|  |
| --- |
| visión ingenieria  Imagen que contiene captura de pantalla  Descripción generada con confianza alta |
| MANUAL TÉCNICO CAMBIO DE DOMICILIO |
|  |
|  |
| **Visión Ingeniería** |
| **09/12/2020** |

|  |
| --- |
| Este documento contiene la descripción de las modificaciones realizadas a nivel de proyecto y de base de datos. |

Contenido

[DESARROLLO 2](#_Toc58439229)

[**1.** **Registro Mercantil y ESALES** 2](#_Toc58439230)

[**2.** **Inicio Proceso de Estudio** 3](#_Toc58439231)

[**3.** **Aprobación Inscripción** 4](#_Toc58439232)

[**4.** **Cambio de Domicilio** 5](#_Toc58439233)

[**5.** **Job S3** 7](#_Toc58439234)

[**6.** **Grabación Comerciante** 7](#_Toc58439235)

[**7.** **Reporte Cambio Domicilio Proponentes** 9](#_Toc58439236)

[**8.** **Actualización de noticia y estado a RUES** 10](#_Toc58439237)

[**9.** **Incorporación Transacción Rues** 11](#_Toc58439238)

[**10.** **PreRegistro RUP** 12](#_Toc58439239)

[**11.** **Modelo de datos** 14](#_Toc58439240)

[**12.** **Generación Json** 17](#_Toc58439241)

[**13.** **Cargue Json** 20](#_Toc58439242)

INTRODUCCIÓN

En este documento se realiza la especificación técnica de las modificaciones realizadas en el desarrollo realizado para la implementación del cambio de domicilio.

# DESARROLLO

1. **Registro Mercantil y ESALES**

El primer escenario corresponde al caso cuando la empresa domiciliada en la cámara de comercio de Bucaramanga decide cambiar de domicilio y realiza la solicitud de traslado directamente en las oficinas de nuestra jurisdicción

* 1. En la opción ventanilla 🡪 atención al público una vez se esté liquidando un código de documento 23 o 240 el sistema debe verificar si la empresa esta renovada a tiempo. Si es antes de la fecha máxima de renovación del año vigente entonces se entiende por renovado a tiempo que haya renovado el año inmediatamente anterior, de lo contrario ya debe tener renovación del año actual. Se mostrará un mensaje de advertencia, pero dejará radicar.

Si el estado de una empresa es disolución por vigencia, disolución anticipada o pérdida de calidad de comerciante se debe validar la fecha de la tabla tb\_suspension para saber si su última renovación se hizo a tiempo.

* En la pantalla de Atención al público, se incluyó validación en el momento de liquidar el código de documento 23 o 240.

Para esto se realizaron los siguientes cambios.

* A nivel de clases

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /com/camaradirecta/ventanilla/actions/LiquidacionvtaAction.java | Se incluye validación del estado de la renovación de la matrícula cuando corresponda a un documento 23 o 240. |
| /com/camaradirecta/ventanilla/models/Model.java | Se incluyó el método validaRenovaCambioDomicilio, encargado de ejecutar el procedimiento que valida la renovación de la matrícula. |

* A nivel de base de datos

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| PKG\_CAMBIO\_DOMICILIO | Se crea para incluir procedimientos asociados al cambio de domicilio. |
| SP\_VALIDAR\_RENOVACION | Procedimiento creado para realizar la validación de la renovación del inscrito y mostrar el mensaje cuando no lo está. |

**Scripts Ejecutados**

|  |
| --- |
| * PKG\_CAMBIO\_DOMICILIO.sql |

1. **Inicio Proceso de Estudio**

Cuando el abogado este iniciando el proceso de estudio de la liquidación generada para los conceptos 23 o 240 se debe verificar si la empresa esta renovada a tiempo y mostrar un mensaje de advertencia.

* Se incluyó validación en la pantalla de Inscripción, en el momento en el que el usuario da clic en Inscribir y se visualiza la ventana emergente.

Para esto se realizaron los siguientes cambios.

* A nivel de clases

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /com/camaradirecta/documento/actions/ CargainscripciongeneraAction.java | Se incluye validación del estado de la renovación de la matrícula cuando corresponda a un documento 23 o 240. |

* A nivel de base de datos

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| PKG\_CAMBIO\_DOMICILIO | Se crea para incluir procedimientos asociados al cambio de domicilio. |
| SP\_VALIDAR\_RENOVACION | Procedimiento creado para realizar la validación de la renovación del inscrito y mostrar el mensaje cuando no lo está. |

**Scripts Ejecutados**

|  |
| --- |
| * PKG\_CAMBIO\_DOMICILIO.sql |

1. **Aprobación Inscripción**

Cuando el abogado apruebe la inscripción el sistema actualmente cambia el estado de la liquidación en la tabla Radicador\_documentos a estado 3 y cambia el estado del comerciante en la tabla inscritos a 10. Se requiere guardar un certificado en formato PDF del comerciante antes y después del cambio de estado en inscritos.

