|  |
| --- |
| Unidad de Modernización y Gobierno Electrónico – Minsegpres |
| PISEE - REST |
| Documentación Técnica |
|  |

**Marzo 2014**

**Elaborado por:**Daniel Troncoso R.

**Revisado por:**

Equipo PISEE

|  |
| --- |
| Documentación técnica de la nueva versión de la aplicación backoffice para la plataforma de interoperabilidad del estado. |
| **Información del Documento:**   |  |  |  |  | | --- | --- | --- | --- | | **Nombre del Informe:** | REST – Documentación Técnica | | | | **Responsable del informe:** | Daniel Troncoso Rojas | | | | **Versión del Documento:** | 1.0 | **Fecha de la versión del documento:** | 10-03-2014 | |  | | | | | **Preparado por:** | Daniel Troncoso Rojas | **Fecha de Creación:** | 10-03-2014 | | **Revisado por:** | Equipo Pisee | **Fecha de Revisión:** |  |   **Control de Cambios:**   |  |  |  |  | | --- | --- | --- | --- | | **Versión:** | **Fecha:** | **Revisado por:** | **Descripción:** | | 1.0 | 10-03-2014 | Daniel Troncoso R. | Creación de documento | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |

[1 Introducción 4](#_Toc382238651)

[2 Situación Actual 5](#_Toc382238652)

[3 Objetivos y Beneficios 6](#_Toc382238653)

[3.1 Objetivo Principal 6](#_Toc382238654)

[3.2 Beneficios 6](#_Toc382238655)

[4 Requerimientos 7](#_Toc382238656)

[5 Tecnologías Utilizadas 8](#_Toc382238657)

[6 Impacto en las Soluciones 10](#_Toc382238658)

[6.1 PISEE – CORE 10](#_Toc382238659)

[6.2 Implementación BUS PISEE 10](#_Toc382238660)

[6.3 Backoffice 10](#_Toc382238661)

[6.4 Base de Datos 11](#_Toc382238662)

[6.4.1 Tablas 11](#_Toc382238663)

[6.4.2 Secuencias 12](#_Toc382238664)

[7 Arquitectura del Proyecto 13](#_Toc382238665)

[7.1 Proyecto: rest-business 13](#_Toc382238666)

[7.2 Proyecto: rest-web 14](#_Toc382238667)

[8 Archivos de Configuración 17](#_Toc382238668)

[8.1 Plantillas de Servicios 17](#_Toc382238669)

[8.2 Properties 18](#_Toc382238670)

[9 Configuración de Entorno de Desarrollo 19](#_Toc382238671)

[9.1 Instalación Proyectos 19](#_Toc382238672)

[9.2 Ejecutar Proyecto Web 19](#_Toc382238673)

[10 Indicaciones para Deploy 21](#_Toc382238674)

[10.1 Checklist de Instalación 21](#_Toc382238675)

[11 Glosario de Términos 22](#_Toc382238676)

[12 Anexos 23](#_Toc382238677)

[12.1 Archivo de Propiedades: PiseeContingencia\_REST.properties 23](#_Toc382238678)

# Introducción

En la constante búsqueda de cómo ayudar de mejor forma a los diferentes organismos que utilizan la PISEE, es que el área de interoperabilidad decidió desarrollar una nueva aplicación para el consumo de los servicios actualmente integrados.

Actualmente la utilización de REST (Representational State Transfer o Transferencia de Estado Representacional) ya es un “estándar” de facto para las aplicaciones web, vemos como empresas de diferentes ámbitos como Facebook a la Banca en línea utiliza esta técnica para entregar y comunicar información ya sea interna como a usuarios finales, en pocas palabras la idea es visitar una URI común y visualizar de forma rápida la información lo que se traduce en mejoras de lectura y procesamiento de la información.

En el ámbito de la PISEE con esta técnica es posible que nuevos organismos se integren a la plataforma como consumidores de servicios ya dispuestos en un tiempo mucho menor del actual lo que se traduce en un ahorro de dinero para todos los organismos.

