**RAAL: Aclaraciones sobre el código fuente**

Autores: Miguel Torres González, Nieves Carrera Bécares

Tabla de contenido

[Introducción 1](#_Toc345491392)

[Action Base: *RAALActionBase* 2](#_Toc345491393)

[Form Base: *RAALBaseForm* 3](#_Toc345491394)

[Sistema de “pestañas” en pantalla principal 3](#_Toc345491395)

[Tratamiento y funcionamiento de mensajes y errores 5](#_Toc345491396)

[Flujo de procesamiento de mensajes y errores 7](#_Toc345491397)

[Funcionalidades de los “actions” principales 9](#_Toc345491398)

[RAALEntidadTabsDispatcherAction 9](#_Toc345491399)

[RAALEntidadesAlimentariasAction 9](#_Toc345491400)

[RAALGenerarDocumentosAction 10](#_Toc345491401)

[RAALAutorizacionesEntidadAction 11](#_Toc345491402)

[RAALAutorizacionesRGSAAction 11](#_Toc345491403)

[RAALAutorizacionesLocalesAction 12](#_Toc345491404)

[“Forms” principales y “OT” más importantes relacionados 12](#_Toc345491405)

[Clases “OT” y “OTExt” 13](#_Toc345491406)

[RAALEntidadesAlimentariasForm 13](#_Toc345491407)

[RAALGenerarDocumentosForm 13](#_Toc345491408)

[RAALDatosReportOT 14](#_Toc345491409)

[RAALEntidadesAlimentariasOTExt 14](#_Toc345491410)

[RAALEntidadesAlimentariasOT 14](#_Toc345491411)

## Introducción

Es este documento se intenta aclarar en lo posible el código fuente java de la aplicación RAAL.

La motivación principal para la generación de este documento es ciertas peculiaridades existentes en el código fuente de RAAL, parte de las cuales se han intentado subsanar a la par que se elaboraba este documento, y que son las siguientes:

* Sistema de pestañas ideado para la pantalla principal de la aplicación. Mediante este documento se trata, sobre todo, de documentar su funcionamiento, de cara a una posible mejora (o cambio total) futura de este sistema. Dicho sistema de pestañas ha condicionado otras funcionalidades base de la aplicación, como es el tratamiento de errores.
* Desorden en el reparto de la funcionalidad entre los “actions”. Es este documento se documenta la funcionalidad de los “actions” principales y/o más problemáticos para facilitar el mantenimiento del código.
* Poca calidad general en buena parte del código, causada sobre todo por:
  + Inclusión en los “actions” de lógica de negocio en abundancia, casi toda ella, además, en interminables métodos “ejecutar”.
  + Nefasto uso del “copiar” y “pegar”: existencia de trozos de código con idéntica funcionalidad repetidos por toda la aplicación.

Este último punto es, con diferencia, el mayor problema del código fuente de la aplicación.

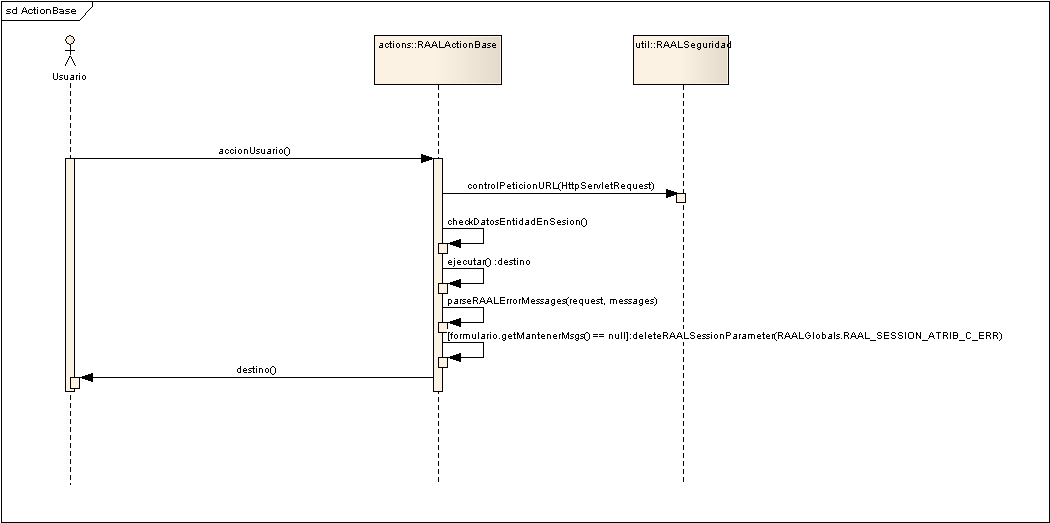
Se aprovechará la investigación realizada al elaborar este documento para (además de documentar) reorganizar los métodos “ejecutar” en tantos métodos como acciones diferentes y eliminar el código muerto que se vaya localizando.

