Diseño de la Arquitectura del módulo de recepción del Instituto Nacional de Estadística e Informática – Diseño informático del Sistema de Información Estadístico

Diseño de la Arquitectura

**13/09/2017**

Carlos E. León Vela

Consultor especialista en Tecnologías de Información Ingeniero de Sistemas CIP: 136626



Contenido

[Introducción 3](#_Toc493652738)

[Alcance 3](#_Toc493652739)

[Audiencia 3](#_Toc493652740)

[Marco Conceptual 3](#_Toc493652741)

[Siglas y abreviaturas 3](#_Toc493652742)

[Definiciones y Conceptos 4](#_Toc493652743)

[Web service REST 4](#_Toc493652744)

[Cliente web 5](#_Toc493652745)

[Metadata 5](#_Toc493652746)

[CARACTERÍSTICAS DE LA ARQUITECTURA Y DISEÑO DE SOFTWARE DEL MODULO DE RECEPCION DEL INEI Y EL SERVICIO WEB 5](#_Toc493652747)

[Principios de diseño 7](#_Toc493652748)

[De implementación 7](#_Toc493652749)

[Principios de la arquitectura 8](#_Toc493652750)

[Orientado a una arquitectura empresarial 8](#_Toc493652751)

[Comprensible 8](#_Toc493652752)

[Robusto 8](#_Toc493652753)

[Completo 8](#_Toc493652754)

[Consistente 8](#_Toc493652755)

[Estable 8](#_Toc493652756)

[Patrones de Diseño 9](#_Toc493652757)

[Service Façade 9](#_Toc493652758)

[Session Facade 9](#_Toc493652759)

[Business Delegate 9](#_Toc493652760)

[Service Locator 9](#_Toc493652761)

[Data Access Object 9](#_Toc493652762)

[Data Transfer Object 9](#_Toc493652763)

[Componentes de la Arquitectura de Software 10](#_Toc493652764)

[Arquitectura de Detalle 13](#_Toc493652765)

[Persistencia en base de datos 13](#_Toc493652766)

[Transaccionalidad 13](#_Toc493652767)

[Lógica de negocio 14](#_Toc493652768)

[Capa web 15](#_Toc493652769)

[ESTRUCTURAS DEL SERVICIO WEB 15](#_Toc493652770)

[ESTRUCTURA DEL PROYECTO 16](#_Toc493652771)

[Diagrama de Paquetes 16](#_Toc493652772)

[Modelo global – Estructura de proyecto 16](#_Toc493652773)

[Modelo especifico – Estructura de los sub-proyectos: 16](#_Toc493652774)

[SISTEMA INTEGRADO DE ESTADÍSTICAS DE LA CRIMINALIDAD Y SEGURIDAD CIUDADANA 17](#_Toc493652775)

[Componentes de la Arquitectura de Software 17](#_Toc493652776)

[Modificaciones especificas 17](#_Toc493652777)

# Introducción

El presente documento tiene como objetivo detallar la arquitectura y estructura de software del modulo de recepción del INEI y el servicio web REST que expondrá para recibir la información de las entidades participantes, se detallaran los componentes principales, patrones de diseño.

# Alcance

A lo largo del documento, se identifican y describen las herramientas, componentes y/o productos de software que serán empleados para la construcción de la solución; así como patrones de diseño y estructura del desarrollo; como los paquetes, componentes y como será lógicamente desplegados para cada una de las capas de la aplicación para el modulo de recepción de información y el servicio web.

# Audiencia

El contenido del presente documento está dirigido a:

* El equipo de desarrollo, quienes utilizarán este documento como insumo para el proceso de implementación de software.
* El equipo del INEI, los cuales encontrarán en este documento el detalle de la arquitectura de software utilizada en la implementación del modulo de recepción y el servicio web, el mismo que servirá de guía para el posterior mantenimiento