# Situación Actual

Los servicios integrados en la PISEE cumplen con la normativa legal que dice que toda la información sea vía SOAP, pero a la espera del cambio de los decretos que norman dicho comportamiento es que se comenzó el análisis y posterior desarrollo de una aplicación que tomando como fuente el consumo de servicios de la plataforma fuese capaz de transformarlos a REST.

Para algunas instituciones persisten los problemas para integrarse a la plataforma acudiendo a variados excusas, tales como:

* Falta de personal capacitado.
* Falta de recursos para contratar proveedores a sus desarrollos.
* Tiempos excesivos en la integración.

Por otro lado existe una dependencia exclusiva con el protocolo SOAP para el consumo de servicios web con lo cual se ha dejado de estar actualizados en nuevas tecnologías y/o técnicas las cuales favorecen en todo aspecto a los nuevos desarrollos, ya sea en tiempos de creación como en la disminución de sus costos.

# Objetivos y Beneficios

Una vez identificada la situación actual junto con los requerimientos que debe resolver la nueva versión de la aplicación, es que se planteó el objetivo principal y una serie de beneficios que incluye este nuevo desarrollo.

## Objetivo Principal

Disponer de una nueva técnica (REST) para la entrega de información de los servicios integrados en la PISEE que permita disminuir los tiempos de integración de los organismos consumidores.

## Beneficios

* Actualización de tecnología en la plataforma actual.
* Disponer de una nueva vía para exponer los servicios SOAP.
* Acortar tiempos de integración de los organismos consumidores.
* Disminución de costos en la integración a la PISEE.
* Aumento de transacciones en las Plataforma.

# Requerimientos

A continuación se mencionan los requerimientos que la aplicación debe cumplir.

* REQ001: El sistema debe proveer los servicios integrados en la PISEE en formato vía REST.
* REQ002: El sistema debe contar con un administrador de TOKEN en donde se asocie el servicio a consumir por cierto tramite.
* REQ003: El sistema debe contar con un reporte que permita visualizar los tipos de transacciones ya sean SOAP o REST.
* REQ004: El sistema debe contar con un mecanismo de contingencia.
* REQ005: El sistema debe permite bloquear el consumo de servicios vía REST en cualquier momento sin detener la aplicación.
* REQ006: El sistema debe permitir una integración sencilla y rápida servicios SOAP de la PISEE.
* REQ007: El sistema debe permitir integrar los servicios con OAUTH.

# Tecnologías Utilizadas

En esta sección se detallan las tecnologías utilizadas tanto en la construcción de la aplicación como las tecnologías que debían de utilizarse dado que ya se contaban con estas y formaban parte de las condicionantes de diseño.

En la construcción de la aplicación se utilizaron las siguientes tecnologías:

* **Jersey** (Versión 1.17.1): Es una implementación de la API para REST Web Services (JAX-RS) la cual proporciona soporte en la creación de servicios web de acuerdo con el estilo arquitectónico Representational State Transfer (REST).1 Utiliza anotaciones, introducidas en Java SE 5, para simplificar el desarrollo y despliegue de los clientes y puntos finales de los servicios web.
* **GSON** (Versión 2.2.4): Es una biblioteca de código abierto para el lenguaje de programación Java que permite la serialización y deserialización entre objetos Java y su representación en notación JSON.
* **Spring** (Versión 2.5.6):Es un framework para el desarrollo de aplicaciones de código abierto para la plataforma Java, sus características principales son inyección de dependencias y programación orientada a aspectos, se encuentra dividido en diferentes módulos los cuales solucionas los problemas típicos de una aplicación tales como, DAO, WS, Testing, ORM.
* **Maven** (versión 2.2.1) – (<http://maven.apache.org>): Es una herramienta de software para la gestión y construcción de proyectos, es similar en funcionalidad a Apache Ant, pero tiene un modelo de configuración de construcción más simple, basado en un formato XML. Utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.
* **JPA** (versión 1.0) – (<http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html>): Es la API de persistencia desarrollada para la plataforma Java EE, corresponde a un framework que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE). Cubre tres áreas: La API en sí misma (definida en el paquete javax.persistence), La Java Persistence Query Language (JPQL) y Metadatos objeto/relacional.
* **Hibernate** (versión 3.6.10.Final) – (<http://www.hibernate.org>): Es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.
* **Bootstrap** (versión 2.2.2) – (<http://twitter.github.com/bootstrap>): Es un framework diseñado para simplificar el proceso de creación de diseños web. Para ello ofrece una serie de plantillas CSS y de ficheros JavaScript, los cuales permiten conseguir: Interfaces que funcionen de manera correcta en los navegadores actuales, un diseño que pueda ser visualizado de forma correcta en distintos dispositivos (a distintas escalas y resoluciones) y una mejor integración con librerías que suelen ser utilizadas habitualmente, como por ejemplo jQuery.
* **JQuery** (versión 1.3.2) – (<http://jquery.com>): Es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Por otro lado, existían una serie de restricciones tanto en el diseño como en la construcción de la nueva aplicación, éstas se mencionan a continuación:

* JBoss Application Server (versión 5.2) – (<http://www.jboss.org/jbossas>)
* JBoss IDE (versión 4.1.2.GA) – (<http://www.jboss.org/jbosside>)
* Oracle (versión 10g) – (<http://www.oracle.com>)

# Impacto en las Soluciones

Como se puede identificar, ésta aplicación impacta a toda a toda la Plataforma dado que no es solo una aplicación más sino que es otra forma de proveer servicios por lo cual la importancia y el impacto es enorme. A continuación se describen todos los cambios en los diferentes componentes de la Plataforma.

## PISEE – CORE

Con objeto de tener una base a para futuros desarrollos e ir incrementando la lógica de la Plataforma es que se decidió crear este nuevo proyecto el cual sirve como acceso a toda la base de datos para entregar lógica a nivel de JAVA. Actualmente este proyecto se utiliza en Backoffice y ahora en la nueva aplicación PISEE REST.

Finalmente y para poder identificar y magnificar de forma correcta este proyecto es que se generará nueva documentación enfocada solo a este proyecto.

## Implementación BUS PISEE

En la actual implementación de la PISEE se debieron hacer una serie de cambios enfocados, en gran parte, a poder diferencia si el tipo de transacción fue consumido desde un formato SOAP o en el nuevo formato REST.

* **Contingencia**: Los cambios están enfocados en la creación de un nuevo archivo de propiedades donde se encuentra la configuración base para realizar el consumo de los servicios SOAP integrados en la plataforma, de esta forma una vez que se active contingencia la nueva aplicación va a leer ese archivo para poder consumir los servicios.
* **Trazabilidad**: Para poder diferenciar las transacciones de servicios SOAP y servicios REST se realizaron modificaciones para que éstas últimas queden registradas en la base de datos con el tipo “R” cuando sean consultadas desde la nueva aplicación REST.

## Backoffice

En esta aplicación se realizaron modificación enfocadas a poder vislumbrar la diferenciación de transacciones y los insumos previos a la integración de un servicio SOAP para exponerlo vía REST como es el caso de la generación del TOKEN.

* **Reporte de Transacciones por Organismo**: Con el nuevo tipo de transacciones es necesario poder identificarlas, es por esto que se realizaron modificaciones tanto a nivel de capa de presentación como de negocio para poder clasificar las transacciones.
* **Mantenedor TOKEN**: Tal como se mencionó anteriormente, una parte importante del proyecto consiste en la creación de un TOKEN para asociar el tramite con el servicio a consumir, es por esto que se realizó este funcionalidad.
* **Refactoring**: Se realizó un cambio global en la aplicación generado por la creación de un nuevo proyecto (PISEE - CORE) con objeto de poder separar de forma correcta las diferentes responsabilidades de cada proyecto.

## Base de Datos

A continuación se describen todos los cambios en la actual base de datos para que soporte la solución REST, los cambios fueron tanto a nivel de tabla como de secuencias.

