



CRITERIOS Y LINEAMIENTOS PARA EL DISEÑO DE INTERFACES



ÍNDICE

- [INTRODUCCIÓN](#)
- 1. [CÓDIGO DE COLORES](#)
- 2. [DISEÑO DE REQUERIMIENTOS](#)
- 3. [INTERFACES DE USUARIO](#)
- 4. [INTERFACES DE SISTEMA](#)
- 5. [INTERFACES DE COMPONENTES - DIAGRAMA DE COMPONENTES](#)
- 6. [INTERFACES DE COMPONENTES – DIAGRAMA DE CLASES](#)
- 7. [DISEÑO DE BASE DE DATOS](#)
- 8. [CUMPLIMIENTO DE REQUERIMIENTOS](#)
- 9. [GUÍA DE DISEÑO](#)

INTRODUCCIÓN

El presente documento tiene como objetivo, describir los lineamientos y criterios que deberán ser aplicados en el diseño de los requerimientos y las diferentes interfaces que se definirán en la construcción de la solución, incluyendo el diseño detallado y de base de datos.

Para tal efecto, se han considerado tres tipos de interfaces:

- | | |
|-------------------------------------|--|
| 1. Interfaces de Usuario: | Tienen como objetivo permitir la interacción del usuario con el sistema. |
| 2. Interfaces de Sistema: | Tienen como objetivo comunicar la solución que se construirá con otras soluciones y/o sistemas construidos previamente o a futuro. |
| 3. Interfaces de Componentes | Definen la comunicación y dependencia entre los diferentes componentes que se construirán como parte de la solución. |

Al cumplir con estos criterios de diseños de interfaces se asegurará de estar construyendo un producto bien diseñado y le evitará el levantamiento de observaciones y no conformidades en las verificaciones que involucran el diseño de la solución técnica del proyecto.

1. CÓDIGO DE COLORES

La finalidad del código de colores es ayudar y facilitar al equipo de desarrollo, la identificación de los distintos tipos de componentes definidos en las secciones **04 Componentes**, **05 BaseDeDatos** y **06 DiagramaClases**.

Por tal motivo, deberán aplicarse en todos los componentes que se definan en dichas secciones. A continuación se describe la tabla de los códigos de colores.



Universidad Tecnológica de León
Modelo de Madurez de la Capacidad Integrado - CMMI
CRITERIOS Y LINEAMIENTOS PARA EL DISEÑO DE INTERFACES



Los componentes que serán de nueva creación y, que requieren codificarse completamente, se indicarán con un fondo verde.



Los componentes que harán reuso de código fuente sin modificar, se indicarán con un fondo gris. En el caso del diagrama de base de datos, el color gris indica una tabla previamente existente.



Los componentes que harán reuso de código fuente pero, requieren alguna modificación, se indicarán con un color amarillo. En el caso del diagrama de base de datos, el color amarillo indica que se requiere modificar una tabla que previamente existe.



Los componentes que representan sistemas, se representarán en color rosa.



Los componentes que representan librerías compiladas y, que se utilizarán en la construcción de la solución, se representarán en color azul. Por ejemplo, librerías .jar y .dll.

2. DISEÑO DE REQUERIMIENTOS

Al definir nuevos requerimientos, asegúrese agregar la información descrita en los siguientes lineamientos, para cada requerimiento.

1. Cada requerimiento debe contener un nombre con una nomenclatura específica según el tipo de requerimiento:

- a) Requerimiento Funcional:

RFXX: Nombre del Requerimiento Funcional

Donde:

XX Representa el número consecutivo del requerimiento funcional.

- b) Requerimiento Funcional Derivado:

RDFXX_YY: Nombre del Requerimiento Funcional Derivado

Donde:

XX Representa el número consecutivo del requerimiento funcional.

YY Representa el número consecutivo del requerimiento funcional derivado.

- c) Requerimiento No Funcional:

RNFXX: Nombre del Requerimiento No Funcional

Donde:

XX Representa el número consecutivo del requerimiento no funcional.