# Marco Conceptual

## Siglas y abreviaturas

Las principales siglas utilizadas a lo largo del presente documento son:

|  |  |
| --- | --- |
| **Sigla** | **Descripción** |
| CRUD | Create Read Update Delete |
| CSS | Cascading Style Sheets |
| DAO | Data Access Object |
| DTO | Data Transfer Object |
| EJB | Enterprise Java Bean |
| IDE | Integrated Development Environment |
| JDO | Java Data Objects |
| JEE | Java Platform Enterprise Edition |
| JPA | Java Persistence API |
| JSF | Java Server Faces |
| JVM | Java Virtual Machine |
| JSTL | Java Server Pages Standard Tag Library |
| JPQL | Java Persistence Query Language |
| JTA | Java Transaction API (JTA) |
| ORM | Object Relational Mapping |
| POJO | Plain Old Java Object |
| SQL | Structured Query Language |

## Definiciones y Conceptos

En esta sección se están incluyendo los conceptos y definiciones más destacados que se verán durante todo el documento.

## Web service REST

Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura en la actualidad se usa en el sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc.) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP. Es posible diseñar sistemas de servicios web de acuerdo con el estilo arquitectural REST de Fielding y también es posible diseñar interfaces XMLHTTP de acuerdo con el estilo de llamada a procedimiento remoto (RPC), pero sin usar SOAP. Estos dos usos diferentes del término REST causan cierta confusión en las discusiones técnicas, aunque RPC no es un ejemplo de REST.

Los sistemas que siguen los principios REST se llaman con frecuencia RESTful.

REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

* Un protocolo cliente/servidor sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST)
* Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Con frecuencia estas operaciones se equiparan a las operaciones CRUD en bases de datos (CLAB en castellano: crear, leer, actualizar, borrar) que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.
* Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.
* El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

## Cliente web

El cliente es una aplicación informática o un ordenador que consume un servicio remoto en otro ordenador conocido como servidor, normalmente a través de una red de telecomunicaciones.1​ También se puede definir un cliente es cualquier cosa (que no sea un servidor) que se conecta a un servidor.2​

El término se usó inicialmente para los llamados terminales tontos, dispositivos que no eran capaces de ejecutar programas por sí mismos, pero podían conectarse e interactuar con computadores remotos por medio de una red y dejar que éste realizase todas las operaciones requeridas, mostrando luego los resultados al usuario. Se utilizaban sobre todo porque su costo en esos momentos era mucho menor que el de un computador. Estos terminales tontos eran clientes de un computador mainframe por medio del tiempo compartido.

Actualmente se suele utilizar para referirse a programas que requieren específicamente una conexión a otro programa, al que se denomina servidor y que suele estar en otra máquina. Ya no se utilizan por criterios de costo, sino para obtener datos externos (por ejemplo páginas web, información bursátil o bases de datos), interactuar con otros usuarios a través de un gestor central (como por ejemplo los protocolos BitTorrent o IRC), compartir información con otros usuarios (servidores de archivos y otras aplicaciones Groupware) o utilizar recursos de los que no se dispone en la máquina local (por ejemplo impresión).

Uno de los clientes más utilizados, sobre todo por su versatilidad, es el navegador web. Muchos servidores son capaces de ofrecer sus servicios a través de un navegador web en lugar de requerir la instalación de un programa específico.

## Metadata

Las metadas son plantillas pre-determinadas y/o definidas por el usuario de negocio con la finalidad de ser asociados a una trasferencia de datos estadísticos que se realice.

# CARACTERÍSTICAS DE LA ARQUITECTURA Y DISEÑO DE SOFTWARE DEL MODULO DE RECEPCION DEL INEI Y EL SERVICIO WEB

El concepto del modulo de recepción conjuntamente con el servicio web ambos del INEI es automatizar el proceso de publicación de estadísticas a nivel nacional.

El uso de una solución web simplifica la comunicación y difusión de la información; permitiendo que todos tengan acceso a ésta, bajo los privilegios que cada usuario tiene asignado, logrando de esta forma procesos ágiles y transparentes.