* Se incluyó funcionalidad para que cuando el abogado apruebe la inscripción correspondiente a los documentos 23 y 240, se genere un certificado en formado PDF justo antes de la inscripción y otro justo después de la inscripción, los cuales se almacenan en un directorio parametrizado en el archivo de propiedades Ruta.properties.

Para esto se realizaron los siguientes cambios.

* A nivel de clases

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /com/camaradirecta/documento/actions/ InscripcionlibropanelAction | Se incluye proceso para verificar si corresponde a un documento 23 o 240 y generar el certificado antes de registrar los correspondientes a la aprobación y el certificado después de aplicar los cambios. |
| /com/camaradirecta/documento/models/Model.java | Se modificó el procedimiento insertaDocumento, para asignar el modelo de documento correspondiente si se trata de un documento 23 o 240.  Se modificó el método guardaPanelInscripcionesLibros, para actualizar el estado del inscrito y asociar el documento aprobado. De tal manera que los cambios se vean reflejados en el certificado generado.  Se incluyó el método getModeloCodDoc, para obtener el modelo asignado a un código de documento. |
| /com/cooprac/seguridad/models/Ruta.properties | Se incluyó la entrada DirectorioCambioDomicilio, para parametrizar el directorio en el que se guardan los certificados |
| /com/cooprac/public/util/Uti.java | Se incluye la variable estática DirectorioCambioDomicilio.  Se incluye la lectura de la entrada DirectorioCambioDomicilio y su valor se asigna a la nueva variable estática. |

1. **Cambio de Domicilio**

Crear una opción en ventanilla web (menú consultas) que muestre las liquidaciones de los códigos 23, 240 (y trámites RUP) por rango de fechas, permitiendo elegir entre las que ya están radicadas a rúes o no lo estén. Esto se sabrá a partir de la nueva tabla, aquella que tiene los dos campos llenos si estará ya radicada, la que tenga el campo vacío liquidación\_rues.

* Se creó una nueva opción denominada Cambio de Domicilio ubicada en el menú Consultas.
* Se creó un procedimiento almacenado que permite consultar los registros a visualizar.
* Se incluyó redireccionamiento a Radiación Rues desde esta nueva opción cuando no se ha realizado este trámite.
* Se incluyó la generación del certificado (RUP y RM).
* Se incluyó la generación de la información en formato json (RUP y RM)

Se realizaron los siguientes cambios:

* A nivel de vista

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /WebRoot/WEB-INF/struts-config-inscrito.xml | Se incluye el mapeo y navegación para la nueva funcionalidad de Radicación de Cambio de Domicilio |
| /WebRoot/inscrito/radicacambiodomicilio.jsp | Página incluida para presentar la información de cambio de domicilio consultadas. |

* A nivel de clases

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /com/camaradirecta/rue/actions/CargaconsultarueAction.java | Se incluye captura de request Opcion, para poder determinar si se trata de un proceso proveniente de la nueva opción y asignar los datos de razón social y matrícula y se redirecciona a radicación rue. |
| /com/camaradirecta/inscrito/models/Model.java | Se incluyeron los métodos:  - **consultarRadCambioDoc:** método utilizado para consultar la información a visualizar en pantalla.  - **generarJson:** método utilizado para generar la información en formato json. |
| /com/camaradirecta/inscrito/forms/RadicaCambioDomicilioFormBean.java | Clase creada para manejar los atributos correspondientes al formulario de la nueva opción. |
| /com/camaradirecta/inscrito/dtos/RadicaCambioDomicilioDto.java | Clase creada para mapear los resultados de la consulta y mostrarlos en pantalla. |
| /com/camaradirecta/inscrito/actions/RadicaCambioDomicilioAction.java | Clase creada para realizar el proceso correspondiente a las acciones ejecutadas en la nueva funcionalidad. |
| /com/camaradirecta/inscrito/actions/CargaRadicaCambioDomicilioAction.java | Clase creada para el cargue inicial de la nueva opción. |
| /com/camaradirecta/ApplicationResources.properties | Se incluyeron las propiedades:  titulo.inscrito.radicacion.cambio.domicilio  mensaje.generacion.json.rues |