## Action Base: *RAALActionBase*

Prácticamente todos los actions de la aplicación extienden del *RAALActionBase.* Por tanto analizar el contenido del mismo es vital para comprender el flujo entre clases de la aplicación.

Las características básicas de dicho *action* son:

* Extiende de *org.apache.struts.action.Action.* Por tanto con él tenemos toda la potencia de struts encapsulada.
* Define como objetos privados, los típicos de un *action* de java: *ActionErrors, ActionMessages, HttpSession.*
* Además implementa métodos propios, que facilita el acceso a funcionalidades comunes desde cualquier sitio de la aplicación: *fijaMensajesScreen()*, *getEntAlimenAutorizacionCompleta(), getRAALSessionParameter(), parseRAALErrorMessages(), setRAALSessionParameter().*

El diagrama de flujo queda definido de la siguiente manera:

Por tanto, para cualquier clase java que extienda de *RAALActionBase* y que implemente el método *ejecutar()*, tendrá la siguiente secuencia común de pasos:

1. RAALSeguridad.*controlPeticionURL*(httpRequest);

Método encargado de evitar peticiones ilegales: evitar peticiones hechas al servidor desde la barra de direcciones del explorador.

1. checkDatosEntidadEnSesion();

Los datos de la entidad alimentaria se cargan en sesión para evitar su recarga constante en cada llamada. Este aspecto se detalla en el apartado .

1. StrDestino = this.ejecutar(formulario, baseMessages);

Llamada el método ejecutar que se implementará en cada action específico.

1. parseRAALErrorMessages(httpRequest, baseMessages);

Método que se encarga de añadir convenientemente a *ActionErrors* o *ActionMessages* los mensajes que se pasan como parámetro. (Se detalla más adelante)

1. deleteRAALSessionParameter(RAALGlobals.*RAAL\_SESSION\_ATRIB\_C\_ERR*);

Por último, si el parámetro del formulario *getMantenerMsgs()* no devuelve unvalor, se borrará de sesión el valor del parámetro *RAALGlobals.RAAL\_SESSION\_ATRIB\_C\_ERR*.

## Form Base: *RAALBaseForm*

Esta clase es el “form” del que extienden el resto de “forms” en la aplicación. Por tanto posee unas propiedades comunes para todos ellos.

A través de dichas propiedades, se gestiona el flujo dentro de la aplicación para que se ejecuten unas acciones u otras, se eliminen parámetros de sesión o no, se almacenan variables globales accesibles desde cualquier punto de la aplicación, etc…

De entre estas propiedades, las más comunes y útiles de cara al funcionamiento general, destacan:

* accion: contiene un literal con la acción a realizar. Será la propiedad que filtre el flujo dentro del método *ejecutar()* de cada action específico.
* origen: acción previa que se realiza.
* cMsgError: literal que contiene la lista de errores o mensajes devueltos por el action principal.
* mantenerMsgs y mostrarMsgs: para determinar si se mantienen en sesión y se muestran los mensajes.

## 

## Sistema de “pestañas” en pantalla principal

Los datos de una entidad alimentaria (en la práctica término equivalente a “autorización sanitaria”) se muestran en la aplicación repartidos en forma de “pestañas”. Tanto el funcionamiento como la implementación de dicho sistema son complejos y no siguen la forma de trabajo habitual con “struts”.

El **funcionamiento** de dichas pestañas, en modo de modificación de datos, es el siguiente:

* Cada vez que se cambia de pestaña:
  1. Se guardan en base de datos los datos de la entidad alimentaria correspondientes a la pestaña actual.
  2. Se cargan de base de datos casi todos los datos de la entidad alimentaria para poder realizar ciertas validaciones sobre los mismos (en teoría, se cargan sólo los relacionados con la pestaña actual). Se refresca la vista de la pestaña actual.
  3. Se cargan de base de datos los datos a mostrar en la pestaña de destino (o puede que más).
  4. Se cargan de base de datos casi todos los datos de la entidad alimentaria para poder realizar ciertas validaciones sobre los mismos y poder mostrar los mensajes de error correspondientes (en teoría, se cargan sólo los relacionados con la pestaña destino). Se muestra la vista de la pestaña destino.
* Cuando se abre una ventana de tipo “popup”, también se realiza un proceso similar al anterior (se guardan los datos, etc…)
* El funcionamiento en modo de sólo lectura es similar, excluyendo la parte de guardado de datos.