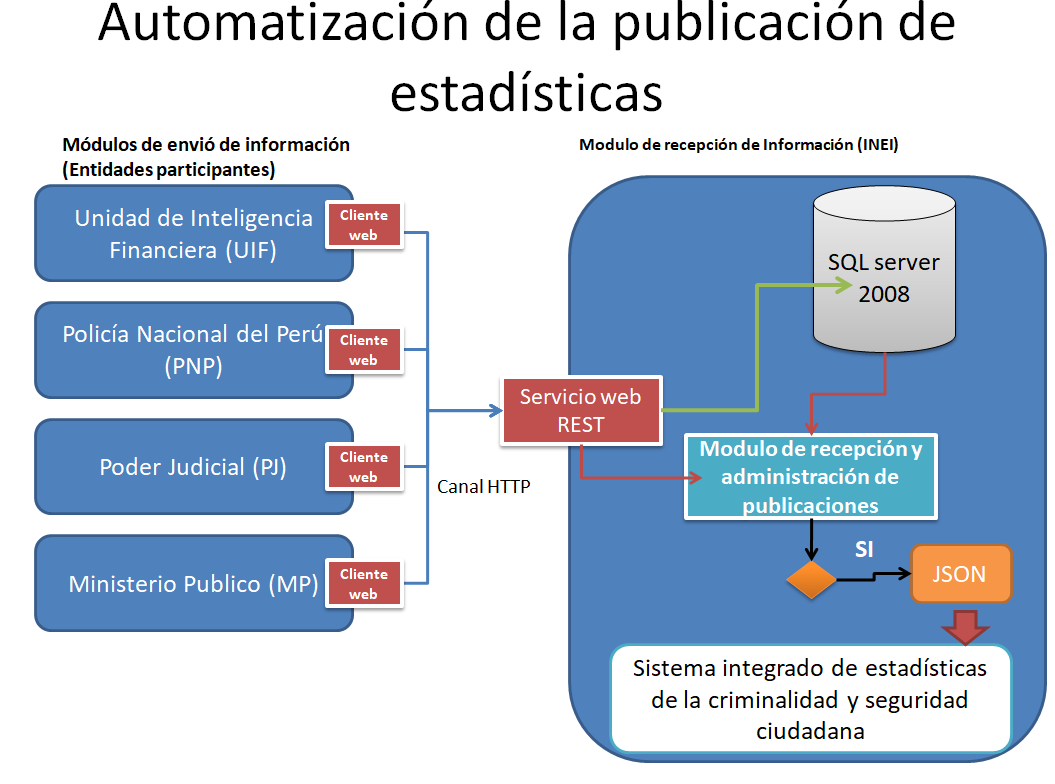
El modulo de recepción y el servicio web contaran con una estructura de proyectos divida en capas especializadas, las cuales manejaran la persistencia de datos, el negocio, controladores y finalmente las vistas, esto para mantener un proyecto ordenado y facilitar el mantenimiento del mismo.

Por otra parte la arquitectura de software propuesta ha considerado otras características importantes, tales como:

1. Los requerimientos funcionales y no funcionales (atributos de calidad interna y externa del software).

2. El ciclo de vida de un proyecto, desde la fase de análisis y diseño hasta la puesta en producción (implementación del sistema web).

3. Las consideraciones técnicas y aspectos adicionales a la implementación que debería tener el sistema



## Principios de diseño

Se han considerado los siguientes principios:

### De implementación

#### Estándar de servicios

La transferencia de mensajes entre los servicios así como su definición está basada en los estándares definidos por la industria en términos tecnológicos.

Para la integración entre los módulos del sistema se utilizará comunicación binaria usando EJB 3.1, como parte de la especificación JEE6. Por otro lado, la construcción de los Web Services de integración con sistemas externos implementa extensiones de XML y JSON, protocolos y prácticas aceptadas como estándares; principalmente: SOAP, WSDL, UDDI, XSD, XSLT y WS-Policy.

#### Reusabilidad de servicios

En el contexto de la arquitectura orientada a servicios, los principios de diseño permitirán la generación de servicios que contarán con la capacidad de combinarse para crear procesos transaccionales de negocio, que serán utilizados por distintos consumidores. De esta forma, estos servicios serán compartidos y reutilizables.

#### Autonomía de servicios

La autonomía de servicios es la capacidad que se le otorga desde su concepción al servicio para su auto gobierno; en la medida que un componente de software exista en un estado de autonomía en tiempo de ejecución es capaz de manejar la lógica independientemente de influencias externas.