* A nivel de base de datos

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| PKG\_CAMBIO\_DOMICILIO.SP\_CONSULTAR\_LIQUIDACIONES | Procedimiento creado para ejecutar la consulta de los procesos de cambio de domicilio en un rango de tiempo y registrados en la tabla LIQUIDACIONES\_CAMBIO\_DOMICILIO |
| PKG\_CAMBIO\_DOMICILIO.SP\_GENERAR\_JSON | Procedimiento creado para generar la información del proponente o inscrito en formato json dado el tipo recibido (RUP o RM) y los valores para matrícula, registro y cámara. |
| PKG\_CAMBIO\_DOMICILIO.SP\_CARGAR\_DATOS\_JSON | Procedimiento creado para validar la información cargada en el archivo. |

**SCRIPTS EJECUTADOS**

|  |
| --- |
| * PKG\_CAMBIO\_DOMICILIO.sql * MenuConsulta.sql |

1. **Job S3**

Modificar la tarea programada que se encarga de leer la carpeta /home/rue/rueS3 la cual se encarga de leer los archivos que allí reposan, subirlos al Amazon S3 y actualizar el RUES usando el servicio web de actualización de trámites.

* Se incluyó verificación del servicio, cámara receptora y cámara origen para establecer el estado a enviar cuando corresponda a servicios de cambio de domicilio.

Se realizaron los siguientes cambios:

* A nivel de clases

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /com/camaradirecta/apiRues/ApiRues.java | Se modifica el método actualizarEstadoRadDocumentos para incluir lectura del servicio, cámara Receptora y Cámara Destino. Se valida los servicios que corresponden a Cambio de Domicilio y en caso de cumplir las condiciones se envía el estado 15. |

1. **Grabación Comerciante**

Validar que la liquidación sea de código documento 299 (liquidaciones\_Acum\_view)

Si es 299 solicitar que seleccionen el archivo de extensión .JSON que se ha recibido

A la matricula creada, realizar el update en la tabla inscritos y los inserts en las demás tablas necesarias con la información que contiene este archivo.

* Se modificó la modificación de la opción para incluir opción de cargue de archivo cuando cumple con las condiciones.

Se realizaron los siguientes cambios:

* A nivel de vista

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /WebRoot/theme/Master.css | Se incluye la clase css upFile |
| /WebRoot/inscrito/grabacioninsc.jsp | Se modifica para incluir funcionalidad de consultar la liquidación una vez se ingresa su número y saber si corresponde a un cambio de domicilio para solicitar el cargue del archivo. Se incluyó el método javascript consultarLiquidacion, encargado de enviar el formulario para realizar la consulta de la información de la liquidación ingresada.  Se incluye sección que permite realizar el cargue del archivo con extensión .json  Se incluye javascript que permite validar el archivo cargado, se insertaron los métodos:  - cargarDoc: utilizado para simular clic en el input tipo File y lanzar el popup de cargue del archivo.  - actualizarNombreArchivo: método que permite validar que el archivo seleccionado tenga extesión .json  - mostrarMensaje: método que permite mostrar los mensajes de error, se utiliza cuando la extensión del archivo no corresponde. |

* A nivel de clases

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /com/camaradirecta/inscrito/models/Model.java | Se incluyeron los métodos:  - **guardarJson:** método utilizado para leer el archivo a cargar y guardarlo en la base de datos para su posterior validación y distribución en tablas.  **- validarJson:** método utilizado para ejecutar el procedimiento almacenado que se encarga de verificar que la información recibida en el archivo json sea correcta validando campos obligatorios y tipos de datos.  **-consultarIdCargueJson:** método utilizado para obtener el siguiente número de la secuencia.  -**consultarErroresCargue:** método utilizado para consultar el log de errores generados en el cargue del archivo. |
| /com/camaradirecta/inscrito/forms/GrabacioninscFormBean.java | Se incluyeron los siguientes atributos con sus respectivos get y set requeridos para el cargue del archivo json:  **-FormFile archivoJson:** almacena la información del archivo seleccionado.  **-String nombreDoc:** guarda el nombre del archive seleccionado  **-String codConcepto:** guarda el concepto de la liquidación a generar en el proceso de incorporación rues.  **-Boolean cargarJson:** registra si se muestra o no la opción de cargue del archivo. |
| /com/camaradirecta/inscrito/actions/GrabacioninscAction.java | Se incluyó la acción consultarLiquidacion, que permite verificar si se muestra o no la opción de cargar el archivo json. |

* A nivel de base de datos

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| PKG\_CAMBIO\_DOMICILIO.SP\_CARGAR\_DATOS\_JSON | Procedimiento creado para validar la información cargada en el archivo. |

**SCRIPTS EJECUTADOS**

|  |
| --- |
| * PKG\_CAMBIO\_DOMICILIO.sql} * SEQ\_CARGUE\_JSON.sql |

1. **Reporte Cambio Domicilio Proponentes**

Crear un reporte que muestre la información de los proponentes ACTIVOS cuya matrícula mercantil este en estado CAMBIO DE DOMICILIO.

