ARQUITECTURA PARA EL SISTEMA DE GESTIÓN DE INFORMACIÓN PARA LAS COORDINACIONES REGIONALES DE PREVENCIÓN DEL DELITO DE LA REPÚBLICA BOLIVARIANA DE VENEZUELA.

Ing. Asdrúbal Torres Camilo [1] acamilo@grm.uci.cu

Ing. Ané Caridad Aguilar Machado [2] anecary@grm.uci.cu

Universidad de las Ciencias Informáticas-Facultad Regional de Granma (UCI-FRG), Cuba.

RESUMEN

El presente trabajo se realizó a raíz de la necesidad de lograr una mejor integración de los componentes del Sistema de Gestión de Información para las Coordinaciones Regionales de Prevención del Delito de la República Bolivariana de Venezuela. Para darle solución a esta situación se decidió definir la Arquitectura de Software del sistema.

Dicha arquitectura cuenta con los elementos necesarios para el desarrollo del sistema. Los mismos tuvieron su éxito por las decisiones arquitectónicas que fueron tomadas a lo largo del ciclo de vida del desarrollo del sistema. Otro aspecto que garantizó el éxito fue la descripción de la arquitectura, representada por medio de las vistas de la arquitectura definidas por la metodología de desarrollo RUP. Con el objetivo de que las principales tecnologías y herramientas garanticen una realización exitosa del sistema se llevó a cabo un profundo estudio de las mismas, teniéndose en cuenta principalmente que estuviesen bajo licencia libre. Fue utilizada como plataforma de desarrollo JEE. Se desarrolló el sistema empleando el patrón arquitectónico arquitectura en capas y el mismo fue implementado haciendo uso de los frameworks Dalas y Spring fundamentalmente.

Finalmente, se realizó la evaluación de la arquitectura haciendo uso del método de evaluación Architecture Tradeoff Analisys Method (ATAM) y obteniéndose resultados satisfactorios.

Palabras Clave: Arquitectura de Software, Dalas, Java, JEE, Spring.

INTRODUCCIÓN

Con el triunfo del presidente venezolano Hugo Rafael Chávez Frías, se prioriza una amplia agenda social, en la que se encuentran sectores como la salud, la educación, el empleo así como la seguridad ciudadana. Para garantizar la paz y el bienestar de la sociedad se hace necesario disminuir la delincuencia que presenta el pueblo venezolano, por lo que se buscan vías de solución. Como parte de la cooperación entre Cuba y la República Bolivariana de Venezuela, se desarrolla el proyecto Prevención del Delito por parte de la Universidad de las Ciencias Informáticas (UCI) conjuntamente con el Ministerio del Poder Popular para las Relaciones Interiores y de Justicia (MPPRIJ) de la República Bolivariana de Venezuela.

El MPPRIJ, teniendo en cuenta su misión institucional de garantizar la seguridad ciudadana, promueve la formulación y puesta en marcha del Proyecto Implantación de Solución Integral para el perfeccionamiento del Sistema de Prevención del Delito de la República Bolivariana de Venezuela, con el propósito de brindar a las Coordinaciones Regionales una tecnología que permita un mejor desempeño y de esta manera brindar una asistencia de mayor calidad al ciudadano.

La DGPD, con el propósito de lograr una mejora en el procesamiento de las informaciones y el cumplimiento de las actividades a realizar, necesita de los datos de las distintas Coordinaciones Regionales ubicadas en diferentes puntos de la geografía venezolana, requiriéndose una integración entre dichos entes para el envío de la información. Existe además diversidad en la tecnología utilizada, lo

que significa que algunos de los procesos se manejan en formato digital, aunque muchos de ellos son elaborados de manera manual y guardados en formato duro. Esto provoca que el envío de las informaciones sea más lento y que no exista una adecuada seguridad de las informaciones manejadas. Además, no se hace posible una eficiente toma de decisiones y en la DGPD no se cuenta con información centralizada para elaborar informes referentes a la seguridad ciudadana, crear proyectos y programas orientados al crimen y la violencia.

Una solución viable a esta situación es desarrollar un Sistema de Gestión de Información para las Coordinaciones Regionales de Prevención del Delito.