2. Se debe agregar una descripción del requerimiento. La información para el punto anterior y éste, puede obtenerse del documento de requerimientos Requerimientos_x.x.xlsx, donde x.x representa la versión del documento.
3. Establezca un status para cada requerimiento: Cancelado, Entregado, Rechazado, Registrado, Terminado, Validado o Verificado, según corresponda.
4. Establezca el Tipo de Requerimiento: Funcional, Derivado o No Funcional, según corresponda.
5. Establezca la Prioridad del requerimiento: Alta, Media o Baja, según corresponda.
6. Establezca la Fase en la cual se definió dicho requerimiento, dentro del documento InformacionTecnicaDesarrollo.eap.

La Figura 1.1 muestra la definición correcta de un requerimiento.

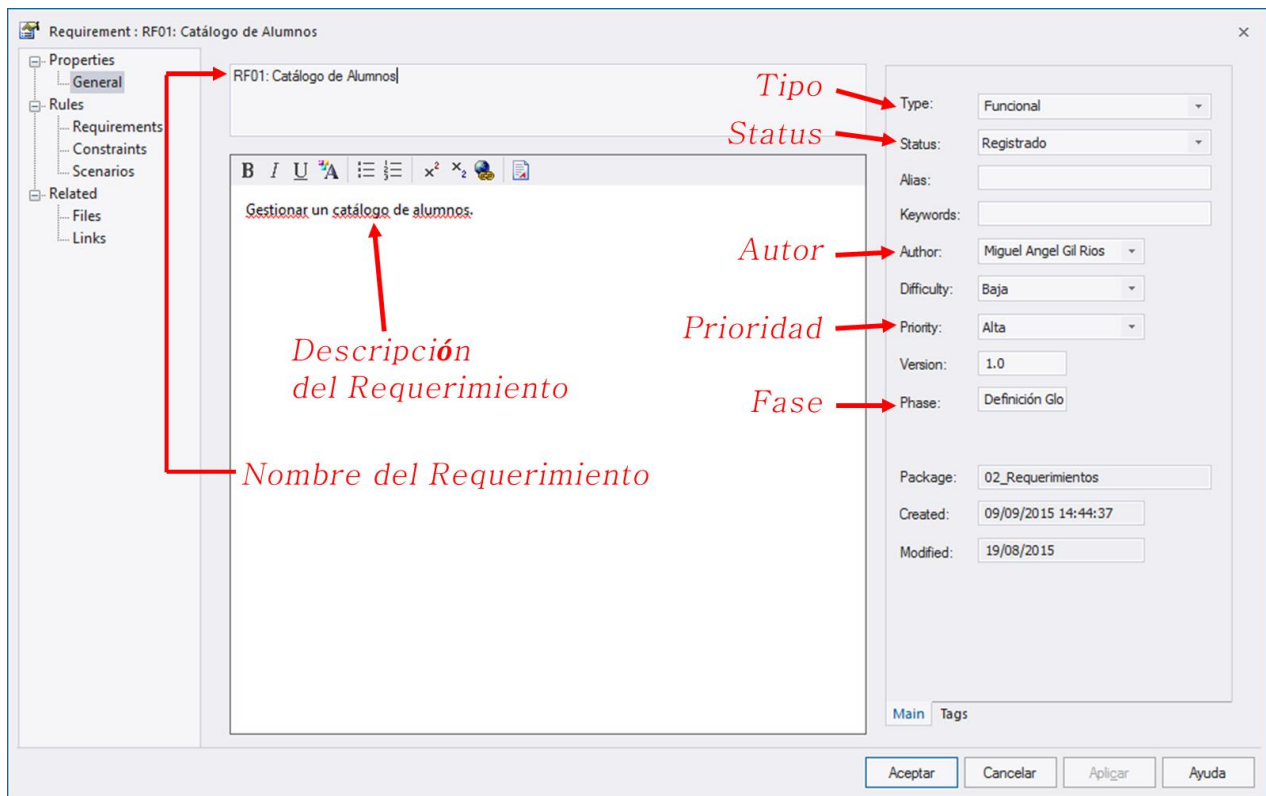


Figura 1.1. Ejemplo de los datos que debe contener un requerimiento.