* Se incluyó el menú Cambio de Domicilio en el menú Consultas🡪Reporte Proponentes
* Se creó el reporte jasper que permite obtener la información de los proponentes.

**REPORTES GENERADOS**

|  |
| --- |
| * reporteCambioDomicilio.jrxml * reporteCambioDomicilio.jasper |

**SCRIPTS EJECUTADOS**

|  |
| --- |
| * PKG\_CAMBIO\_DOMICILIO.sql * MenuConsulta.sql |

1. **Actualización de noticia y estado a RUES**

Se debe crear una sonda que recorra la tabla rp\_inscripción 1 vez por hora, buscando registros cuyo tipo\_tramite = 18 y numero\_publicación NULO; si se encuentra una coincidencia se debe realizar la actualización de la noticia y estado a RUES, para ello se debe usar los servicios de actualización RUES RR30N y RR9N, los cuales actualmente se consumen desde el modulo RUP de la ventanilla web en sus opciones de actualizaciones rues.

* Se incluyó un cron de java que se ejecuta cada hora y verifica los trámites de tipo 18 sin número de publicación para realizar el consumo de los servicios RR30N y RR9N.

Se realizaron los siguientes cambios:

* A nivel de clases

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /com/camaradirecta/rup1510/Models/Model.java | Se incluyó el método  **getInscripcionesSinPublicar:** encargado de obtener los registros no publicados al rues. |
| /com/cooprac/publico/útil/Util.java | Se incluyeron las variables estáticas:  - textoCorreoCambioDomicilio  - correosCambioDomicilio  Utilizadas para obtener los valores asignados de Ruta.properties para las llaves textoCorreoCambioDomicilio y correosCambioDomicilio, respectivamente. |
| /com/cooprac/seguridad/models/Ruta.properties | Se incluyeron las llaves  **-textoCorreoCambioDomicilio:** utilizada para configurar el texto a enviar en el correo que realiza la publicación en el rues.  **-correosCambioDomicilio:** permite configurar los correso a los que se debe enviar el correo informando de la publicación en el rues. |
| /com/quartz/servlet/QuarzMB.java | Se modificó el método IniciarQuarz, para incluir el inicio del nuevo job.  Se modificó el método execute, para incluir la ejecución del nuevo job.  Se incluyeron los métodos:  **iniciarProcesoNoticiaCambioDomicilio:** utilizado para iniciar la ejecución del proceso.  **iniciarCambioDomicilio**: utilizado para configurar el esquema de ejecución |
| /com/quartz/servlet/InvocaQuartz.java | Se incluye el método:  **procesoNoticiaCambioDomicilio:** método encargado de ejecutar el proceso de actualización en el rues.  **getRueLog:** método encargado de registrar en el log. |

1. **Incorporación Transacción Rues**

Se debe modificar la opción de incorporar transacciones de rues del menú ventanilla, cuando el trámite corresponda al código 05070000 y la cámara\_destino = 05, el sistema debe generar una liquidación de concepto 27 (esto viene parametrizado desde el mismo sistema ya) y se genera un registro en la tabla rpcaktra el cual debe quedar con el campo tipo\_tramite en 19 y sw\_web en 3.

* Se incluyó la validación del servicio y la cámara destino para hacer la modificación del concepto de liquidación a registrar y el estado de sw\_web a asignar.

Se realizaron los siguientes cambios:

* A nivel de vista

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /WebRoot/rue/incorporacionrue.jsp | Se incluyó campo oculto codServicio para almacenar el código del servicio asociado al registro seleccionado |

* A nivel de clases

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /com/camaradirecta/rue/actions/IncorporacionrueAction.java | Se incluyó modificación para asociar al request el código de servicio. |
| /com/camaradirecta/rue/actions/IncorporacionrueFormBean.java | Se incluyó variable codServicio, para guardar el código del servicio asociado a la liquidación seleccionada. |
| /com/camaradirecta/ventanilla/actions/LiquidacionvtaAction.java | Se incluyó validación del código del servicio para que cuando corresponda a Cambio de domicilio asignar el concepto 27 |
| /com/camaradirecta/ventanilla/models/Model.java | Se modificó el método guardarLiquidacionRue, para que cuando corresponda a una liquidación de concepto 27, coloque SW\_Web en 3. |

1. **PreRegistro RUP**

Si la liquidación ingresada tiene el campo NRO\_REGISTRO vacío se debe buscar el número de registro que le corresponderá, para eso puede invocar el procedimiento almacenado denominado P\_REGISTRO\_CAMBIO el cual recibe como parámetro el número de liquidación. En caso de que liquidación si tenga el campo con valor, el proceso continuara normalmente.