### Tablas

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabla** | CFG\_CONFIG\_SERVICIO | | |
| **Campo** | **Tipo Dato** | **Largo** | **Descripción** |
| CFG\_ID | NUMBER | 18 | Identificador del registro. |
| CFG\_TOKEN | VARCHAR2 | 10 | TOKEN asociado al consumidor de un servicio en particular |
| CFG\_OPERATION | VARCHAR2 | 50 | Corresponde la operación del servicio integrado en la plataforma |
| CFG\_ENDPOINT | VARCHAR2 | 255 | Endpoint del servicio en la PISEE |
| CFG\_TIMEOUT | NUMBER | 5 | Cantidad de Mili segundos en el consumo del servicio |
| CFG\_HTTP\_USERNAME | VARCHAR2 | 50 | Nombre de usuario del organismo consumidor, para la conexión HTTPS |
| CFG\_HTTP\_PASSWORD | VARCHAR2 | 50 | Contraseña de la cuenta de usuario del organismo consumidor |
| CFG\_CODIGO\_INSTITUCION | VARCHAR2 | 10 | Código de la institución consumidora del servicio |
| CFG\_CODIGO\_TRAMITE | VARCHAR2 | 10 | Código del tramite |
| STR\_ID | NUMBER | 18 | ID de la tabla que asocia el Trámite y el servicio. |
| CFG\_ESTADO | VARCHAR2 | 1 | Indica si se puede consumir el servicio vía REST, valores posibles 1=SI, 0=NO |
| CFG\_REQUIERE\_FIRMA | VARCHAR2 | 1 | Indica si dicho servicio requiere firma digital, valores posibles 1=SI, 0=NO |
| CFG\_REQUIERE\_OAUTH | VARCHAR2 | 1 | Indica si dicho servicio se puede consumir vía OAUTH, valores posibles 1=SI, 0=NO |
| COA\_ID | NUMBER | 18 | Clave foránea de la tabla COA\_CONFIG\_OAUTH |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabla** | CFG\_CONFIG\_OAUTH | | |
| **Campo** | **Tipo Dato** | **Largo** | **Descripción** |
| COA\_ID | NUMBER | 18 | Identificador del registro. |
| COA\_CLIENT\_ID | VARCHAR2 | 255 | Corresponde al campo client\_id según especificación OAUTH |
| COA\_CLIENT\_SECRET | VARCHAR2 | 255 | Corresponde al campo client\_secret según especificación OAUTH |
| COA\_NAME | VARCHAR2 | 255 | Corresponde al campo name según especificación OAUTH |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabla** | COR\_ OAUTH\_REDIRECT | | |
| **Campo** | **Tipo Dato** | **Largo** | **Descripción** |
| COR\_ID | NUMBER | 18 | Identificador del registro. |
| COR\_REDIRECT\_URI | VARCHAR2 | 255 | Corresponde al campo redirect\_uri según especificación OAUTH |
| COA\_ID | NUMBER | 18 | Referencia a la tabla CFG\_CONFIG\_OAUTH |

### Secuencias

Para todas las tablas creadas, es decir, CFG\_CONFIG\_SERVICIO, COA\_CONFIG\_OAUTH y COR\_OAUTH\_REDIRECT se creó una secuencia con el objetivo de crear un único identificador para cada registro, la secuencia creada es: SEQ\_CONFIG\_SERVICIO.

# Arquitectura del Proyecto

Con el objetivo de dividir las diferentes lógicas de negocio de la aplicación es que el proyecto se dividió en dos subproyectos cada uno con funcionalidades totalmente acotadas pero que en su conjunto logren satisfacer los requerimientos propuesto en un principio. A continuación se detalla cada uno de los proyectos que forman parte de la aplicación.

## Proyecto: rest-business

El objetivo de este proyecto es contener la lógica de negocio de la aplicación. Dentro de esta lógica existen diferentes paquetes cada uno con un objetivo definido. En la siguiente imagen se puede identificar la estructura del proyecto.

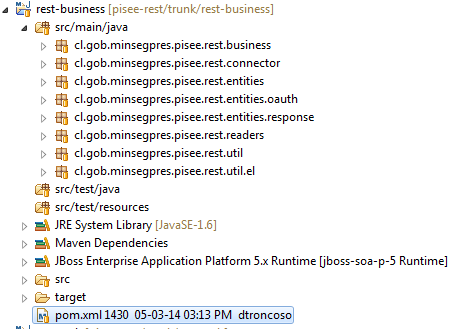


Figura X - Estructura proyecto rest-business.