7. Delimite los requerimientos de acuerdo a la iteración donde serán construidos. Esta delimitación se realiza mediante el uso de paquetes que son definidos dentro de la sección 02_Requerimientos y se

nombran como *Iteración1*, *Iteración2*, ..., *Iteración N*. La Figura 1.2 muestra un ejemplo de diseño de requerimientos con su delimitación por iteración.

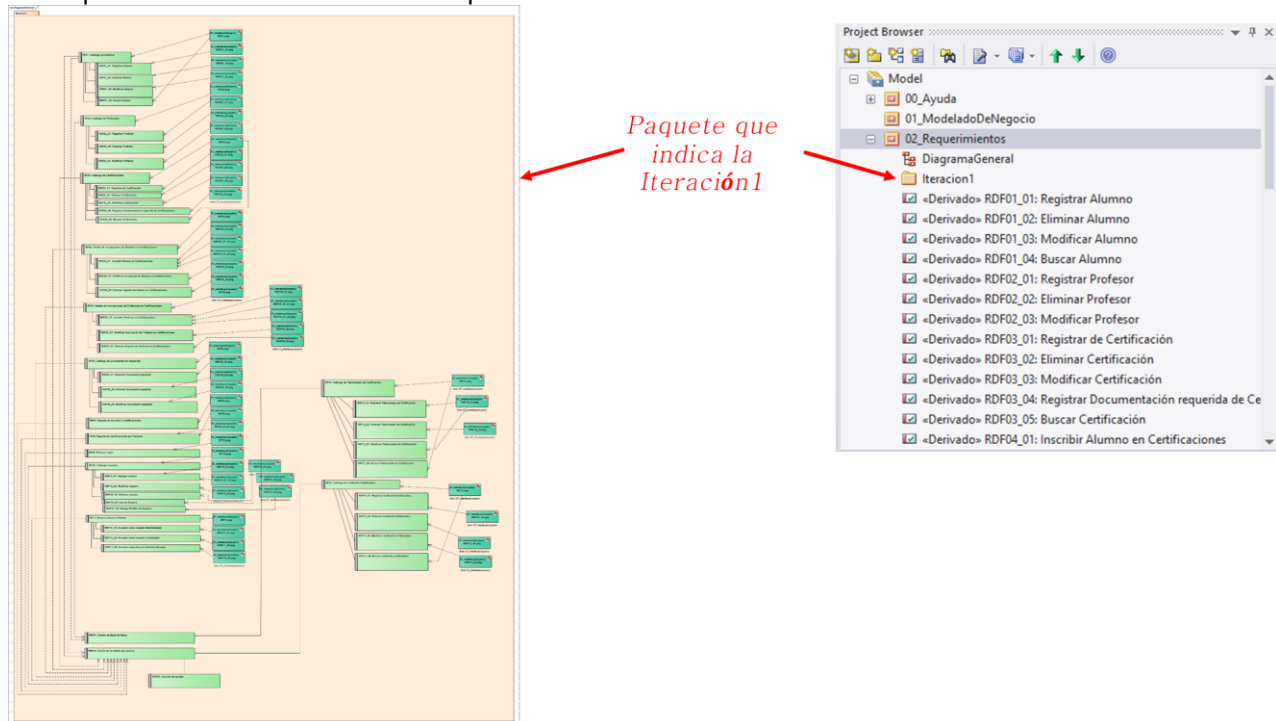


Figura 1.2. Delimitación de requerimientos por iteración.

3. INTERFACES DE USUARIO

A continuación se describen los lineamientos para el diseño de Interfaces de Usuario.