* Se validó, para las liquidaciones correspondientes a cambio de domicilio, que cuando no cuentan con número de registro asignado, se asocia uno mediante la invocación del procedimiento P\_REGISTRO\_CAMBIO.
* Se incluyó funcionalidad para que se pueda cargar, validar y guardar la información del archivo json recibido en las tablas temporales de RUP.
* Una vez guardada la información de manera correcta, se actualizó el campo SW\_PAGO\_VIRTUAL.

Se realizaron los siguientes cambios:

* A nivel de vista

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /WebRoot/rup1510/preregistrorup.jsp | Se incluye sección que permite realizar el cargue del archivo con extensión .json  Se incluye javascript que permite validar el archivo cargado, se insertaron los métodos:  - cargarDoc: utilizado para simular clic en el input tipo File y lanzar el popup de cargue del archivo.  - actualizarNombreArchivo: método que permite validar que el archivo seleccionado tenga extesión .json  - mostrarMensaje: método que permite mostrar los mensajes de error, se utiliza cuando la extensión del archivo no corresponde. |

* A nivel de clases

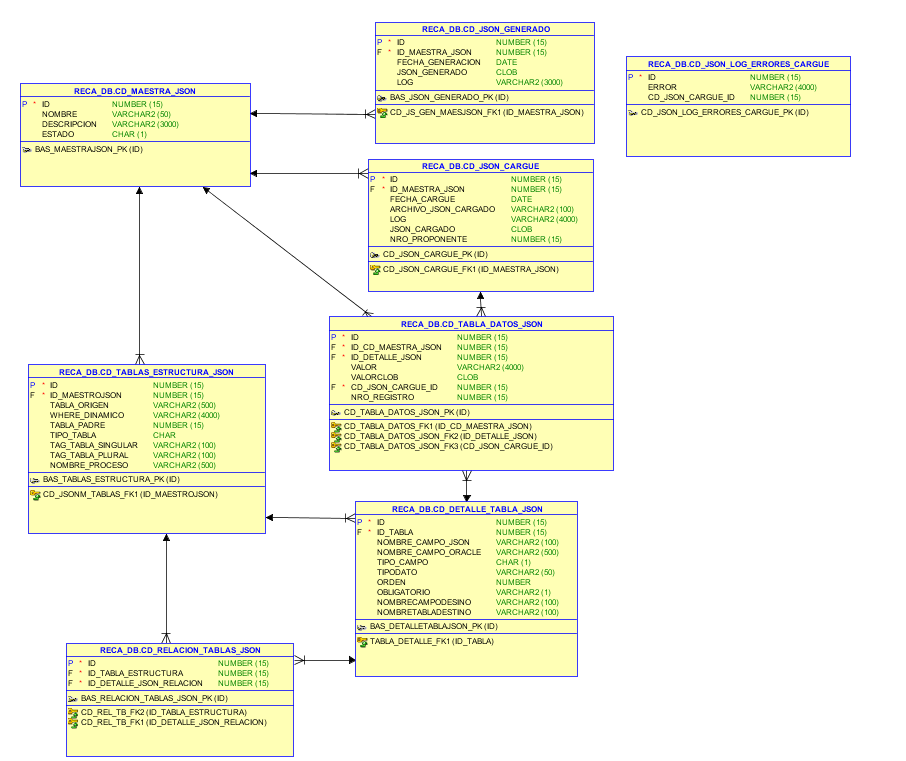
|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| /com/camaradirecta/rup1510/forms | Se incluyeron los atributos  - Boolean cargarJson: permite indicar si se muestra o no la funcionalidad de cargar el archivo json.  - FormFile archivoJson: guarda la información del archivo seleccionado |
| /com/camaradirecta/rup1510/actions | Se incluyó la acción validarJson  Se modificó la acción buscar, para validar que si no cuenta con el número de registro sea asignado mediante el consumo del procedimiento P\_REGISTRO\_CAMBIO |
| /com/camaradirecta/rup1510/models/Model.java | Se incluyó el método  registroCambioProponente: método que permite ejecutar el procedimiento P\_REGISTRO\_CAMBIO |
| /src/com/cooprac/publico/útil/Util.java | Se incluyó el método replaceAll, el cual permite reemplazar parte de un StringBuilder. |