Para el desarrollo de la investigación se ha trazado como **objetivo general**: Definir la Arquitectura de Software del Sistema de Gestión de Información para las Coordinaciones Regionales de Prevención del Delito.

CONCEPTOS GENERALES

La arquitectura, conformada por diferentes visiones del sistema, constituye un modelo de cómo está estructurado dicho sistema, sirviendo de comunicación entre las personas involucradas en el desarrollo y ayudando a realizar diversos análisis que orienten el proceso de toma de decisiones.

Arquitectura de Software

La definición de Arquitectura de Software más usada actualmente es de la IEEE Std 1471-2000, la cual plantea: "La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución". [1]

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. [2]

Estilo Arquitectónico: Familia de sistemas de software en términos de su organización estructural. Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. El estilo arquitectónico define las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo de sistemas de software.

Patrón Arquitectónico: Define la estructura básica de una aplicación, provee un subconjunto de subsistemas predefinidos, incluyendo reglas, lineamientos para conectarlos y pautas para su organización, además constituye una plantilla de construcción.

Patrones de Diseño: Soluciones a problemas específicos, basados en la experiencia y que se ha demostrado que funcionan. Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables.

Estilo de Arquitecturas en Capas: Garlan y Shaw [3] 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 que le brinda la inmediatamente inferior.

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. Las capas de la aplicación pueden residir tanto en el mismo nodo físico como en nodos separados.

Patrón arquitectónico Modelo Vista Controlador (MVC): Separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes: El *modelo* en el que se administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado y responde a instrucciones de cambiar el estado. La *vista* la cual maneja la visualización

de la información. Y el *controlador* que interpreta las acciones del *mouse* (ratón) y el teclado, informando al modelo y/o a las vistas para que cambien según resulte apropiado. [4]

Patrones de Diseño

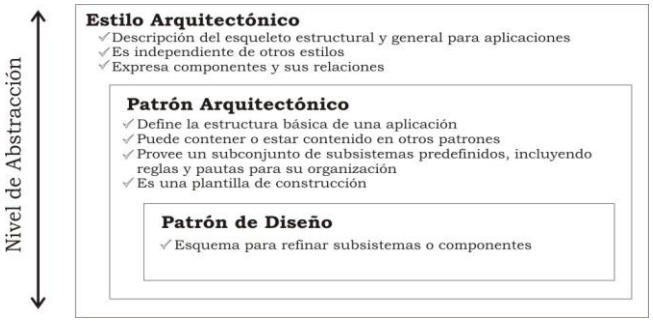
Patrón de diseño Fachada: Este patrón sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Utilizado para reducir la dependencia entre clases, ya que ofrece un punto de acceso al resto de las clases. Si estas cambian o se sustituyen por otras solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones clientes. Fachada no oculta las clases sino que ofrece una forma más sencilla de acceder a ellas.

Patrón de diseño Inyección de Dependencia: (*Dependency Inyection* - DI). Este patrón radica en resolver las dependencias de cada clase (atributos) generando los objetos cuando se arranca la aplicación y luego inyectarlos en los demás objetos que los necesiten a través de los métodos set o del constructor. Es una forma para mitigar la proliferación de dependencias y así fomentar el bajo acoplamiento entre los componentes, y además tiene la intención de reducir la cantidad de código de infraestructura que se debe escribir.

Patrón de diseño DAO: Plantea como solución, utilizar un *Data Access Object* (DAO) para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.

El DAO implementa el mecanismo de acceso requerido para trabajar con la fuente de datos. El DAO oculta completamente los detalles de implementación de la fuente de datos a sus clientes. Como la interfaz expuesta por el DAO no cambia cuando cambia la implementación de la fuente de datos subyacente, este patrón permite al DAO adaptarse a diferentes esquemas de almacenamiento sin que esto afecte a sus clientes o componentes de negocio. Esencialmente, el DAO actúa como un adaptador entre el componente y la fuente de datos. [5]

Relación entre los niveles de abstracción



La imagen muestra la relación de abstracción entre estilos y patrones, estos patrones ayudan al arquitecto a definir la composición y el comportamiento del sistema de software, y una combinación adecuada de ellos permite alcanzar los requerimientos de calidad.