Claramente, el comportamiento anterior es muy poco eficiente. Se ha intentado mejorarlo guardando los datos de la entidad en sesión para recargarlos sólo cuando sea estrictamente necesario. Para ello, existen los siguientes métodos en RAALActionBase:

checkDatosEntidadEnSesion

getEntAlimenAutorizacionCompleta

El primero (checkDatosEntidadEnSesion) se llama siempre en el execute y comprueba, en función de la “acción” anterior, si los datos de la entidad alimentaria almacenados en sesión continúan siendo válidos o no. Si no son válidos, los borra.

El segundo (getEntAlimenAutorizacionCompleta) es llamado desde los “actions” que heredan de RAALActionBase cuando necesitan recargar los datos de la entidad alimentaria. El método devuelve los datos de la entidad alimentaria almacenados en sesión si están disponibles (es decir, si han sido cargados y almacenados en sesión con anterioridad y checkDatosEntidadEnSesion no los ha eliminado) o los carga desde base de datos y almacena en sesión en caso contrario.

**Nota**: el sistema de “cacheado en sesión” de la entidad alimentaria se puede considerar sólo como un parche para hacer que la aplicación funcione de forma algo más ágil, pero lo que habría que hacer es rehacer la aplicación de tal forma que se mantuviesen todos los datos de la entidad en el "form" correspondiente sin necesidad de guardar nada en sesión ni recargar los datos una y otra vez (lo cual seguramente implique rehacer media aplicación).

Los principales “actions” derivados de RAALActionBase e implicados en el proceso anterior son:

* **RAALAutorizacionesEntidadAction**: Se encarga de guardar en BD los datos al pulsar en los botones “guardar”, al cambiar de pestaña o al lanzarse un “popup” (paso 1 de los descritos arriba).
* **RAALEntidadTabsDispatcherAction**: Se encarga de cargar de BD el contenido a mostrar en cada pestaña, validar los datos y generar los mensajes de error consecuencia de la validación (pasos 2, 3 y 4). Para algún caso determinado, también se encarga de guardar datos (paso 1).

El flujo genérico seguido al cambiar de pestaña se muestra en un diagrama a continuación, aunque hay que tener en cuenta que la funcionalidad de guardado de datos realizada por RAALEntidadTabsDispatcherAction, puede ser realizada por otros “actions” diferentes (incluído RAALEntidadTabsDispatcherAction).

## 

## Tratamiento y funcionamiento de mensajes y errores

Según el estándar de struts, dentro de la propiedad *ActionErrors* irán aquellos mensajes que se muestran al usuario como consecuencia de una ejecución incorrecta de la aplicación, ya sea por fallos propios del sistema, por errores en el código o por validaciones que los datos introducidos no cumplen. Dentro de la propiedad *ActionMessages* irán aquellos literales que informen al usuario del resultado correcto de la acción.

Dentro de la aplicación, al tener encapsuladas dichas propiedades dentro del Action base, no se puede acceder directamente a su contenido. Por ello se implementan métodos para la gestión de los mismos.

**RAALActionBase.parseRAALErrorMessages()**

A través del método *RAALActionBase.parseRAALErrorMessages(),* se añaden a las variables globales *errors* y *messages* los distintos mensajes que el usuario pasa, y que se obtienen del resultado de la ejecución de una acción.

Recibe dos parámetros:

* HttpServletRequest request: necesario para obtener el mensaje de error de las excepciones que se hayan producido como resultado de una ejecución incorrecta.
* ActionMessages messages: que contiene los distintos mensajes de éxito o error que se deben tratar para mostrar al usuario. Dichos literales deben estar definidos en el fichero ApplicationResources.properties

El funcionamiento básico del método es el siguiente:

1. Se itera la lista de mensajes que vienen en el parámetro.
   1. si contienen el literal “*error.”*, se añadirán a *errors*,
   2. si contienen el literal “*mensaje.”*, se añadirán a *messages*.
2. Una vez que se han tratado los mensajes, se obtiene el parámetro de la request “*exception”* y se añade como *errors.*

**RAALActionBase.fijarMensajesScreen()**

Método que parsea los errores y mensajes que se pasan como parámetros a una variable *ActionMessages,* evitando además la aparición de elementos repetidos.

Recibe como parámetros:

* String cMsgErr: cadena con la lista de mensajes que se añadirán, para mostrar al usuario. Si no se indica este parámetro, o se pasa una cadena vacía, utiliza como cadena de mensajes entrada los definidos en la variable de sesión RAALGlobals.RAAL\_SESSION\_ATRIC\_C\_ERR y, además, borra dicha variable de sesión.
* ActionMessages messages: objeto destino de los mensajes parseados.