#### Stateless de servicios

El principio de servicios sin estados promueve que las operaciones se realicen dentro del contexto de la información de invocación en forma eficiente. Los servicios no deben ser dependientes en el contexto de los estados de otros servicios; dicho principio genera automáticamente flexibilidad, escalabilidad y confiabilidad a los servicios.

#### Enfocado en Patrones

Los patrones de diseño son una pieza clave para el cumplimiento de los principios y los paradigmas de diseño que en sus conceptos proponen, para la implementación de lo que teóricamente se expresa en el framework de diseño de servicios. Los patrones de diseños describen un problema común y su respectiva solución, la cual es documentada para ser usada en forma sistemática.

#### Gobernanza de servicios

La gobernanza de los servicios es resultado de la implementación de un paradigma de diseño, que expone trazabilidad y visibilidad de los activos de software que componen. Los paradigmas de arquitectura y diseño de servicios que componen específicamente soluciones de gobernanza.

#### Manejo transaccional

Mantener un control sobre la integridad de datos, implementado mediante los principios de diseño de los servicios.

#### Persistencia en base de datos relacional

La persistencia de información en el Módulo de recepción y el servicio web, se realizará en base de datos SQL server 2008, utilizando modelos relacionales. La lógica del negocio se implementará en Java, y como puente entre ambas tecnologías se utilizará el estándar Java Persistence API (JPA) implementado por Hibernate.

.

## Principios de la arquitectura

Los principios por los cuales se rige la arquitectura son:

* Requerimientos Funcionales y No funcionales
* Especificación de casos de uso
* Diseño de funcional de prototipos del sistema
* Modelamiento de datos

## Orientado a una arquitectura empresarial

A continuación se describen cada uno de los principios:

Comprensible: los principios subyacentes pueden ser captados y comprendidos por las personas en toda la organización de forma rápida. La intención del principio es claro y sin ambigüedades, de manera que violaciones, ya sea intencional o no, se reducen al mínimo.

Robusto: permitir a las decisiones de buena calidad sobre las arquitecturas y los planes que se hagan, las políticas y normas de obligado cumplimiento que se creen. Cada principio debe ser lo suficientemente definitivo y preciso para apoyar la toma de decisiones coherentes en situaciones complejas y potencialmente controversiales.

Completo: todo principio potencialmente importante que rige la gestión de la información y la tecnología para la organización es definido. Los principios cubren cada situación percibida.

Consistente: la estricta adhesión a un principio puede requerir una interpretación libre de otro principio. El conjunto de principios debe ser expresado de una manera que permite un equilibrio de interpretaciones. Los principios no deben estar en contradicción con el punto en el que se adhiere a un principio que violaría la esencia de otro. Cada palabra en un principio debe elegirse cuidadosamente para permitir una interpretación consistente pero flexible.

Estable: los principios deben ser duraderos, pero capaz de adaptarse a los cambios. Un proceso de enmienda debe ser establecido para añadir, eliminar o alterar los principios después de haber sido ratificados inicialmente.

## Patrones de Diseño

A lo largo de la arquitectura se utilizan diversos patrones de diseño, los cuales se enumeran en este apartado:

### Service Façade

El patrón “Service Façade” es usado para controlar o hacer de fachada que coordine múltiples “Session Façade”. El servicio de la aplicación es usado para aislar la capa de presentación de la capa de negocio, los parámetros del service façade son los primitivos o Data Transfer.

### Session Facade

Permite encapsular la lógica de negocio y la data de negocio, exponiendo una interfaz de forma de reducir la complejidad de los servicios y su lógica interna a los clientes que los consuman. Es utilizado para exponer la lógica de negocio vía servicios EJB Session Stateless.

### Business Delegate

Permite ocultar los mecanismos de acceso a servicios en el cliente que los utilice, Este patrón es implementado mediante interfaces de negocio, la referencia al servicio se implementa mediante el patrón Service locator.

### Service Locator

Patrón que permite centralizar la complejidad y parametrización de la localización de servicios EJB.