TECNOLOGÍAS Y HERRAMIENTAS

Plataforma de desarrollo: Fue escogida para el desarrollo la plataforma JEE ha sido diseñada para aplicaciones distribuidas con base en componentes o unidades funcionales de software que interactúan entre sí para formar parte de una aplicación empresarial JEE. Un componente de esta plataforma debe formar parte de una aplicación y ser desplegado en un contenedor, o sea, en la parte del servidor JEE que le ofrece al componente ciertos servicios de bajo nivel y de sistema, tales como seguridad, manejo de concurrencia, persistencia y transacciones. La plataforma ofrece un conjunto de APIs de Java para construir aplicaciones, las cuales definen un modelo de programación para las aplicaciones JEE. También ofrece una infraestructura de ejecución para albergar y gestionar aplicaciones. Entre las APIs utilizadas se destacan JSP 2.1, Servlet 2.5, JDBC 3.0, Java Servlet 2.5 y JPA 2.0. Además, como contenedores que representan los períodos de ejecución para gestionar los componentes del sistema se encuentran JEE 6 y Apache Tomcat 6.0.20.

Framework de desarrollo

Framework Dalas 0.2: Framework base de arquitectura para las aplicaciones que utilizan Spring Framework, que actúa como integrador y manejador de la aplicación y fundamenta su funcionamiento bajo la definición de estándares en todas las capas durante el proceso de desarrollo. Este framework permite la orquestación de una solución a través de la integración de componentes reutilizables y la gestión de módulos de negocio. La reducción del consumo de memoria es uno de los principales aportes del framework. Dalas responde a los patrones básicos de diseño Bajo acoplamiento y Alta cohesión.

Framework Spring MVC: Presenta una lógica de diseño bastante sencilla y cuenta con todo el conjunto de librerías de Spring Framework. Dicho framework ofrece una división limpia entre Controladores, Modelos (JavaBeans) y Vistas. Es muy flexible, ya que implementa toda su estructura mediante interfaces. Además, todas las partes del framework son configurables vía plugin en la interfaz. Provee interceptores también como controllers que permiten factorizar el comportamiento común en el manejo de múltiples requests.

Spring Framework 3.0.2: Framework de aplicación de código abierto que ayuda a hacer el desarrollo en JEE mucho más fácil. El mismo interviene en todas las capas arquitectónicas de una aplicación JEE. Spring es modular, lo que significa que pueden utilizarse solo aquellas partes que se necesiten, sin necesidad de recurrir al resto. El mismo se basa en la Inversión de Control, Inyección de Dependencia y el trabajo con POJOs.

Hibernate Framework 3.5: Framework de mapeo objeto/relacional y servicio de consultas para Java. Es la solución ORM más popular en el mundo Java. Es una capa de persistencia objeto/relacional y un generador de sentencias SQL. Es open source. Este framework soporta la mayoría de los sistemas gestores de base de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones JEE y contenedores web.

Framework JasperReport 3.5.2: Framework bastante completo para desarrollar reportes tanto web como desktop en Java. Es una herramienta gratuita y open source que se compone de un conjunto de librerías Java para facilitar la generación de informes en las aplicaciones. Los informes se definen en un fichero xml el cual será compilado por las librerías jasper report y generarán un fichero .jasper que se utilizarán para rellenar y mostrar el informe final.

Entorno de Desarrollo Integrado

Eclipse Galileo 3.5: El entorno de desarrollo Eclipse (también sus plugins) está desarrollado completamente en Java. Emplea módulos (plugins) para adicionar funcionalidades según las necesite el desarrollador. Para la gestión de la configuración y el control de versiones tiene soporte para CVS y Subversion. Además incluye plugins para realizar pruebas de unidad.

Este IDE de desarrollo ha sido el seleccionado por ser totalmente libre y además contar con un robusto mecanismo orientado a plugins, que permite optimizar la calidad del desarrollo y reducir su tiempo de ejecución. No está en contradicción con los atributos de calidad de la arquitectura.