* A nivel de base de datos

|  |  |
| --- | --- |
| **Nombre recurso** | **Modificación realizada** |
| PKG\_CAMBIO\_DOMICILIO.SP\_CARGAR\_DATOS\_JSON | Procedimiento creado para validar la información cargada en el archivo. |

1. **Modelo de datos**

Para la generación y cargue del archivo .json se incluyeron las siguientes tablas:

****

* CD\_MAESTRA\_JSON

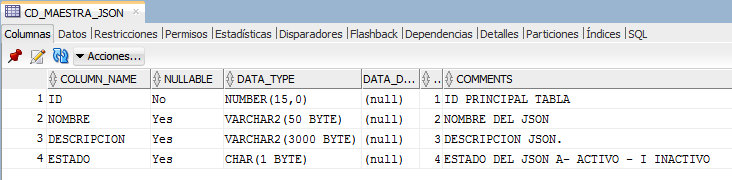


Tabla en la que se registran los tipos de json a generar, RUP y RM.

* CD\_TABLAS\_ESTRUCTURA\_JSON

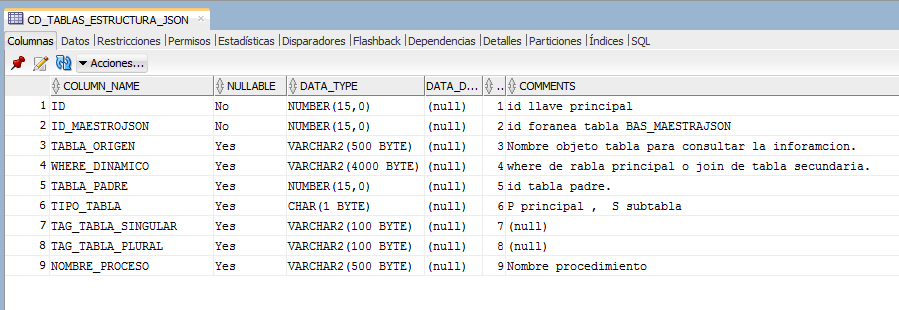


Tabla en la que se configuran los tags principales que conforman el archivo json

* CD\_DETALLE\_TABLA\_JSON

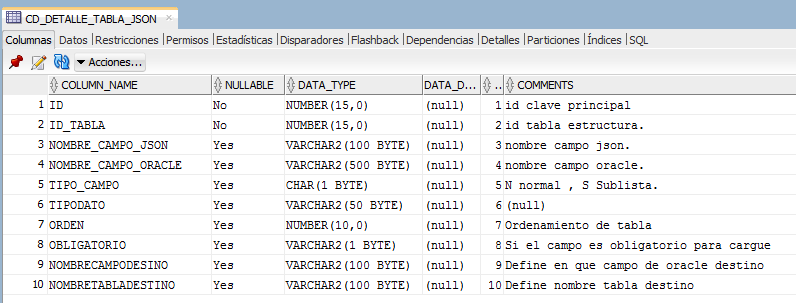


Tabla en la que relacionan los campos asociados a los tags principales que conforman el archivo json.

* CD\_RELACION\_TABLAS\_JSON

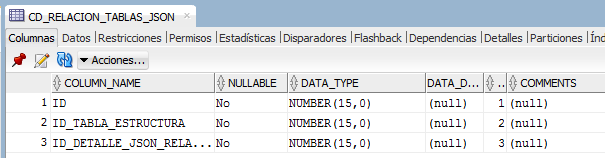


Tabla en la que se relacionan los tags principales con sus campos.

* CD\_TABLA\_DATOS\_JSON

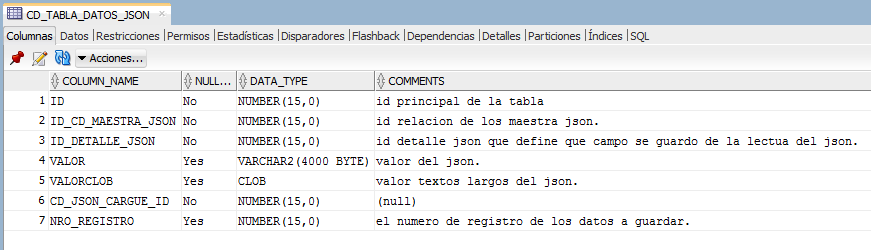


Tabla en la que se almacenan los campos leidos del archivo .json cargado para su posterior distribución en las correspondientes tablas (RUP o Mercantil).

* CD \_JSON\_GENERADO

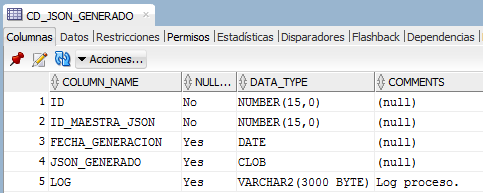


Tabla en la que se registra el log de errores causados en la generación del archivo y se guarda el archivo generado.