Tal como se puede identificar, existen varios “packages” que aíslan ciertos patrones de diseño que están incluidos en la aplicación, todos estos están dentro de “cl.gob.minsegpres.pisee.rest”. El detalle de cada uno de estos es el siguiente:

* Package “business”: Aquí se encuentran todas las clases que tienen lógica del negocio, tales como: la creación del ID sobre, el objetivo es tener aislado las clases que tengan código para funcionalidades específicas.
* Package “connector”: Este package contiene las clases que finalmente se comunican a los Web Services SOAP de la Plataforma, para ciertos casos tales como SRCeI y SII hay conectores especiales dado la lógica de dichas instituciones proveedores.
* Package “entities”: Este package contiene las entidades o POJO utilizadas en el proyecto el objetivo es mantener estas clases apartadas para una correcta visualización de los mensajes enviados en el proyecto.
* Package “oauth”: Contiene las entidades creadas para los servicios que utilizan OAUTH.
* Package “response”: Son las clases que encapsulan el mensaje de respuesta que se entrega una vez solicitado un servicio REST, es decir, acá se encuentran las clases que son transformadas a JSON.
* Package “readers”: Aca se encuentra las clases que se encargan de leer las plantillas XML para el consumo de servicios web de la Plataforma actual (SOAP).
* Package “util”: Clases utilitarias ocupadas por el proyecto.
* Package “el”: Clases que contienen la lógica para utilizar “Expression Language” (EL) con lo cual es procesado las plantillas para el consumo de servicios.

## Proyecto: rest-web

Este es el proyecto web donde normalmente se encuentran todos los archivos que son parte de la aplicación web en sí, tales como: xhtml, css, js, imágenes y los archivos de configuración del proyecto JEE, pero por la funcionalidad que implica solo contiene clases JAVA y archivos de configuración lo que permite procesar las peticiones de los servicios tipo REST.

La estructura del proyecto se puede visualizar en la siguiente imagen.

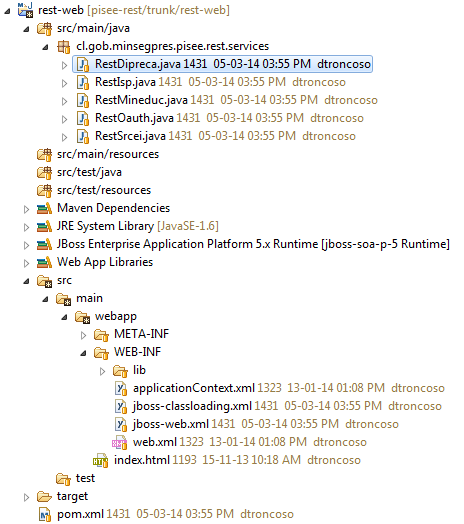


Figura X - Estructura proyecto rest-web.

La descripción de las carpetas más importantes se detalla a continuación:

* “src/main/java”: Tal como lo indica la administración de proyectos de Maven, en esta carpeta se encuentran todas las clases específicamente en el package “cl.gob.minsegpres.pisee.rest.services”, todas las clases que tienen la lógica para los servicios REST, cabe señalar que por cada proveedor de servicios web en la Plataforma existe una clase.
* “src/main/webapp/WEB-INF”: En esta carpeta se encuentran los archivos de configuración los cuales son los siguientes:
  + applicationContext.xml: Archivo de Spring el cual contiene la configuración de las clases JAVA que resuelven las consultas de los servicios.
  + jboss-classloading: Es el archivo de JBOSS con el cual se puede configurar el Class Loader de la aplicación y la forma en que el servidor tome las librerías.
  + joss-web.xml: Es un archivo de JBOSS en donde se indica el nombre del contexto de la aplicación.
  + web.xml: Archivo descriptor del estándar JEE en donde se encuentra la configuración inicial tanto de Spring como Jersey.