Servidor Web

Apache Tomcat 6.0.20: Software de código abierto implementado para las tecnologías Java Servlet y JavaServer Pages. Apache Tomcat es desarrollado en un entorno abierto y participativo y publicado bajo la licencia del software de Apache.

Se ejecuta en varios Sistemas Operativos. Es un servidor configurable de diseño modular, con diversidad que permiten garantizar una elevada seguridad y buenas prestaciones.

Sistema Gestor de Bases de Datos

PostgreSQL 8.4: Sistema objeto-relacional que incluye características de la orientación a objetos como pueden ser la herencia, tipos de

datos, funciones, restricciones, disparadores, reglas e integridad transaccional. PostgreSQL posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.

Herramienta para el Control de Versiones

Subversion y CVS: Software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Subversion es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn.

Metodología de desarrollo

Rational Unified Process (RUP) Proceso Unificado de Desarrollo: Es un proceso para el desarrollo de un proyecto de un software, que define claramente *quién, cómo, cuándo y qué* debe hacerse en el proyecto. El mismo presenta 3 características fundamentales; centrado en la arquitectura, iterativo e incremental y dirigidos por casos de uso. RUP toma en cuenta las mejores prácticas en el modelo de desarrollo de software.

Lenguaje y herramienta para el modelado

Unified Modeling Language (UML) Lenguaje Unificado de Modelado: Lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

Visual Paradigm: Provee soporte para la generación de código, ingeniería inversa para Java, se integra con Eclipse, para soportar las fases de implementación en el desarrollo de software. Es portable y posee gran facilidad de uso. Dicha herramienta ofrece un entorno de creación de diagramas para UML 2.0 así como diseño centrado en casos de uso. Visual Paradigm es muy potente, gratuito, fácil de instalar, utilizar y actualizar.

Métodos de evaluación de la Arquitectura de Software

ATAM: Método inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM. Revela la forma en que una arquitectura específica satisface ciertos atributos de calidad, y provee una visión de cómo los atributos de calidad interactúan con otros. El método se concentra en la identificación de los estilos arquitectónicos o enfoques arquitectónicos utilizados. Estos elementos representan los medios empleados por la arquitectura para alcanzar los atributos de calidad, así como también permiten describir la forma en la que el sistema puede crecer, responder a cambios, e integrarse con otros sistemas. [6]

DESCRIPCIÓN DE LA ARQUITECTURA

Representación arquitectónica

Representa la Arquitectura de Software como el modelo 4+1 de Philippe Kruchten, vinculado a la metodología de desarrollo RUP

- **Vista de casos de uso:** Muestra los casos de uso que describan alguna funcionalidad importante y crítica, o que impliquen algún requisito importante que deba desarrollarse pronto dentro del ciclo de vida del software.
- **Vista lógica:** Muestra la estructura estática del sistema.
- **Vista de Procesos:** Muestra los hilos y procesos de ejecución así como la comunicación entre estos.
- **Vista de despliegue:** Muestra el despliegue de la aplicación en la red de computadoras.
- **Vista de implementación:** Muestra la estructura en modelos del código del sistema.

Objetivos y restricciones arquitectónicas

El sistema debe ser mantenible - fácilmente analizable-, modificable y corregible. Dentro de las restricciones del sistema que tienen un impacto significativo en la arquitectura se pueden mencionar que:

- Se deben recopilar los datos de las distintas Coordinaciones Regionales (CR) que se encuentran distribuidas geográficamente por todo el país.

- Los datos con los que trabajan las CR son de vital importancia y de carácter estratégico para la prevención del delito del país venezolano, por lo cual es necesario garantizar su seguridad.
- Deberá existir un servidor central en la DGPD en el cual se van a almacenar todos los datos recopilados.
- La cantidad de información que es manipulada es bastante grande.

Requerimientos de Software

Servidores

- Sistema Operativo: Red Hat Enterprise Linux
- Máquina Virtual de Java: JRE 1.6
- Gestor de Base de datos: PostgreSQL 8.3.7
- Servidor web: Apache Tomcat 6.0.20.

PC cliente

- Navegadores: Mozilla Firefox 3.5, Internet Explorer 6 o superior.