1. El diseño de las interfaces de usuario puede ser realizado en cualquier software que le facilite esta tarea, pudiendo utilizarse incluso, documentos html con datos de ejemplo precargados.
2. Cada interface de usuario diseñada, deberá exportarse como imagen para que posteriormente sea agregada al documento **InformacionTecnicaDesarrollo.eap**, en la sección **07_InterfacesUsuario**, dentro del diagrama **DiagramaGeneral**, como un *artefacto* **Artifact**. Para obtener ayuda de cómo agregar una interface de usuario, como un artefacto al documento antes mencionado, consulte la sección [9.2. Agregar Interfaces de Usuario como Artefactos](#).
3. Para asegurarse que lo especificado en el paso 2 es correcto, revise que el tipo del artefacto corresponde a *Local File*. Esto lo puede revisar en el apartado *Files*, que se encuentra dentro del nodo *Related*, en las propiedades de cada interface de usuario.

La Figura 3.1 muestra un ejemplo de donde visualizar la propiedad antes descrita.



Universidad Tecnológica de León
Modelo de Madurez de la Capacidad Integrado - CMMI
CRITERIOS Y LINEAMIENTOS PARA EL DISEÑO DE INTERFACES

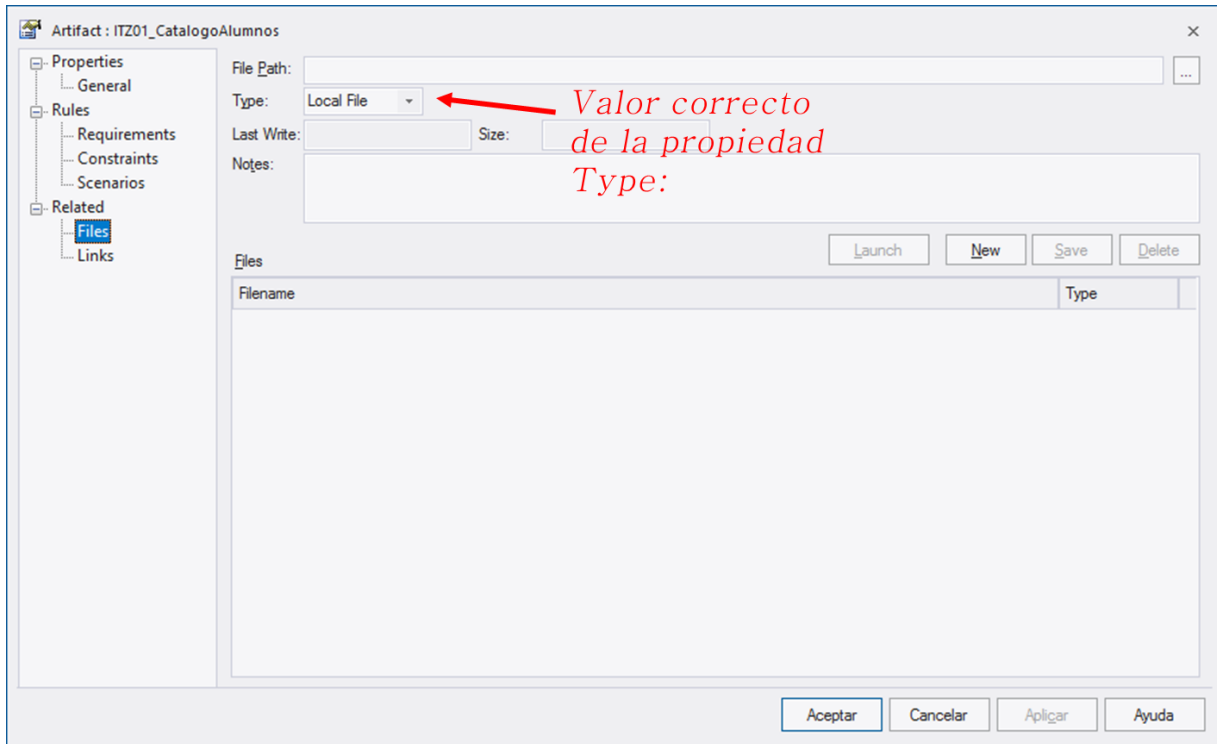


Figura 3.1. Visualización de la propiedad Type, al agregar una nueva interface de usuario como un artefacto interno.

