo se despliega la aplicación web Registro de Visitantes y el servicio web Gestor de Visitantes los cuales se encargan del correcto funcionamiento del sistema.

3.2.5 Vista de Procesos Buscar

La vista de procesos solo suele usarse cuando el sistema presenta procesos concurrentes o hilos. En este caso la solución propuesta no presenta procesos concurrentes por tanto no se hace representación de esta vista.

3.3 Conclusiones

En el presente capitulo se plantea una descripción de la arquitectura detallada mostrándose las metas, restricciones y las vistas arquitectónica dando una explicación concreta para formular completamente la arquitectura propuesta para el desarrollo del sistema.

CONCLUSIONES GENERALES

Durante el desarrollo del presente trabajo se cumplieron las tareas y objetivos propuestos para la investigación, realizándose un análisis del estado del arte referente a las arquitecturas de los softwares similares existentes asociados al campo de acción. Se evaluaron las tecnologías, metodologías y tendencias actuales, permitiendo la selección de las herramientas para software libre adecuadas a las necesidades para el desarrollo del software. Se propuso el diseño de la arquitectura para el Sistema de Registro de Visitantes, teniendo en cuenta su desarrollo en software libre, fundamentándose principalmente en la necesidad que actualmente presenta la Universidad de las Ciencias Informáticas, el cual permite el control de los visitantes.

RECOMENDACIONES

Se recomienda:

1. Realizar la viabilidad de la arquitectura propuesta.

2. Realizar una evaluación de los costes de los hardwares necesarios para el desarrollo del sistema.

3. Insertar un módulo de impresión de credenciales de visitantes para su identificación en la UCI.

4. Obtener los equipos adecuados para lograr el mejor resultado y funcionamiento de la propuesta definida en la investigación.

BIBLIOGRAFÍA CITADA

1. **Dewayne E. Perry, Alexander L. Wolf.** *Foundations for the Study of Software Architecture.* pág. 40.
2. **Mary Shaw, David Garlan. 1996.** *Discipline, Software Architecture : Perspectives on an Emerging.* Prentice Hall : s.n., 1996. pág. 242.
3. **Len Bass, Paul Clements, Rick Kazman. 2003.** *Software Architecture in Practice (2nd Edition).* Addison-Wesley : s.n., 2003.
4. **Kaisler, Stephen H. 2005.** *Software Paradigms.* 2005. pág. 440.
5. **2003.** Bertrán Software. [En línea] 2003. [Citado el: 2008 de abril de 20.] http://www.bertran.com/.
6. Sistemas Tecnologicos S.A. [En línea] [Citado el: 2008 de abril de 23.] http://www.sistecbio.com.ar.
7. Beek Corporativo S.A. [En línea] [Citado el: 2008 de abril de 24.] http://www.beek.com.mx/visitantes/.
8. **2008.** Diseño y tecnología en información. [En línea] 2008. [Citado el: 2008 de mayo de 23.] http://www.ne.com.co/html/esp/calidad.html.
9. **Rojas, Juan Carlos Olivares.** Sep. [En línea] [Citado el: 2008 de mayo de 27 .] http://antares.itmorelia.edu.mx/~jcolivar/documentos/msf.pdf.
10. **Pérez, Carlos Torres.** IEEE. [En línea] [Citado el: 2008 de junio de 25.] http://www.ewh.ieee.org/r9/guadalajara/boletin/marzo02/vistaimpl.htm.

BIBLIOGRAFÍA CONSULTADA

1. **Clemens Szyperski, Dominik Gruntz & Stephan Murer. 2002.** *Component Software : Beyond Object-Oriented Programming.* 2002. pág. 589.
2. **David M. Dikel, David Kane, James R. Wilson. 2001.** *Software Architecture : Organizational Principles and Patterns.* Prentice Hall : s.n., 2001. pág. 320.
3. **Erich Gamma, Kent Beck. 2003.** *Contributing to Eclipse: Principles, Patterns, and Plug-ins.* 2003. pág. 395.
4. **Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal. 1996.** *Pattern-Oriented Software Architecture, Volume 1 : A System of Patterns.* 1996. pág. 284.
5. **Jacobson, Booch & Rumbaugh. 1999.** *El Lenguaje Unificado de Modelado.* 1999.
6. **Larman, Craig. 1999.** *UML y Patrones. Introducción al análisis y diseño orientado a objectos.* Prentice-Hall : s.n., 1999. pág. 536.
7. **Len Bass, Paul Clements, Rick Kazman. 2003.** *Software Architecture in Practice (2nd Edition).* Addison-Wesley : s.n., 2003.
8. **Mary Shaw, David Garlan. 1996.** *Discipline, Software Architecture : Perspectives on an Emerging.* Prentice Hall : s.n., 1996. pág. 242.
9. **2008.** MeRinde. [En línea] 2008. [Citado el: 27 de junio de 2008.] http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=62&Itemid=296.
10. **Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford. 2003.** *Documenting Software Architectures : Views and Beyond.* [ed.] Paul Clements. Addison Wesley Professional : s.n., 2003. pág. 512.
11. **Pecos, Daniel.** PostGreSQL vs. MySQL. [En línea] [Citado el: 22 de 6 de 2008.] http://www.netpecos.org/docs/mysql_postgres/index.html..
12. **Sklar, David. 2004.** *Learning PHP 5: A Pain-free Introduction to Building Interactive Web Sites.* 2004. pág. 350.
13. **Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Zeev Suraski,. 15-04-2001.** *Manual de PHP.* [ed.] Rafael Martínez. 15-04-2001.
14. **2008.** Visual Paradigm. [En línea] 2008. [Citado el: 2008 de junio de 15.] http://www.visual-paradigm.com/product/dbva/.2008.
15. **Zandstra, Matt. 2004.** *PHP 5 Objects, Patterns, and Practice.* 2004. pág. 437.