# Archivos de Configuración

A continuación de detallan todos los archivos de configuración utilizados por la aplicación:

## Plantillas de Servicios

En esta sección se describen los archivos que se utilizan para el consumo de los servicios SOAP que actualmente se encuentran integrados en la Plataforma, cabe señalar que solo se describe el objetivo para el cual se utilizan y **NO el detalle de su contenido dado que esto se encuentra en el Manual para Nuevas Integraciones, ubicado en:**

http://www.app.pisee.cl/svn/PISEE/pisee-rest/trunk/documentacion/manual\_nuevas\_integraciones

Para cada servicio es necesario configurar dos archivos: el primero es para crear el XML de entrada al web service SOAP integrado en la Plataforma y el segundo es para procesar la respuesta de este servicio extrayendo los campos que se desea retornar en el servicio REST. Finalmente y solo para los servicios que necesiten firma digital, como es el caso de SRCEI, es necesario contar con un tercer archivo XML el cual se utiliza para configurar la información del certificado que se utiliza para firma y desencintar la respuesta del servicio.

El nombre de archivo para estos casos se divide en dos secciones en donde la primera indica el servicio para el cual se utiliza la plantilla y la segunda sección indica el tipo de archivo, de esta forma se pueden encontrar los siguientes casos:

* SRCEI\_CertificadoNacimientoGENCHI**\_input.xml**: Corresponde al archivo que se utiliza como plantilla para consumir los servicios SOAP de la Plataforma actual.
* SRCEI\_CertificadoNacimientoGENCHI**\_output.xml**: Corresponde a la configuración para procesar la respuesta del servicio.
* SRCEI\_CertificadoNacimientoGENCHI**\_keystore.xml**: corresponde a la configuración del keystore para los servicios que lo requieran.

Tal como se puede observar en el ejemplo, el texto marcado en rojo indica para que se utiliza el archivo, encantándose los siguientes casos: “\_input.xml”, “\_output.xml” y “\_keystore.xml”.

## Properties

En caso de que se active contingencia en la Plataforma el archivo **PiseeContigencia\_REST.properties** toma vital importancia dado que en él se encuentran la configuración de los servicios SOAP que se consultan a la Plataforma. El contenido de este archivo es básicamente la tabla CFG\_CONFIG\_SERVICIO el cual contiene el siguiente orden de campos:

TOKEN||ENDPOINT||OPERATION||TIMEOUT||HTTP\_USER\_NAME||HTTP\_PASSWORD||CODIGO\_INSTITUCION||CODIGO\_TRAMITE||ESTADO||REQUIERE\_FIRMA||SERVICIO\_ORGANISMO\_SIGLA||SERVICIO\_ORGANISMO\_NOMBRE\_CORTO||TRAMITE\_ORGANISMO\_SIGLA||TRAMITE\_ORGANISMO\_NOMBRE

El archivo se debe encontrar en la ruta **%JBOSS\_SERVER%**\deploy\base\piseeConf\PiseeContigencia\_REST.properties. En donde **%JBOSS\_SERVER%** indica la ruta donde se encuentra instalado JBOSS y su correspondiente instancia.

# Configuración de Entorno de Desarrollo

La configuración del proyecto cumple con el mismo procedimiento que el proyecto Backoffice, el detalle de este procedimiento se describe en el documento: [http://www.app.pisee.cl/svn/PISEE/backoffice/trunk/documentacion/tecnica/ [backoffice] - documentación técnica v.1.5.docx](http://www.app.pisee.cl/svn/PISEE/backoffice/trunk/documentacion/tecnica/%20%5bbackoffice%5d%20-%20documentacion%20técnica%20v.1.5.docx), “Capitulo 9 – Configuración de Entorno de Desarrollo”

## Instalación Proyectos

Desde el repositorio se deben importar los siguientes proyectos:

* **rest-business**: La ruta en el repositorio es http://www.app.pisee.cl/svn/PISEE/pisee-rest/trunk/rest-business
* **rest-web**: La ruta en el repositorio es http://www.app.pisee.cl/svn/PISEE/backoffice/trunk/rest-web