La implementación de este método tiene sentido para aquellos casos en los que se pasan varios errores o mensajes complejos, es decir, más de un mensaje con datos de entidades específicos. Para poder parsear el literal que se pasa como parámetro a una lista de *ActionMessages,* se utiliza como carácter separador de mensajes el carácter almohadilla #.

El uso de este método no es necesario para los casos en que se desea incluir un sólo mensaje a *ActionMessage.*

**RAALActionBase.setRAALSessionParameter()**

A través de este método, se añade a la sesión la clave y valor pasados como parámetros.

Recibe dos parámetros:

* String key: literal con el nombre del parámetro de sesión.
* Object parameter: objeto con el valor que se almacenará asociado al literal.

Su funcionamiento es muy sencillo: añade la tupla variable-valor a la sesión.

**RAALActionBase.getRAALSessionParameter()**

Método que devuelve el objeto asociado a la clave que se pasa como parámetro.

Recibe un parámetro:

* String key: literal con el nombre del parámetro de sesión.

Devuelve el objeto asociado a la clave.

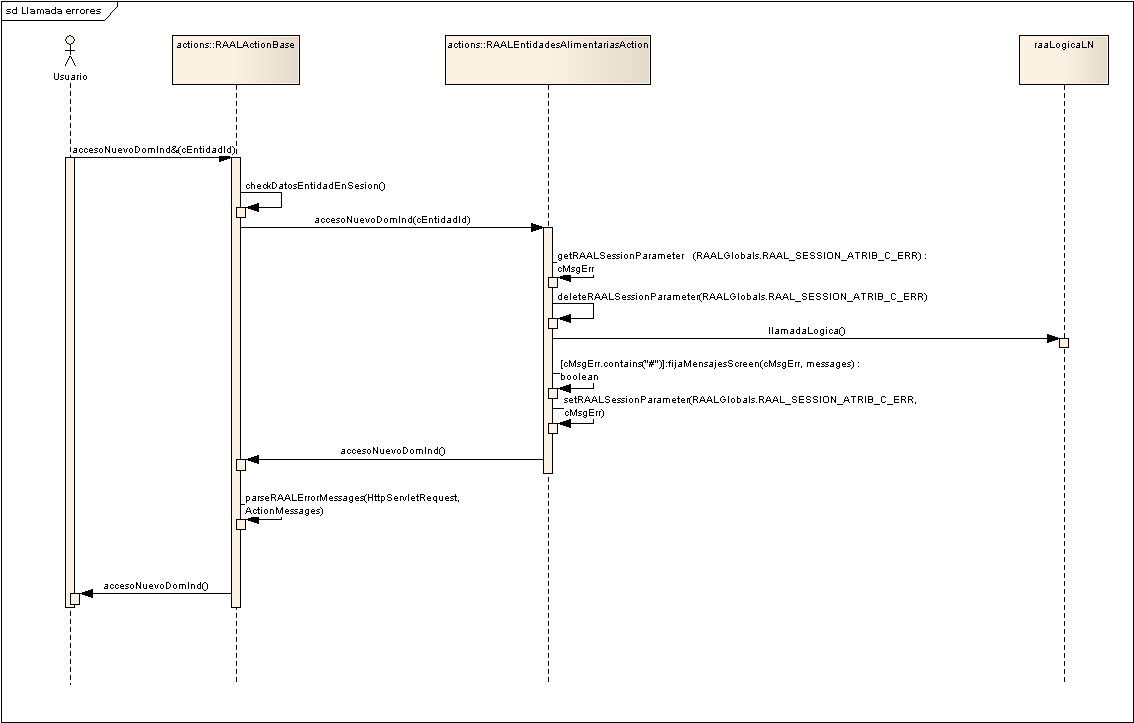
**RAALActionBase.deleteRAALSessionParameter()**

Método que se encarga de eliminar la clave que se pasa como parámetro del objeto *session.*

Recibe un parámetro:

* String key: literal con el nombre del parámetro de sesión.

### Flujo de procesamiento de mensajes y errores

Una vez definida la funcionalidad básica de los métodos relacionados con el tratamiento de mensajes, pasaremos a definir la forma en que la aplicación hace uso de los mismos.

La secuencia de llamadas que se tendría que dar sería la siguiente:

Un detalle importante del diagrama anterior a tener en cuenta es que la variable almacenada en sesión que almacena los mensajes (RAALGlobals.RAAL\_SESSION\_ATRIC\_C\_ERR), se borra hacia al principio del “ejecutar” (no sin antes almacenarla en una variable local) y se vuelve a definir al final. Entre medias, sin embargo, suele haber cierto procesamiento para mantener ciertos (o todos) mensajes que estaban en dicha variable y que no interesa (por la razón que sea) perder. Además, realmente esto no se hace así en todos los actions, sino que, según convenga, se juega con dicha variable almacenada en sesión, utilizándola o no si se quieren conservar los mensajes de error o no.