16. **Zaninotto, Fabien Potencier & Francois. 2008.** Symfony, la guía definitiva. [En línea] 2008. [Citado el: 13 de mayo de 2008.] http://www.librosweb.es/symfony/index.html.
17. **2008.** Diseño y tecnología en información. [En línea] 2008. [Citado el: 2008 de mayo de 23.] http://www.ne.com.co/html/esp/calidad.html.
18. **Kaisler, Stephen H. 2005.** *Software Paradigms.* 2005. pág. 440.
19. **2003.** Bertrán Software. [En línea] 2003. [Citado el: 2008 de abril de 20.] http://www.bertran.com/.
20. Sistemas Tecnologicos S.A. [En línea] [Citado el: 2008 de abril de 23.] http://www.sistecbio.com.ar.
21. Beek Corporativo S.A. [En línea] [Citado el: 2008 de abril de 24.] http://www.beek.com.mx/visitantes/.
22. **Rojas, Juan Carlos Olivares.** Sep. [En línea] [Citado el: 2008 de mayo de 27 .] http://antares.itmorelia.edu.mx/~jcolivar/documentos/msf.pdf.
23. **Pérez, Carlos Torres.** IEEE. [En línea] [Citado el: 2008 de junio de 25.] http://www.ewh.ieee.org/r9/guadalajara/boletin/marzo02/vistaimpl.htm.
24. **Kruchten, Philippe. 2003.** *The Rational Unified Process: An Introduction.* 2003. pág. 336.
25. **Simoes, Dorival.** SOA.¿Cual es el próximo paso? [En línea] [Citado el: 2008 de mayo de 20 .] http://64.116.224.233/detras_el_paso_que_sigue/fichas/soa.html.
26. **Dewayne E. Perry, Alexander L. Wolf.** *Foundations for the Study of Software Architecture.* pág. 40.

GLOSARIO DE TÉRMINOS

API: Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Metodología: En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo.

Rational Unified Process (RUP): Es un proceso de desarrollo de software.

TCP/IP: (Transfer Control Protocol/ Internet Protocol, Protocolo de Control de Transmisión/ Protocolo de Internet) Protocolo de transmisión de datos en Internet, ampliamente utilizado, permite conectar computadoras con diferentes sistemas operativos.

Servicios Web: Un servicio Web es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, a través de la Web. Los Servicios Web pueden ser utilizados por distintas aplicaciones de software, desarrollados en diferentes lenguajes de programación y ejecutados sobre cualquier plataforma, para intercambiar datos en redes de ordenadores como Internet.

HTTP (Hyper Text Transfer Protocol): En español, protocolo de transferencia de hipertexto, es el protocolo usado en cada transacción de la Web. El hipertexto es el contenido de las páginas Web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido. También sirve el protocolo para enviar información adicional en ambos sentidos, como formularios con campos de texto.

Interfaz: Un límite a través del cual dos entidades independientes se encuentran y se relacionan o comunican la una con la otra.

Release: Se refiere a un producto final, preparado para lanzarse como versión definitiva a menos que aparezcan errores que lo impidan. En esta fase el producto implementa todas las funciones del diseño y se encuentra libre de cualquier error que suponga un punto muerto en el desarrollo.

Plugin: Pieza de software, generalmente pequeña, que añade funcionalidades a un software mayor. El usuario, partiendo de una versión reducida del software, añade aquellas capacidades nuevas que necesita, eliminando así la necesidad de tener que instalar un software con todas las posibilidades.