Requerimientos de Hardware

Servidores

Tanto el servidor de Base de Datos (PostgreSQL), el servidor de ficheros así como el servidor de aplicaciones, deberán contar como mínimo con:

- RAM: 1GB.
- HDD: 1 GB.
- Procesador: P IV 1.6 GHz.
- NIC: 1x GB

Requisitos de hardware para PC cliente.

- Intel PIII 700 MHz.
- Adaptador de red LAN.
- RAM: 64 MB.
- HDD: 200 MB

Requerimientos de Seguridad

- La autenticación será la primera acción del usuario en el sistema y consistirá en proveer un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica.
- Se debe garantizar que la información sensible sólo pueda ser vista por los usuarios con el nivel de acceso adecuado y que las funcionalidades del sistema se muestren según el usuario que esté activo.

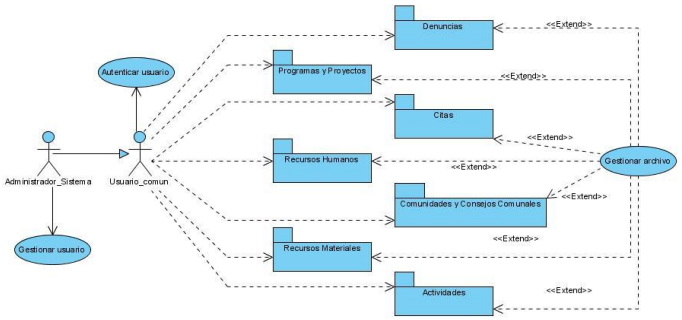
Requerimientos de Usabilidad

El sistema debe permitir a los usuarios un acceso fácil y rápido, contando con un menú que satisfaga las necesidades de los usuarios pues el sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de una computadora y del ambiente web.

Vistas arquitectónicas

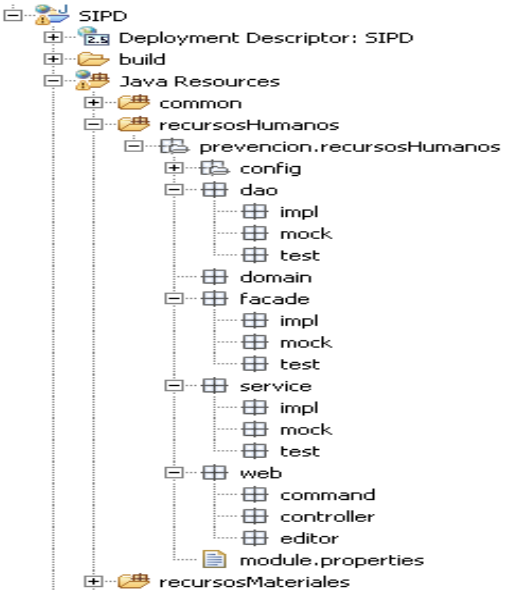
Siguiendo la metodología RUP, se desarrolla la descripción de la arquitectura mediante las 4+1 vista. La esencia de las vistas de la arquitectura es la simplificación o abstracción de los modelos, de los cuales se destacan los detalles más significativos.

Vista de Casos de Uso



Casos de uso significativos para la arquitectura.

Vista Lógica

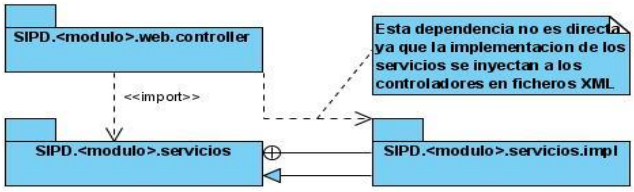


Principales paquetes de la aplicación

Servicios de Fachada.