Como se comentó en la explicación del RAALActionBase, éste incluye unas directivas comunes para todos los actions que extienden de él, y que sirven para el tratamiento de los errores. En particular, en la llamada al método *parseRAALErrorMessages()* es donde se lleva a cabo el tratamiento de los mensajes que se pasan como parámetro. Por tanto, si en un action de la aplicación, hemos dado valor a la variable messages, este método será el encargado de tratarlos convenientemente.

Al estar incluido en el action genérico, dicho tratamiento se realizará en cada llamada de los actions que extiendan de él.

No obstante, existen varias formas de aplicación del tratamiento de los mensajes y errores.

#### Caso A: añadiendo mensajes simples directamente al objeto ActionMessages

Es la forma básica y basada en struts de añadir los mensajes a través del objeto *ActionMessages*.

messages.add("mensaje.domIndustrial.grabar.ok",

new ActionMessage("mensaje.domIndustrial.grabar.ok"));

messages.add("error.aplicacion.dispatch.no.valido",

**new** ActionMessage("error.aplicacion.dispatch.no.valido"));

Se puede especificar si se desea que los mensajes permanezcan en la sesión o si se muestran al usuario, con las propiedades del form setMantenerMsgs() y setMostrarMsgs();

Cuando en el action genérico se llega a la invocación del método *parseRAALErrorMessages(),* dado que los mensajes están correctamente indexados en la lista, el método los añadirá sin realizar ningún tratamiento sobre ellos.

#### Caso B: añadiendo mensajes simples a la variable de sesión cMsgErr

A través de esta variable de sesión se pueden mostrar mensajes simples pero con datos específicos de una entidad alimentaria.

setRAALSessionParameter(RAALGlobals.*RAAL\_SESSION\_ATRIB\_C\_ERR*, "mensaje.validacionEnt.ok("+ formulario.getEntidadSeleccionada().getAutorizacionEntidad().getCCodigo() + ")");

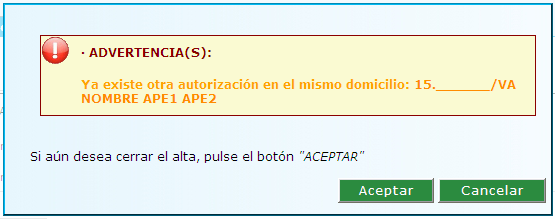
Al definir un mensaje en la variable global *cMsgErr*, el valor de esta se mantiene en la sesión, hasta que ésta termine o hasta que se elimine de forma manual, invocando al método deleteRAALSessionParameter().

En este caso, cuando se llega a la invocación del método *parseRAALErrorMessages(),* éste se encargará de transformar el literal en un ActionMessage entendible para struts, separando la parte que contiene los datos de la entidad alimentaria y que va entre paréntesis del mensaje propiamente dicho.

#### Caso C: añadiendo varios mensajes a través de la variable de sesión cMsgErr

Para poder mostrar varios mensajes al mismo tiempo y además añadir datos específicos de entidades alimentarias, se ha implementado el método *RAALActionBase.fijarMensajesScreen().* Como se ha comentado previamente, éste se encarga de transformar el literal que contiene la cadena de mensajes a mostrar separados por el carácter #, en una lista de *ActionMessages().*

Por ejemplo, para mostrar el siguiente mensaje informativo:



*cMsgErr* debería contener el siguiente contenido:

“error.altaFinalAdvertencia#advertencia.validacionEnt.dIndustrial.existente(15.\_\_\_\_\_\_/VA NOMBRE APE1 APE2)#”

Dentro del diagrama de secuencia, habrá que comprobar si *cMsgErr* contiene el carácter #, para invocar el método *fijarMensajesScreen()* para que pueda separarlos y convertirlos en messages entendibles para struts. Posteriormente serán tratados por el método *parseRAALErrorMessages()*.

## Funcionalidades de los “actions” principales

A continuación se detalla el reparto de funcionalidades entre algunos de los “actions” que conforman RAAL (se tienen en cuenta los más importantes y/o con funcionalidad menos evidente).

### RAALEntidadTabsDispatcherAction

Atiende las acciones relacionadas con las cargas de datos contenidos en las "pestañas" en las cuales se distribuyen los datos de una entidad alimentaria, y también otras relacionadas con el almacenamiento de algunos de esos datos (domicilio industrial y entidad matriz).