4. Al agregar una nueva interface de usuario, asegúrese de establecer la siguiente información :
 - a) El nombre del autor de la misma (campo Author).
 - b) El lenguaje de programación y/o plataforma en la que será construida la interface (campo Language).
 - c) La fase en la cual se agregó la interface al documento (campo Phase).

La Figura 3.2, muestra un ejemplo del correcto llenado de la información antes descrita.

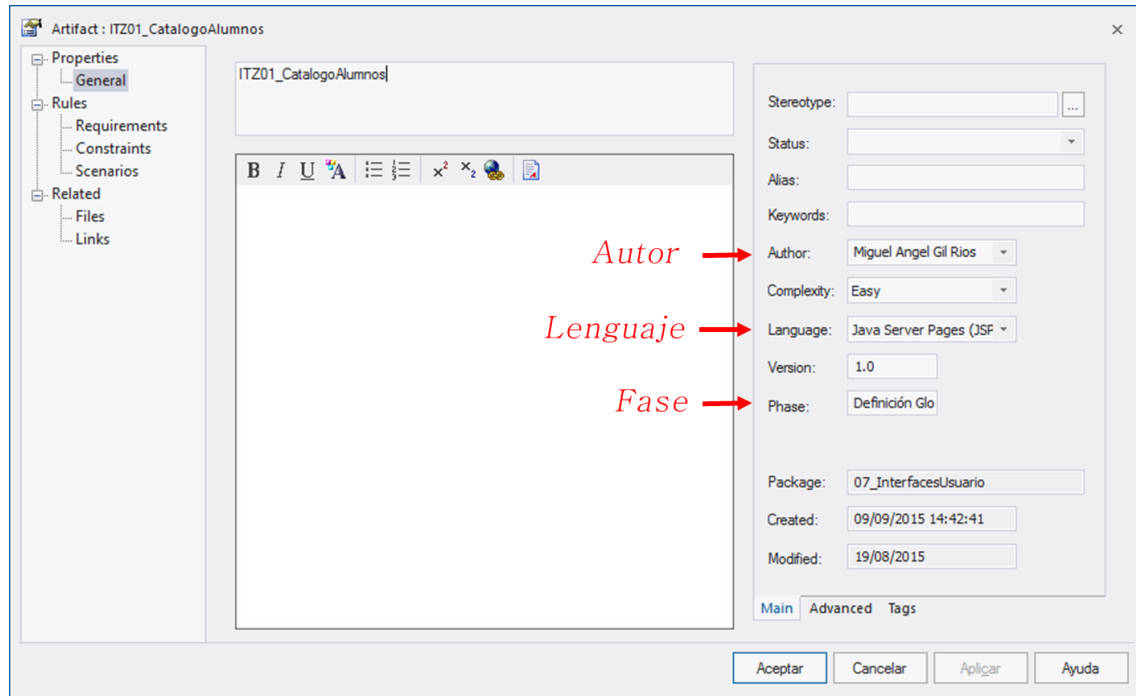


Figura 3.2. Llenado correcto de los atributos de una interface de usuario al agregarla.

5. Las interfaces de usuario deberán estar asociadas con los requerimientos, como una dependencia. Esto deberá realizarse en **DiagramaGeneral**, dentro de la sección **02_Requerimientos**.
6. La asociación de las interfaces de usuario, deberá indicar que la interface de usuario depende del requerimiento. Para conocer cómo realizar asociaciones de interfaces de usuario con requerimientos, consulte la sección [9.3. Asociación de Interfaces de Usuario con Requerimientos](#).

A continuación, en la Figura 3.3, se muestra un ejemplo de interfaces de usuario asociadas con sus requerimientos.