### Data Access Object

Patrón de centralización y normalización de acceso a datos, ya sea para acceso a base de datos o a sistemas externos. Para acceso a base de datos, implementa métodos CRUD, permitiendo la manipulación de entidades en base de datos.

### Data Transfer Object

Este patrón permite estructurar la información de negocio y técnica a transferir entre componentes. Los objetos DTO corresponden a contenedores de información y no presenta lógica de negocio.

## 

## Componentes de la Arquitectura de Software

Para el proyecto de construcción del Módulo de recepción y el servicio web se hará uso de los siguientes componentes presentados en el siguiente cuadro:

|  |  |  |  |
| --- | --- | --- | --- |
| **Capa de la Arquitectura** | **Componentes de la Vista de Arquitectura de Referencia** | **Componentes de la Vista de Arquitectura Concreta** | **Explicación** |
| Capa de Presentación | Interfaces graficas de usuario | Java Server Pages 2.2.10 | Framework propio de la especificación JEE que contiene los tags generales el diseño de las páginas web, a su vez que esta versión del JSF se puede inyectar directamente a los elementos HTML tradicionales |
| Bootstrap y jquery | Framework javascript que permite dar el diseño a las interfaces graficas de usuario |
| Reportes | JasperReports Library 5.6 | Este componente es el responsable de utilizar información procedente de cualquier tipo de fuente de datos y producir reportes. |
| Capa de Negocio | Servicios | EJB 3.1 | Este componente se encargara de encapsular la lógica de negocio del módulo de recepción del INEI. Estas viajaran entre capas mediante la inyección de dependencias |
| Capa de integración | Servicios Web / Servicios REST | Especificación JEE | Estos componentes son los encargados de realizar las comunicaciones externas con el INEI |
|
| Capa de Datos | Persistencia | JPA 2.1 - Hibernate | JPA es un framework de mapeo y persistencia para ser usado en arquitecturas de JEE. Algunas de las características principales a destacar son: flexibilidad, no intrusivo, optimizado para aplicaciones de alto y óptimo rendimiento, capacidades de consultas mediante JPQL, SQL nativo |
| Capa de Componentes Transversales | Gestión de errores: Visualización de excepciones, Tratamiento de excepciones | Componente de software implementado como parte de la solución (desarrollo propio) | Este componente es el encargado de gestionar y tratar las excepciones que se produzcan en la ejecución del módulo de recepcion del INEI. Incluye tres características importantes y por ello incluidos dentro de la vista de arquitectura concreta y de referencia: 1) Jerarquía de excepciones: Este componente representa la implementación que se muestra en la sección “Gestión de errores” de este documento. 2) Visualización de excepciones: Este componente permite guardar las excepciones que se generen en los logs de errores. 3) Tratamiento de excepciones: Este componente permite usar la jerarquía de excepciones, la visualización de estos y cuál será la forma general de emplearlos. |
| Gestión de Logging | Especificación JEE | Este componente es el encargado de almacenar los mensajes de error que se hayan producido en el módulo de recepcion del INEI. Existen diversos niveles de error, estos son: 1) DEBUG: este nivel estará activo en el ambiente de desarrollo y prueba (pruebas funcionales), para propósitos de depuración. 2) INFO: este nivel estará activo en el ambiente de prueba (pruebas funcionales), donde se mostrará información que se producen raramente, como inicializaciones de configuración, inicio y fin de tareas muy largas, etc. WARN: este nivel estará activo en el ambiente de prueba (Prueba de Stress), donde mostrará mensajes de alerta para eventos que se desea mantener constancia. La aplicación puede continuar. 3) ERROR: este nivel estará activo en ambiente de producción, para registrar errores o excepciones de aplicación, la aplicación puede continuar, aunque es posible que parte de ella no funcione. Por ejemplo, algún parámetro de configuración no es correcto y se carga el parámetro por defecto. FATAL: este nivel estará activo en ambiente de producción, para registrar errores críticos en la aplicación, la aplicación no puede continuar. Por ejemplo, pérdida de la conexión con la base de datos, etc. |
| Gestión de Memoria | Especificación JEE - memoria de aplicación | Para los @sessionScope se utilizara la memoria de sesión de la aplicación, lo mismo para los @ApplicationScope |
| Capa de Otros Sistemas | Módulo de seguridad | Integracion con la seguridad del INEI | Integracion con el sistema de seguridad del INEI |
|
| Infraestructura | Software Base | SQL server 2008 | Base de datos |
| JBOSS EAP 6.4 | Servidor de aplicaciones |

# Arquitectura de Detalle

En esta sección haremos mención de algunos conceptos ligados con las interacciones con las diversas capas de la arquitectura.