RAAL\_ACCION\_AUTORIZACIONES. Carga datos de pestaña: “Industria/ Establecimiento/ Actividad”

RAAL\_ACCION\_TITULAR\_ACTIVO. Carga datos de pestaña: “Titular”

RAAL\_ACCION\_ENTIDAD\_CON\_INSTALACIONES. Carga datos de pestaña: “Entidad con instalaciones”

RAAL\_ACCION\_ENTIDAD\_SIN\_INSTALACIONES. Carga datos de pestaña: “Entidad sin instalaciones”

RAAL\_ACCION\_GUARDAR\_DOMICILIO\_INDUSTRIAL. Guarda datos de pestaña “Entidad con instalaciones”, tanto al dar al botón “guardar” de dicha pestaña como al salir de la misma pinchando en otra.

RAAL\_ACCION\_SUCURSALES. Guarda datos de la entidad matriz de la entidad actual. Ocurre al seleccionar una entidad tras pulsar en “Definir como sucursal de otra I/E/A” en la pestaña “Entidad con instalaciones”.

RAAL\_ACCION\_REFRESH\_LIST\_DIR. Recarga ciertas listas de objetos presentados en las pestañas de la aplicación "Entidad con instalaciones" (sucursales, almacenes, datos de direcciones) y "Titular" (tipos de personas jurídicas, comuneros y datos de domicilios/direcciones). Al parecer, la acción ocurre al cambiar un elemento seleccionado en alguna lista (por ejemplo, lista de municipios) que implique la recarga de otras listas (en el ejemplo, implicaría las listas de localidades y vías). Se desconoce por qué recarga otras listas que, en principio, no deberían estar afectadas (ejemplo, al cambiar el municipio se recarga la "lista" con la entidad matriz, o la lista de comuneros…).

* Al finalizar cada una de las acciones anteriores, se validan ciertos datos de la entidad alimentaria (según la pestaña actual) y se generan los mensajes de validación correspondientes.

### RAALEntidadesAlimentariasAction

Atiende un popurrí de acciones relacionadas con la pantalla de búsqueda de entidades alimentarias, la eliminación física de los datos de una entidad, el inicio del cese definitivo de una entidad, la finalización del alta y las modificaciones de titular, domicilio industrial, domicilio social y tipo de entidad alimentaria.

RAAL\_ACCION\_CONSULTA. Acceso a pantalla de búsqueda de entidades y acciones de recarga diversas en dicha pantalla.

RAAL\_ACCION\_BUSCAR. Acción de buscar en pantalla de búsqueda de entidades.

RAAL\_ACCION\_FINALIZAR\_ALTA. Finalización del alta de una entidad en proceso de alta.

RAAL\_ACCION\_ELIMINAR\_AUTORIZACION. Borrado físico de datos de una entidad en proceso de alta.

RAAL\_ACCION\_ACCESO\_NUEVO\_TIT. Inicio de cambio de titular y acciones de recarga relacionadas con la pantalla correspondiente.

RAAL\_ACCION\_NUEVO\_TIT . Fin de cambio de titular.

"baja". Inicio de cese definitivo de entidad alimentaria: redirige a pantalla de búsqueda de entidad a dar de baja.

"accesoNuevoDomInd". Inicio de cambio de domicilio industrial y acciones de recarga relacionadas con la pantalla correspondiente.

"nuevoDomInd". Fin de cambio de domicilio industrial.

"accesoNuevoDomSocial". Inicio de cambio de domicilio social y acciones de recarga relacionadas con la pantalla correspondiente.

"nuevoDomSocial". Fin de cambio de domicilio social.

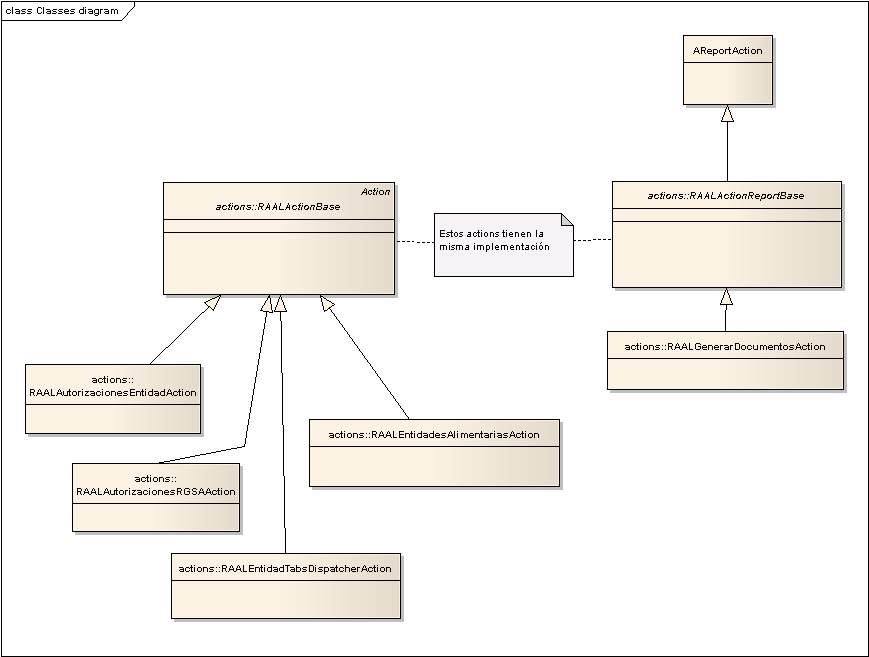
"accesoCambioTipo". Inicio de cambio de tipo de entidad alimentaria y acciones de recarga relacionadas con la pantalla correspondiente.