## Ejecutar Proyecto Web

Finalmente para ejecutar el proyecto en el servidor local, se debe seleccionar el proyecto “rest-web” y en el menú contextual ir a “Run As > Run on Server” tal como lo indica la figura:

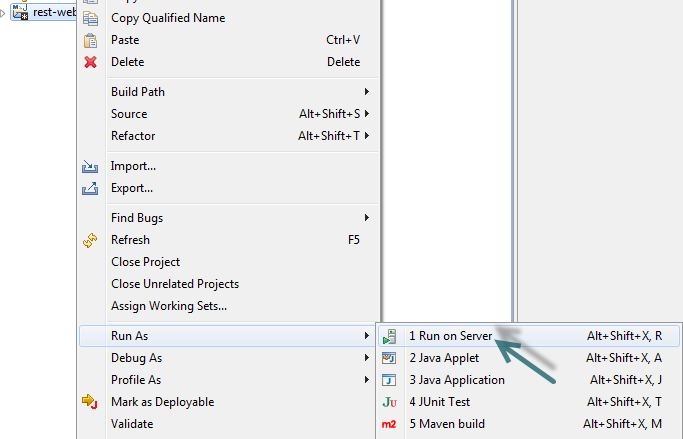


Figura X – Ejecutar Proyecto Web.

# Indicaciones para Deploy

Las indicaciones para generar e instalar la aplicación son los mismos descritos para el proyecto de Backoffice con la salvedad de que ahora los proyectos a instalar son “rest-business” y “rest-web”.

El detalle del procedimiento se encuentra en el documento La configuración del proyecto cumple con el mismo procedimiento que el proyecto Backoffice, el detalle de este procedimiento se describe en el documento: [http://www.app.pisee.cl/svn/PISEE/backoffice/trunk/documentacion/tecnica/ [backoffice] - documentación técnica v.1.5.docx](http://www.app.pisee.cl/svn/PISEE/backoffice/trunk/documentacion/tecnica/%20%5bbackoffice%5d%20-%20documentacion%20técnica%20v.1.5.docx), “Capitulo 10 – Indicaciones para Deploy”

## Checklist de Instalación

Las indicaciones para generar e instalar la aplicación son los mismos descritos para el proyecto de Backoffice con la salvedad de que ahora los proyectos a instalar son “rest-business” y “rest-web”.

# Glosario de Términos

El siguiente es el glosario de los términos utilizados en el presente documento.

* JEE: Java Edición Empresarial (Java Enterprise Edition).
* IDE: Entorno de Desarrollo Integrado (Integrated Development Environment).
* API: Interfaz de Programación de Aplicaciones (Application Programming Interface).
* DAO: Data Access Object (Objeto de Acceso a Datos).
* EAR: Archivo Empresarial (Enterprise Archive).
* Package: Paquete de clases Java.
* JNDI: Interfaz de Nombrado y Directorio Java (Java Naming and Directory Interface).
* ESB: bus de servicios de empresa (Enterprise Service Bus).
* JSON: JavaScript Object Notation
* SVN: Subversion.
* SOAP:
* REST:
* EL:
* JSON:
* OAUTH:
* XML:

# Anexos

## Archivo de Propiedades: PiseeContingencia\_REST.properties

|  |
| --- |
| #TOKEN||ENDPOINT||OPERATION||TIMEOUT||HTTP\_USER\_NAME||HTTP\_PASSWORD||CODIGO\_INSTITUCION||CODIGO\_TRAMITE||ESTADO||REQUIERE\_FIRMA||SERVICIO\_ORGANISMO\_SIGLA||SERVICIO\_ORGANISMO\_NOMBRE\_CORTO||TRAMITE\_ORGANISMO\_SIGLA||TRAMITE\_ORGANISMO\_NOMBRE  Xo7WkT3||https://www.pisee.cl/MINEDUCDispoLicenciaEnsenanzaMediaProxy||solicitarTramiteOp||50000||CCHEN||eS0PhVOu||666666||4444||1||0||MINEDUC||DISPONIBILIDAD LICENCIA ENSENANZA MEDIA||CCHEN||SOLICITUD DE INFORMACION LICENCIA MEDIA |