### Persistencia en base de datos

Se utilizará el estándar JPA sobre Hibernate para el mapeo entre objetos Entities Java y base de datos SQL server, utilizando anotaciones Java 7. JPA es el estándar de mapeo relacional y permite reducir el esfuerzo y complejidad de desarrollo.

Las relaciones entre tablas de base de datos se implementarán en Java mediante relaciones entre objetos Entities, mediante referencias a clases padres y listas de clases hijas, como se muestra a continuación:



La implementación de acceso se realizará utilizando el patrón DAO en donde se definirán los métodos CRUD necesarios. Para operaciones en las que se deben aislar completamente transacciones de actualización de registros, se utilizará bloqueo de registros en la base de datos. Se descarta el uso de bloqueo a nivel de Java debido a la naturaleza distribuida (en clúster) de la aplicación en entorno productivo, lo que requiere mecanismos de coordinación entre las distintas JVM. La alternativa de solución es utilizar mediante los drivers JPA un bloqueo optimista y versiones de registros (como se muestra en la anterior figura los atributos versión), lo que permite controlar la última versión vigente de un registro utilizando un número de versión. En la implementación se traduce que estas actualizaciones en desarrollo serán controladas verificando que la versión del dato que requiere actualizarse no ha sufrido modificaciones, control que es automático cuando se utiliza una solución de persistencia como JPA utilizando serialización.

### Transaccionalidad

Para manejo de eventos transaccionales se utilizará EJB Session Stateless Local con transaccionalidad Required, a nivel de base de datos se utilizará el nivel de aislación transaccional Read Commited. En este nivel se permite un compromiso entre aislación transaccional y rendimiento, permitiendo desestimar acciones de SELECT como parte del rollback segment. Esto permite acelerar comandos de SELECT, que son normalmente los comandos más utilizados así como disminuir la carga de bloqueo sobre la base de datos.

El uso del patrón Facade permite tener una exposición de servicios junto con el rendimiento de un único punto de entrada transaccional, implementando internamente la lógica en clases POJO. La transaccionalidad Required permite indicar que el servicio se ejecutará en un contexto transaccional, ya sea iniciando una transacción o acoplándose a una transacción existente.

El contar con un único punto de inicio de transacciones permite optimizar la ejecución en la capa de lógica de negocio y a cada EJB invocado agrega una capa de componentes de control transaccional. En casos de componer un EJB de otros EJB, se agregan capas de control transaccional innecesarias.

El uso de transaccionalidad Required permite al EJB controlar el commit o rollback de las acciones realizadas contra la base de datos, lo que asegura la consistencia de la información en base de datos.

Se considera la utilización de EJBs locales debido a su mejor rendimiento en comparación con EJB remotos tradicionales ya que EJB remotos serializan la información de parámetros y retorno por sobre una capa de comunicación TCP/IP.

El uso de servicios EJB Session Stateless Local permite un mayor performance en la ejecución de servicios, delegando cualquier necesidad de manejo de información de estados a nivel de sesión Web para la interacción con usuario.

### Lógica de negocio

La lógica de negocio del sistema será implementada en clases Java POJO, utilizando el patrón de contrato-implementación definiendo interfaces y desarrollando clases Java que las realicen.

La implementación de la capa de lógica de negocio se implementa en clases “service‟; las cuales manipulan los objetos de dominio para realizar las funcionalidades del sistema y se diagraman a continuación:



### Capa web

La capa web del sistema se implementará con los tags de Java Server Faces como base del DOM y para los estilos de las paginas bootstrap con jquery.