"cambioTipo". Fin de cambio de tipo de entidad alimentaria.

### RAALGenerarDocumentosAction

Se encarga de todas las acciones relacionadas con la generación de los documentos oficiales, gestionando el “popup” que hay al respecto y la creación en sí de los documentos utilizando “jasper reports”.

Este action tiene una peculiaridad importante: no extiende de RAALActionBase, sino que lo hace de RAALActionReportBase. Éste es un action genérico creado para la generación de informes con Jasper Reports. Su contenido es idéntico a RAALActionBase, sólo que a su vez este extiende de AReportAction. La ventaja principal reside en esto último: tenemos las funcionalidades para la generación de informes, y mantenemos las del action base genérico.



Incluye gran cantidad de lógica de negocio relacionada con la generación de documentos y más, aunque al menos está organizada de forma bastante modular en forma de métodos privados de la propia clase del “action”.

“cargaGenerarDocumentos". Mostrar “popup” generación de documentos y acciones de recarga relacionadas con la pantalla correspondiente.

"generarDocumento". Generación del documento mediante “jasper reports”.

"accesoBuscarFirmante". Mostrar ventana búsqueda firmantes.

"buscarFirmante". Búsqueda de firmante.

"accesoAltaFirmante". Mostrar ventana alta nuevo firmante.

"altaFirmante". Alta de nuevo firmante.

### RAALAutorizacionesEntidadAction

Atiende acciones relacionadas con el guardado de datos de las pestañas, inicio y fin de proceso de alta de entidad alimentaria, devolver entidad alimentaria a proceso de alta y baja (cese definitivo) de entidad alimentaria.

RAAL\_ACCION\_ALTA. Inicio alta entidad alimentaria.

RAAL\_ACCION\_MODIFICACION. Almacenar datos en pestañas de entidad alimentaria: ocurre al dar al botón guardar de cada pestaña, al cambiar de pestaña o al mostrar un popup desde alguna pestaña.

"accesoBajaIAE". Mostrar pantalla baja de entidad alimentaria (cese definitivo).

"bajaIAE". Acción de baja (cese definitivo) de entidad alimentaria.

"modVolverAlta". Acción devolver a proceso de alta entidad alimentaria.

### RAALAutorizacionesRGSAAction

Atiende las acciones relacionadas con la asociación de actividades a las entidades alimentarias con número de identificación de carácter nacional.

RAAL\_ACCION\_ALTA : acción para el acceso y alta de actividades asociadas a una entidad. Para distinguir entre ambas, según el valor de la variable *dispatch:*

DISPATCH\_TRUE: acción de alta de nuevas actividades.

DISPATCH\_FALSE: acceso a la pantalla de selección de actividades para dar de alta.

RAAL\_ACCION\_MODIFICACION: acción para el acceso y modificación de actividades asociadas a una entidad. Para distinguir entre el acceso y la modificación, según el valor de la variable *dispatch*:

DISPATCH\_TRUE: acción de modificación de los datos de actividades.

DISPATCH\_FALSE: acceso a la pantalla donde se solicitan los datos de la actividad seleccionada que se van a modificar.

RAAL\_ACCION\_BAJA. Acción para el borrado físico de los datos de actividades asociadas a la entidad alimentaria.

RAAL\_ACCION\_ ACCESO\_CESE\_ACT. Acción para el acceso a la función de cese de actividades asociadas a una entidad.

RAAL\_ ACCION\_CESE\_ACT. Acción de cese de los datos de actividades asociados a la entidad.

“accesoSup”. Acción para el acceso a la función de suspensión de actividades asociadas a una entidad alimentaria.

“suspension”. Acción que lleva a cabo la función de suspensión de actividades asociadas a una entidad, con los datos introducidos por pantalla.

“accesoFinSusp”: Acción para el acceso a la función de fin de suspensión de actividades asociadas a una entidad alimentaria.