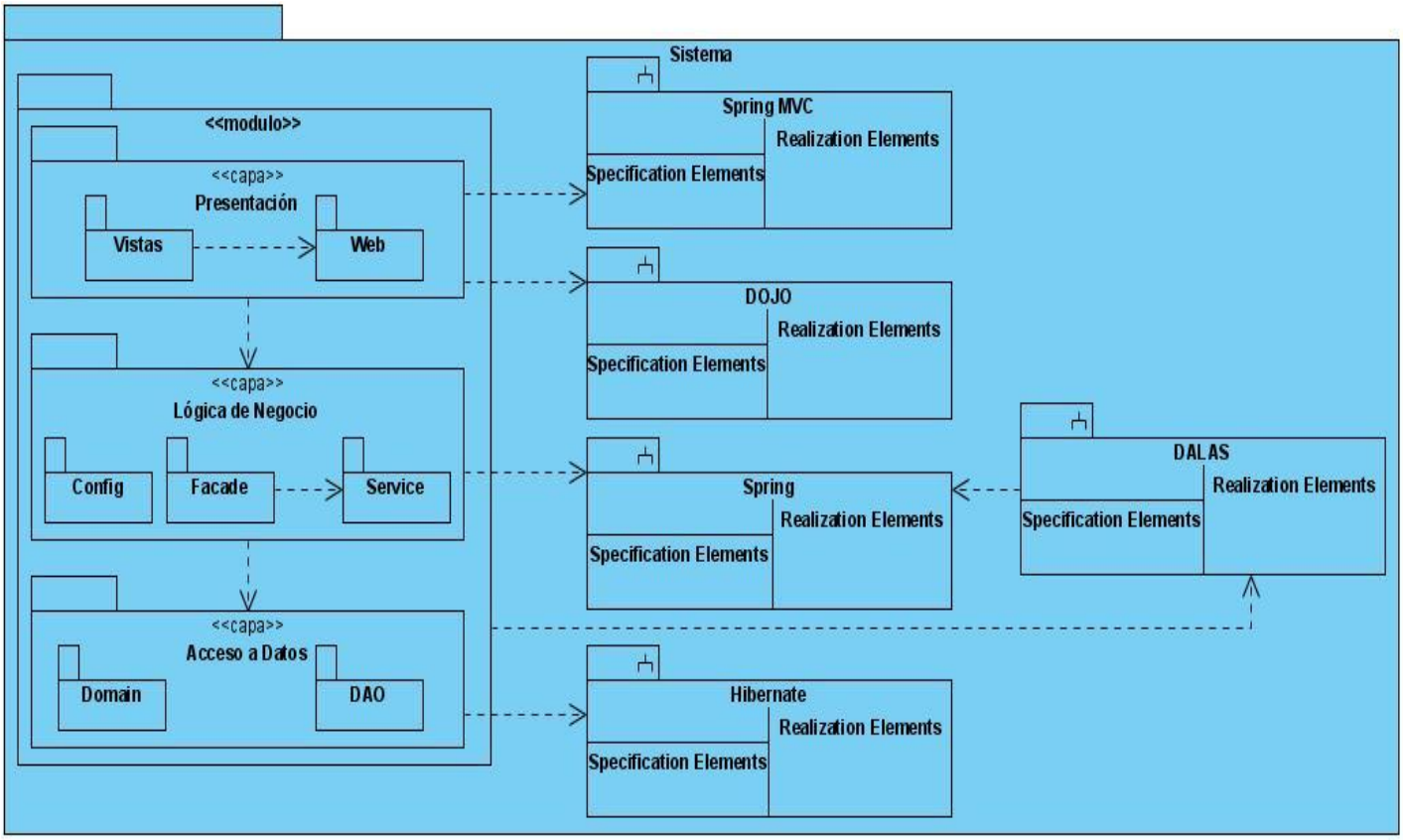
Vista Lógica - Patrón Fachada en la Aplicación Web.

Bajo Acoplamiento e Inyección de Dependencia.