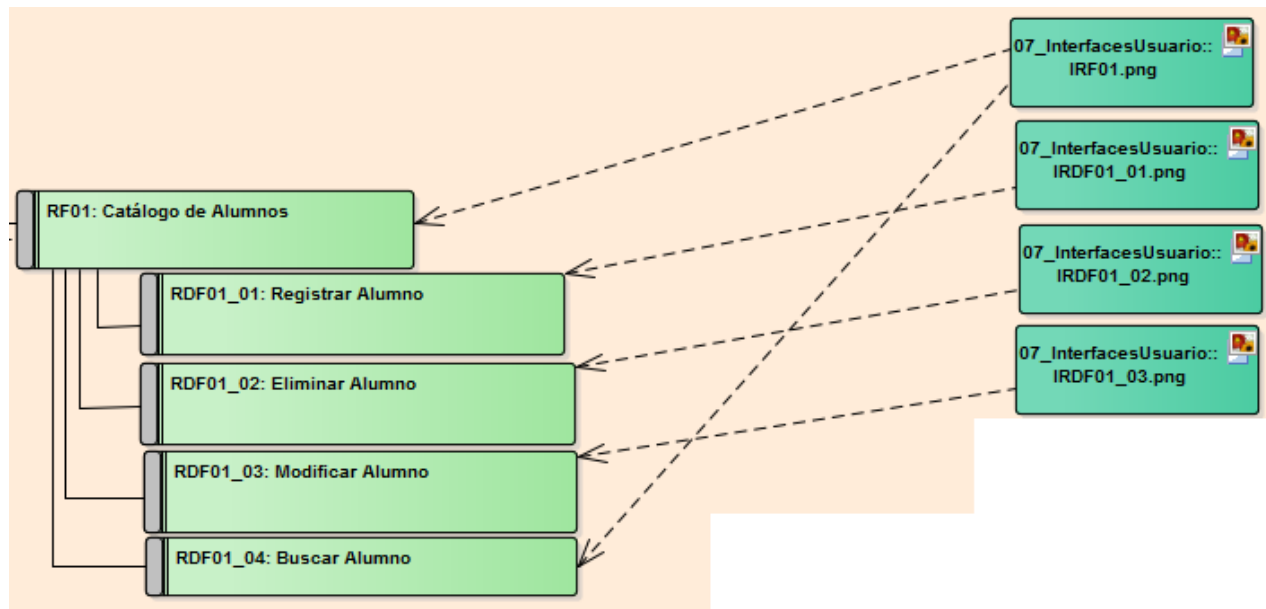


Figura 3.3. Ejemplo de interfaces de usuario asociadas con requerimientos.

7. Finalmente, al definir un requerimiento asegúrese de describir:
 - a) Una descripción del funcionamiento general de la clase.
 - b) El nombre del autor de la misma.
 - c) El lenguaje de programación.

La Fase en la que se definió dicho elemento: Inicio, Definición Global, Planeación Global, Análisis y Diseño por Iteración, Construcción por Iteración, Pruebas por Iteración, Implementación por Iteración, Cierre por Iteración o Cierre Global.

4. INTERFACES DE SISTEMA

A continuación se describen los lineamientos para el diseño de Interfaces de Sistema.

1. El diseño de las interfaces de sistema, se plasma en el documento **InformacionTecnicaDesarrollo.eap**, dentro de la sección **04_Componentes**. A su vez, también puede ser plasmada dentro de la sección **06_DiagramaClases**.
2. Cuando se desea representar un sistema, dentro de los diagramas de componentes y clases, éste se representa como un componente y, se le aplica el *color rosa* de fondo (de acuerdo al código de colores), para indicar que representa un sistema.
3. Las interfaces de sistema se diseñan como relaciones entre un componente y un componente de sistema.
4. A su vez, la relación entre un componente y un componente de sistema, representa una interface de sistema y ésta puede ser realizada como una **asociación** o una **dependencia**. Para conocer como



Universidad Tecnológica de León
Modelo de Madurez de la Capacidad Integrado - CMMI
CRITERIOS Y LINEAMIENTOS PARA EL DISEÑO DE INTERFACES

agregar una interface de usuario como una asociación de dependencia, consulta la sección [9.4. Asociación Dirigida de Dependencia entre Elementos](#).