* CD \_JSON\_CARGUE

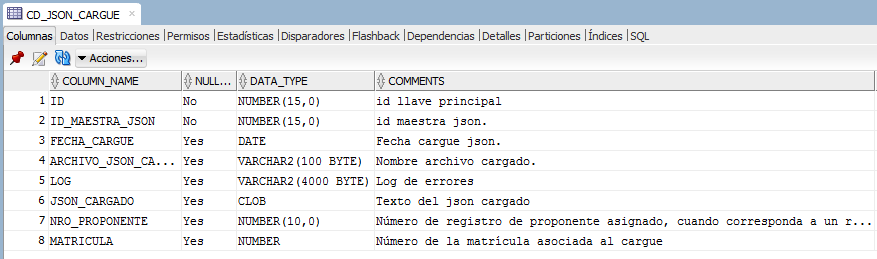


Tabla en la que se guarda el archivo seleccionado, para su posterior lectura y registro de campos en la tabla CD\_TABLA\_DATOS\_JSON

* CD \_JSON\_LOG\_ERRORES\_CARGUE

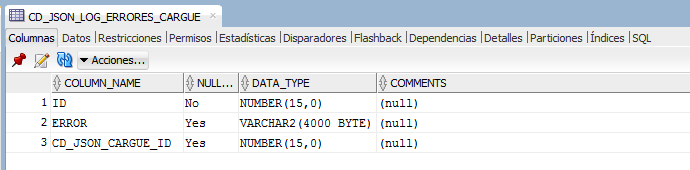


Tabla en la que se registran los errores producto de la validación del archivo cargado.

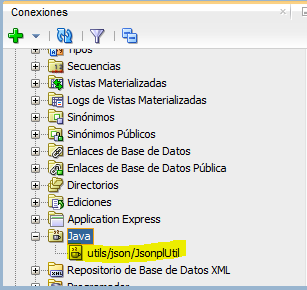
1. **Generación Json de Cambio de Camara y Cargue Json.**

Todo el proceso de la generación del archivo .json se realizó a nivel de base datos, para lo que se realizaron los siguientes cambios:

* Se realizó la instalación de la librería json-20140107.jar, mediante el comando

**call dbms\_java.loadjava('-force -resolve -grant PUBLIC -synonym D:\2020\camara2\CambioDomicilio\json-20140107.jar');**

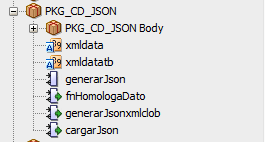
* Se incluyó la clase java JsonPlUtil, dentro de la base de datos,



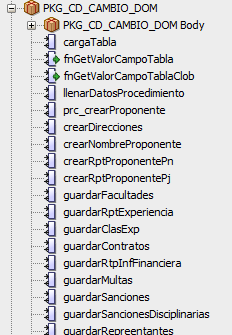
Esta clase se encarga de transformar de XML a JSON y leer un JSON y almacenarlo en la estructura temporal para la lectura de datos que vienen del json.

Existen 2 métodos públicos de java que son compilados para el uso de métodos PLSQL.

* **JsonplUtil.generarJsonxmlclob**
  + Este método se encarga de recibir una cadena CLOB de Oracle a transformarlo a un JSON. Este es el encargado final de generar de un SQL XML a ser transormado a JSON.
  + El metodo retorna un CLOB del String completo JSON generado para ser exportado como archivo de cambio de cámara.
* **JsonplUtil.Jsontodataoracle**
  + Este método se encarga de leer un JSON con base a la estructura definida en las tablas : cd\_tablas\_estructura\_json, cd\_detalle\_tabla\_json
  + Una ves leído el json con base a la estructura y base a los campos configurados el sistema almacena los datos referenciando a que campo columna de la definición de estructura en la tabla : CD\_TABLA\_DATOS\_JSON
* Se crearon los siguientes paquetes
  + PKG\_CD\_JSON



* + - El procedimiento **generarJson,** se encarga de generar la información en formato json. Recibiendo como parámetro el id de la estructura json a generar. Este mismo procedimiento se encarga de generar un SQL XML para posteriormente ser transformado a una estructura JSON.
    - La función **fnHomologaDato**, se utiliza para homologar los datos que lo requieren con las tablas Bas\_ del Rues
    - La función Oracle **gnerarJsonxmlclob**, se utiliza para una ves generado el XML CLOB se pasa al método java **JsonplUtil.generarJsonxmlclob** para que este transforme el texto XML a JSON.
* La función Oracle **cargarJso**n, se utiliza para llamar al método java **JsonplUtil.Jsontodataoracle.**