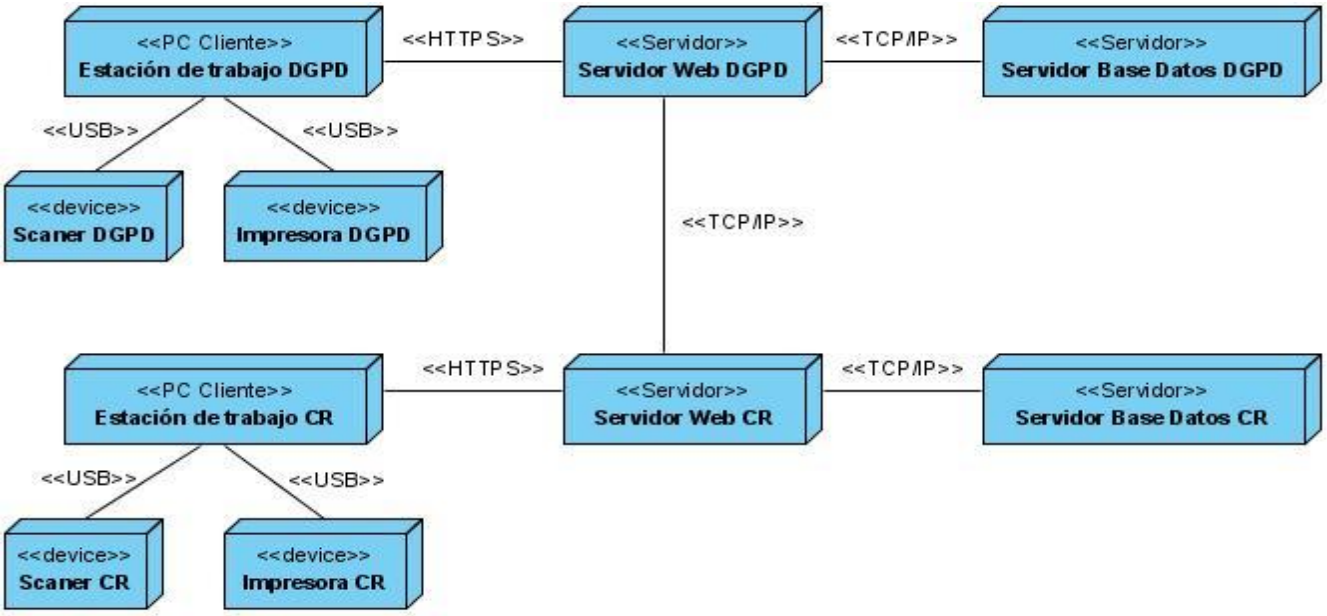


Vista Lógica - Patrones Inyección de dependencia y Bajo Acoplamiento en la Aplicación Web.

Visión general de la arquitectura



Vista de Despliegue



Vista de Implementación

Constituye una selección de los aspectos fundamentales del diagrama de componentes del sistema, el cual modela el empaquetado físico del sistema en unidades reutilizables llamadas "componentes" y sus relaciones. Un componente es una unidad física de implementación que encapsula una o más clases del diseño.

EVALUACIÓN DE LA ARQUITECTURA PROPUESTA

Calidad de la arquitectura

Funcionalidad: Con el uso de la tecnología JEE y el framework Spring que permiten el desarrollo de aplicaciones web de experimentada calidad y seguridad gracias al framework Spring Security de Spring, junto con el estilo arquitectónico en capas, garantizan la habilidad del sistema para cumplir el trabajo para el que fue previsto. Además, es relevante el SGBD PostgreSQL dada la importancia que tiene para la aplicación, la cantidad de datos que se manipulan.

Confiabilidad: La medida de la habilidad del sistema a conservarse activo a lo largo del tiempo se garantiza con el uso del servidor web Apache Tomcat, que es el encargado de restablecer el nivel de desempeño del sistema, también el uso del framework Spring Security de Spring que garantiza el mecanismo de software para manejar las excepciones.

Eficiencia: Dentro de las ventajas que favorecen la decisión arquitectónica de desarrollar una aplicación web está los pocos recursos que la misma demanda de las PC clientes, puesto que realmente la aplicación donde está corriendo es en el servidor web. Además, allí las decisiones arquitectónicas para asegurar el desempeño del sistema como el grado en el cual cumple con las funciones designadas dentro de ciertas restricciones como velocidad, exactitud o uso de memoria, están tomadas con la elección del SGBD PostgreSQL y el framework Hibernate para el acceso a los datos.

Mantenibilidad: La capacidad de que el sistema pueda ser objeto de reparaciones y evoluciones de forma rápida y a bajo costo se garantiza con el uso de los patrones de diseño DAO, Inyección de Dependencia, Fachada, Bajo Acoplamiento y Alta cohesión. Los patrones seleccionados para darle soporte a la capacidad de mantenibilidad del sistema son los mismos que favorecen la habilidad de modificabilidad del mismo.