“finSuspension”: Acción donde se lleva a cabo la función de fin de suspensión de actividades asociadas a la entidad alimentaria seleccionada, con los datos introducidos por pantalla.

Dado que algunas de estas acciones tienen la funcionalidad común de obtener, a partir de un literal, una lista con los OT’s de actividades asociados a la entidad, se ha incluido la función *obtenerListaActividades().*

### RAALAutorizacionesLocalesAction

Atiende las acciones relacionadas con la asociación de actividades a las entidades alimentarias con número de identificación de carácter local.

RAAL\_ACCION\_ALTA : acción para el acceso y alta de actividades asociadas a una entidad. Para distinguir entre ambas, según el valor de la variable *dispatch:*

DISPATCH\_TRUE: acción de alta de nuevas actividades.

DISPATCH\_FALSE: acceso a la pantalla de selección de actividades para dar de alta.

RAAL\_ACCION\_MODIFICACION: acción para el acceso y modificación de actividades asociadas a una entidad. Para distinguir entre el acceso y la modificación, según el valor de la variable *dispatch*:

DISPATCH\_TRUE: acción de modificación de los datos de actividades.

DISPATCH\_FALSE: acceso a la pantalla donde se solicitan los datos de la actividad seleccionada que se van a modificar.

RAAL\_ACCION\_BAJA. Acción para el borrado físico de los datos de actividades asociadas a la entidad alimentaria.

RAAL\_ACCION\_ ACCESO\_CESE\_ACT. Acción para el acceso a la función de cese de actividades asociadas a una entidad.

RAAL\_ ACCION\_CESE\_ACT. Acción de cese de los datos de actividades asociados a la entidad.

“accesoSup”. Acción para el acceso a la función de suspensión de actividades asociadas a una entidad alimentaria.

“suspension”. Acción que lleva a cabo la función de suspensión de actividades asociadas a una entidad, con los datos introducidos por pantalla.

“accesoFinSusp”: Acción para el acceso a la función de fin de suspensión de actividades asociadas a una entidad alimentaria.

“finSuspension”: Acción donde se lleva a cabo la función de fin de suspensión de actividades asociadas a la entidad alimentaria seleccionada, con los datos introducidos por pantalla.

## “Forms” principales y “OT” más importantes relacionados

La clase de “transferencia de datos” RaalEntidadesAlimentariasOTExt contiene la mayoría de los datos relacionados con una entidad alimentaria. Los distintos “forms” de la aplicación que requieren de todos, o buena parte, de los datos de una entidad alimentaria contienen un objeto de dicha clase.

En la descripción que se hace de cada clase, se hace algún comentario de cómo se podría mejorar el diseño actual.



### Clases “OT” y “OTExt”

En general, se sigue la terminología habitual del “framework” de la JCyL para denominar a los objetos de transferencia de datos que se corresponden con directamente con una tabla del modelo de datos (clases cuyo nombre termina en “OT”), mientras que las clases terminadas en “OTExt” extienden las clases “OT” con datos adicionales según las relaciones presentes en el modelo de datos.

### RAALEntidadesAlimentariasForm

Utilizado en las acciones de la ventana principal de la aplicación (pestañas) y en las acciones de modificación de datos de una entidad (cambio domicilio industrial, titular, etc.). Contiene un objeto de tipo RaalEntidadesAlimentariasOTExt, aunque ciertos datos de la entidad que podrían pertenecer a dicho “OT”, están en el “form”. El propio “form” se utiliza también como objeto de intercambio de datos para generar el informe “ficha de autorización sanitaria”.

### RAALGenerarDocumentosForm

Utilizado en las acciones para generar documentos. Contiene un objeto de tipo RaalDatosReportOT con los datos necesarios para generar los documentos correspondientes.

### RAALDatosReportOT

Objeto de intercambio de datos, con la información necesaria para generar los “documentos” de RAAL. Contiene un objeto del tipo RaalEntidadesAlimentariasOTExt. Ciertos datos propios de una entidad alimentaria contenidos en RAALDatosReportOT probablemente podrían sería más correcto que estuvieran en RaalEntidadesAlimentariasOTExt

### RAALEntidadesAlimentariasOTExt

Objeto de intercambio de datos que contiene la mayoría de los datos de una entidad alimentaria. Debería revisarse para que incluyese otros datos que actualmente no contiene y que, cuando son necesarios en la aplicación, se incluyen en los “forms” correspondientes, de tal forma algunos están inútilmente replicados entre los distintos “forms” e, incluso, entre el propio “form” y el RaalEntidadesAlimentariasOTExt que contiene.

### RAALEntidadesAlimentariasOT

Contiene los datos básicos (los de la tabla RAAL\_ENTIDADES\_ALIMENTARIAS) de una entidad alimentaria.