5. Toda relación (comunicación) entre interfaces de sistema debe tener un nombre descriptivo.
6. Toda relación (comunicación) entre interfaces de sistema deberá estar documentada. Esta documentación deberá agregarse en las propiedades del elemento de relación que comunica a dos interfaces de sistema. La documentación implica la descripción de los mensajes que serán enviados entre dichas interfaces, siguiendo la estructura definida a continuación:

Envía:

[NombreDelComponenteA]: [Parámetro1], [Parámetro2], ..., [ParámetroN]

Responde:

[NombreDelComponenteB]: [Mensaje1] ó [Mensaje2], ó ..., ó [MensajeN]

Nota: Cuando aplique, es posible reemplazar la definición individual de cada parámetro por una descripción del tipo de comunicación que se realiza entre ambas interfaces.

En la estructura anterior, deberá reemplazar los elementos *[NombreComponenteA]* y *[NombreComponenteB]*, con los nombres de los componentes. Así mismo, deberá reemplazar los nombres de los parámetros y mensajes por sus nombres correspondientes.

La Figura 4.1, muestra un ejemplo con la documentación correcta de la comunicación entre interfaces de sistema:

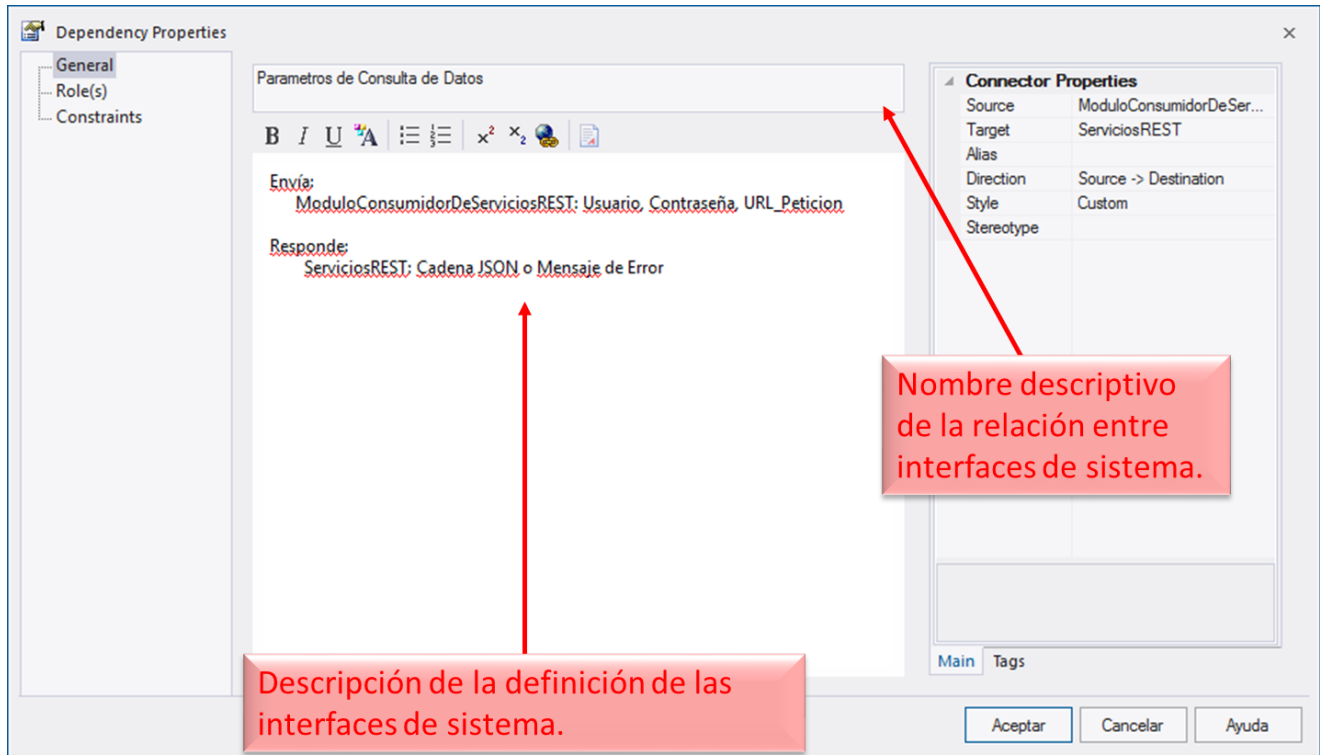


Figura 4.1. Documentación de la comunicación entre dos interfaces de sistema.