Portabilidad: La habilidad de que el sistema pueda ejecutarse en diferentes ambientes de computación se garantiza por la tecnología usada en su implementación ya que todas

fueron escogidas bajo muchos criterios pero uno de los principales fue la habilidad de ser multiplataforma, este es el caso de JEE, PostgreSQL y Apache Tomcat. Es preciso esclarecer de que estas herramientas son necesarias en el punto del despliegue de la aplicación en el servidor Web, pero la propia aplicación puede ser accedida por cualquier ordenador que tenga instalado un navegador Web de los especificados independientemente del Sistema Operativo en el que se trabaje.

, los atributos de calidad que se quieren alcanzar en el diseño arquitectónico son los propuestos en el modelo ISO/IEC 9126 adaptado para arquitecturas de software, para lograr los mismos se tomaron las siguientes decisiones arquitectónicas.

Aplicación del método de evaluación seleccionado.

Para llevar a cabo la evaluación dela propuesta se utilizó el método ATAM. Luego de realizados los pasos planteados por el método, se presentan las principales salidas para la evaluación de la arquitectura, estos son: el conjunto de escenarios priorizados, el árbol de utilidad actualizado, se encontraron 8 puntos de sensibilidad, 4 puntos de tradeoff, 11 no riesgos y 3 riesgos.

Se cumplieron las distintas tareas planteadas para el desarrollo exitoso del sistema. Al evaluar la arquitectura propuesta mediante el método ATAM, se pudo observar como la arquitectura satisface las necesidades del sistema a desarrollar al detectarse menos riesgos que no riesgos y pocos puntos de sensibilidad así como puntos de tradeoff. Dichos riesgos no son tan graves para el sistema pues se cuentan con decisiones arquitectónicas para ayudar a mitigar los mismos.

CONCLUSIONES

Al finalizar la investigación se obtuvieron las siguientes conclusiones:

1. Quedó definida la línea base de la arquitectura necesaria para el sistema.
2. Se describió la arquitectura del sistema haciendo uso de las diferentes vistas arquitectónicas.
3. Se especificaron 31 requisitos funcionales arquitectónicamente significativos.
4. Fue evaluada la arquitectura haciendo uso del método de evaluación Architecture Tradeoff Analysis Method (ATAM), arrojando resultados satisfactorios para el sistema.
5. La arquitectura definida cumple con los requisitos de calidad requeridos por el cliente.
6. Las tecnologías y herramientas seleccionadas para suministrar el soporte al desarrollo del sistema, garantizarán que el mismo sea desarrollado en un entorno confiable y seguro.

RECOMENDACIONES

Después de analizados los resultados obtenidos en el presente trabajo, se recomienda lo siguiente:

1. La arquitectura propuesta teniendo en cuenta los atributos de calidad, puede ser utilizada como referencia para aplicaciones web que hagan uso de la tecnología JEE y framework como Spring.
2. Estudiar las decisiones arquitectónicas que constituyen riegos según el método de evaluación propuesto y valorar su sustitución por otras, que no pongan en peligro el sistema.

REFERENCIAS BIBLIOGRÁFICAS

1. **IEEE.** *IEEE Std. 1471-2000, IEEE Recommended Practice for Architectural Description of Software Systems.*
2. **Casanovas, Josep.** [En línea] Septiembre 9, 2004. [Citado: Diciembre 14, 2009.] http://www.desarrolloweb.com/articulos/1622.php
3. **Garlan, David and Shaw, Mary.** *An Introduction to Software Architecture.* 1994. CMU-CS-94-166.

4. **Burbeck, Steve.** *Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC).* 1992.
5. **Lou Torrijos, Ricard.** Programación en castellano. [En línea] Diciembre 14, 2003. [Citado: Enero 16, 2010.] http://www.programacion.com/java/tutorial/patrones2/8
6. **Camacho, Erika, Cardeso, Fabio and Nuñez, Gabriel.** *Arquitecturas de Software.* 2004