* + - El procedimiento **cargaTabla**, siendo el método el cual lee la información para trasformar los datos que lee de las tablas : cd\_tabla\_datos\_json a , cd\_detalle\_tabla\_json b, cd\_tablas\_estructura\_json .carga de los datos JSON.
* La función **fnGetValorCampoTablaClob****,** se encarga de leer un dato TIPO CLOB que se cargaron por campo de la tabla CD\_TABLA\_DATOS\_JSON.VALORCLOB y registro
  + Recibe los parámetros :
    - proceso number,
    - idTabla number,
    - campoTabla varchar2,
    - numReg number
* La función **fnGetValorCampoTabla,** de traer un dato tipo Varchar2 leyendo campo y registro de la tabla CD\_TABLA\_DATOS\_JSON
  + Recibe los parámetros:
    - proceso number,
    - idTabla number,
    - campoTabla varchar2,
    - numReg number
* El procedimiento **llenarDatosProcedimiento**, se encarga de ir por cada estructura json verificar que proceso pertenece y llamar el procedmiento de lógica de inserción de datos. Es decir en la tabla CD\_TABLAS\_ESTRUCTURA\_JSON en el campo NOMBRE\_PROCESO se almacena el nombre del proceso para que el mismo procedimiento cumpla con los programas y la lógica de insesrion de datos.
  + Ejemplo el proceso PRCPROPONENTE de una estructura cargada de json se verifica para llamar los subprocesos que almacenan el proponente.
* Se crearon procedimientos específicos para registrar la información en las tablas correspondientes de rpt y mercantil.
* Se crearon las siguientes funciones
* F\_ULTIMO\_AGNO\_INF\_FIN: función que retorna el último año reportado en información financiera del proponente.
* F\_NOMBRE\_SOCIO: función que retorna el nombre de un socio.
* F\_CIUU\_INSCRITO: función que retorna los códigos ciiu de un inscrito.
* F\_FEC\_INI\_CIIU: función que retorna la fecha de inicio del ciiu.
* TO\_BASE64: función que encripta en base 64 una cadena
* F\_FECHA\_CORTE\_INF\_FIN: función que retorna la fecha de corte de la información financiera.
* F\_TRAERINFO\_FIN\_TIPO: función que devuelve el valor financiero dado el tipo.
* F\_CLASIFICACIONES: función que devuelve las clasificaciones de una experiencia, separadas por coma.
* F\_CONSULTADIRECCION: función que retorna la dirección dado el tipo.
* F\_FACULTADES\_TEXTO: función que retorna el texto de las facultades del proponente.
* F\_ULTIMA\_INSCRIPCION: función que retorna el número de la última inscripción de un proponente.

* Se crearon las siguientes vistas
* V\_VINCULOS\_INSCRITOS: vista que contiene la información de todos los vínculos asociados a un inscrito: representantes, revisores, juntas, socios.
* V\_CAPITAL\_INSCRITOS: vista que contiene la información del capital de un inscrito
* V\_PROPIETARIOS\_ESTAB: vista que contiene la información de los propietarios de los establecimientos del inscrito
* V\_CASAS\_INSCRITO: vista que contiene la información de las casas(sedes o sucursales) de los inscritos
* V\_INSCRIP\_KARDEX: vista que contiene la información de las inscripciones de kardex de los inscritos
* V\_REPRESENTANTES\_LEGALES: vista que contiene la información de los representantes legales del proponente.
* VI\_CD\_RMINFOFINANCIERA: vista que contiene la información financiera de los últimos 5 años del inscrito.
* Se crearon las siguientes secuencias
* CD\_JSON\_GENERADO\_SEQ
* CD\_TABLA\_DATOS\_JSON\_SEQ
* CD\_JSON\_LOG\_ERRORES\_CARGUE\_SQ
* Se crearon los siguientes types
* RP\_DIRECCION\_TP

1. **Cargue Json**

Todo el proceso de la validación y cargue del archivo .json se realizó a nivel de base datos, para lo que se realizaron los siguientes cambios:

* En la clase java creada dentro de la base de datos, se incluyó el método XXXXX
* Se crearon los siguientes paquetes
* Se crearon las siguientes funciones
* FROM\_BASE64
* Se crearon las siguientes tablas
* RPT\_CONTRATO
* RPT\_MULTA
* RPT\_SANCION
* RPT\_SANCION\_DISCIPLINARIA
* Se crearon las siguientes secuencias
* SEQ\_CARGUE\_JSON