Para la integración entre la capa web y la lógica de negocio se usaran los ManagedBean propios de la especificación JEE, para no impactar en el rendimiento de la aplicación y seguir los estándares de buenas prácticas en la construcción del software, la comunicación con la capa lógica de negocios se realizará por medio de la capa de EJB mediante la inyección de dependencias.

**Integración**

A nivel de integración, se contempla:

* Implementar a través de Web Services.
* Implementar a través de EJB

# ESTRUCTURAS DEL SERVICIO WEB

Para fines prácticos el servicio web será un proyecto el cual se incorpore a la estructura del proyecto del modulo de recepción del INEI, esto es para aprovechar estas estructura y mantener una unicidad del sistema estadístico en relación al INEI.

El servicio web REST estará basado en la especificación de J2EE y mantendra una estructura tal similar al ejemplo:

@POST

@Produces(MediaType.*APPLICATION\_JSON*)

@Consumes(MediaType.*APPLICATION\_JSON*)

@Path("ejemplo/get/")

**public** String getEjemplo(String strJSONDatos, String strJSONEnvio)

**@POST**.- Corresponde a la anotación propia de la especificación de java, esta quiere decir que el servicio solo podrá ser consumido o invocado por métodos post

**@Produces(MediaType.APPLICATION\_JSON).-** Corresponde a la anotación propia de la especificación de java, esta quiere decir que el servicio producirá respuestas síncronas del tipo JSON

**@Consumes(MediaType.APPLICATION\_JSON).-** Corresponde a la anotación propia de la especificación de java, esta quiere decir que los parámetros de entrada del servicio tienen que ser estructuras de datos de tipo JSON

**String strJSONDatos, String strJSONEnvio**.- Parámetros de entrada, los cuales enviaran las entidades, el primero corresponde a la estructura de la información estadística, el segundo la estructura de los datos de envió.

# ESTRUCTURA DEL PROYECTO

## Diagrama de Paquetes

Para la estructura se está tomando como modelo el estándar de capas de estructura de proyecto que a continuación se diagramas:

### Modelo global – Estructura de proyecto



### Modelo especifico – Estructura de los sub-proyectos:

* **pe.gob.module.inei.app**.- Proyecto generador de la aplicación empresarial, contenedor de los demás proyectos.
* **pe.gob.module.inei.be**.- Proyecto encargado de maneja las clases ORM relacionadas la base de datos.
* **pe.gob.module.inei.bp**.- Proyecto que contiene la lógica de negocio de la aplicación.
* **pe.gob.module.inei.bp.service**.- Proyecto que expone el servicio REST
* **pe.gob.module.inei.wd**.- Proyecto que contiene las interfaces graficas de usuario.

# SISTEMA INTEGRADO DE ESTADÍSTICAS DE LA CRIMINALIDAD Y SEGURIDAD CIUDADANA

## Componentes de la Arquitectura de Software

Para el proyecto de construcción de los cuadros estadísticos se hará uso de las siguientes tecnologías:

|  |  |  |  |
| --- | --- | --- | --- |
| **Capa de la Arquitectura** | **Componentes de la Vista de Arquitectura de Referencia** | **Componentes de la Vista de Arquitectura Concreta** | **Explicación** |
| Vista | Interfaces de estadísticas | Jquery y pivot table | jquery.- Framework javascript que facilita la programacion javascript, y pivot table, es una herramiente BI javascript que permite generar tablas dinamicas |
|
|
| Controlador | PHP4 | codeIgniter5 | Framework PHP que cumple el patrón MVC |
| Modelo | PHP4 | codeIgniter5 | Framework PHP que cumple el patrón MVC |

## Modificaciones especificas

Las modificaciones se deben realizar solo en la sección de "Lavado de activos y financiamiento de terrorismo"



**El punto 1**.- Debe mantener el titulo

**El punto 2**.- Debe tener el nombre de la entidad envía las estadísticas

**El punto 3**.- Deben ser los reportes que genera cada una de las entidades, en especial sobre este punto la modificación es que para la obtención de su información debe consultar las tablas del modelo de datos del modulo de recepción del INEI (ver documento de modelo de datos)