|  |  |
| --- | --- |
| NOMBRE DEL PROYECTO | SIGLAS DEL PROYECTO |
| Módulo de envió de estadísticas al INEI | MEEI |

1. **INTRODUCCION**

El presente documento tiene como objetivo detallar la arquitectura y estructura de software del módulo de envió de datos al INEI, se detallaran los componentes principales, patrones de diseño.

1. **PROPOSITO**

El presente documento tiene como principal objetivo realizar el diseño de la arquitectura de software a implementarse sobre en el módulo de envió de datos al INEI, el mismo que deberá ser implementado en la etapa de desarrollo del presente proyecto.

1. **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.

1. **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 de desarrollo de la PNP, los cuales encontrarán en este documento el detalle de la arquitectura de software utilizada en la implementación del módulo de registro de la PNP, el mismo que servirá de guía para el posterior mantenimiento

1. **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.

1. **CARACTERÍSTICAS DE LA ARQUITECTURA Y DISEÑO DE SOFTWARE DEL MODULO DE ENVIO DE DATOS AL INEI**

El concepto del módulo de envió de datos al INEI es automatizar y facilitar el registro de información, a su vez que pueda interconectarse con otras instituciones para compartir información estadística.

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 módulo de envió de datos al INEI contara 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 modulo de envio de datos al INEI.

1. **ARQUITECTURA Y DISEÑO DE SOFTWARE**

conocido por las siglas MSA (del inglés *MicroServices Architecture*) es una aproximación para el desarrollo software que consiste en construir una aplicación como un conjunto de pequeños servicios, los cuales se ejecutan en su propio proceso y se comunican con mecanismos ligeros (normalmente una API de recursos HTTP). Cada servicio se encarga de implementar una funcionalidad completa del negocio. Cada servicio es desplegado de forma independiente y puede estar programado en distintos lenguajes y usar diferentes tecnologías de almacenamiento de datos

Con esta arquitectura se busca desdoblar el proyecto en dos microservicios:

1. Microservicios de negocio: en este proyecto estará toda la logia de negocio expuesta en servicios REST, de modo que las transacciones que se realeizen al no guardar estado y ser ligeras se ejecutaran muy rápido.
2. Microservicio de interfaces.- En este proyecto albergara las interfaces graficas de usuario, como tambien los controladores, los mismos que funcionaran tambien como clientes web

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 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 | HTML 5  JQUERY  BOOTSTRAP | Componentes estándar web, la integración del framework bootstrap con las paginas html5 hacen que la pagina pueda ser vista desde cualquier dispositivo conectado a internet, las reglas de validaciones y acciones que se puedan manejan dentro del cliente usaran el framework de JQUEY |
| Thymeleaf | Framework que permite la comunicación de las paginas html con los controladores. |
| 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 | Spring-boot 1.5.9 | Este conjunto de librerías propia de la solución de spring para los microservicios cuenta los servicios que se usaran en las fachadas de datos y en lo services que serán expuestos |
| Capa de integración | Servicios Web / Servicios REST | Especificación JEE | Estos componentes son los encargados de realizar las comunicaciones externas con el INEI |
| Seguridad /[OAuth2](https://tools.ietf.org/html/rfc6749) |
| Capa de Datos | Persistencia | JPA 2.1 - Hibernate | JPA es un framework de mapeo y persistencia para ser usado en arquitecturas de JEE. Además forma parte de la suit de framework integrador de la solución de spring boot |
| 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. 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 registro de la PNP. 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 | La memoria se gestinara a nivel del proyecto de interfaces, específicamente en las clases controladoras |
| Capa de Otros Sistemas | Módulo de seguridad |  |  |
| Entidades Externas | Sistemas Externos | Este componente representa los sistemas externos a la organización con las que interactuará el módulo de registro de la PNP, entre ellos se encuentran: 1) INEI |
| Infraestructura | Software Base | Oracle Database 12c | Este componente representa el repositorio de datos Oracle DataBase 11g. La aplicación se comunicará con la base de datos a través de un origen de datos en el servidor de aplicaciones. Driver: Oracle JDBC Driver Thin |
| tomcat | Este servidor embebido que forma parte de la solución de spring boot. |

revisión: 04\_\_/10\_\_/\_2016\_\_\_

Fecha de aprobación: 04\_\_/10\_\_/2016\_\_\_\_

|  |  |
| --- | --- |
| *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
| *[Nombres y Apellidos]*  *Usuario del Sistema* | *[Nombres y Apellidos]*  *Coordinador del Proyecto* |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
| *[Nombres y Apellidos]*  *Analista* |

ANEXOS