Conforme a descripción de la ventana anterior, el módulo ModuloConsumidorDeServiciosREST, envía parámetros (Usuario, Contraseña, URL_Peticion) al módulo ServiciosREST y, éste último, responde con una cadena JSON o un Mensaje de Error. Para ver el diseño de la comunicación definida en la Figura 4.1, vea la Figura 4.2.

7. Adicionalmente, puede agregar comentarios describiendo consideraciones y explicaciones que no pueden ser descritas dentro de los mensajes de comunicación.

A continuación se muestra un ejemplo correcto de cómo realizar el diseño de interfaces de sistema en un diagrama de componentes.

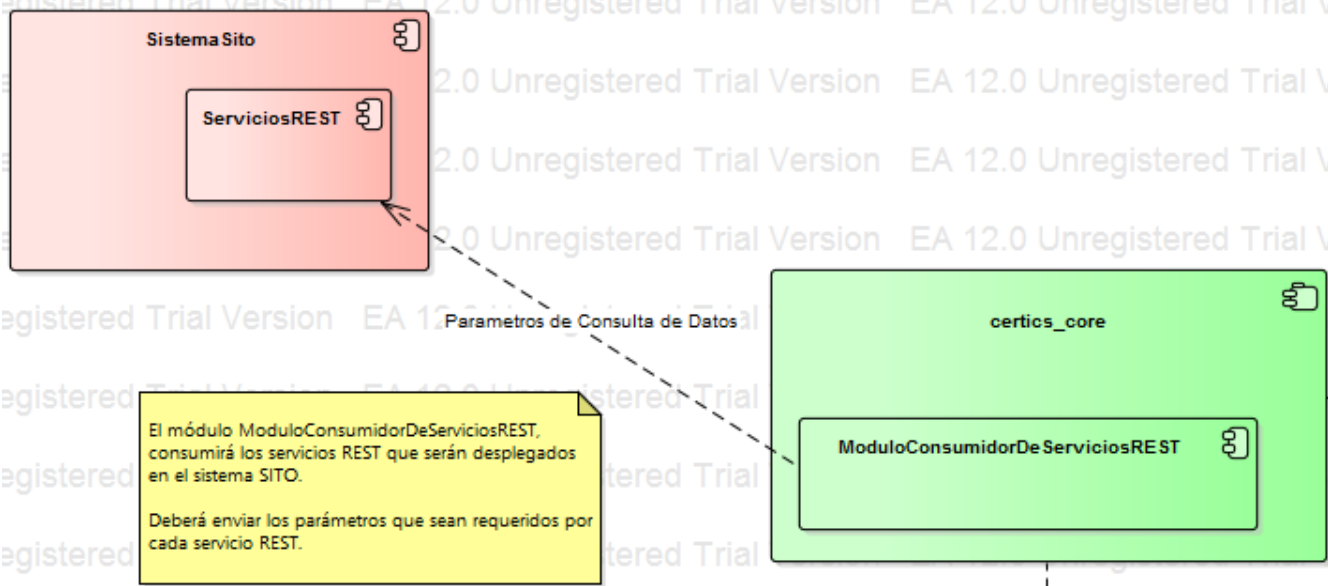


Figura 4.2. Ejemplo de diseño correcto de interfaces de sistema.

5. INTERFACES DE COMPONENTES – DIAGRAMA DE COMPONENTES

A continuación se describen los lineamientos para el diseño de Interfaces de Componentes que son definidos en la sección **04_Componentes**, dentro del documento **InformacionTecnicaDesarrollo.eap**.

1. El diseño de las interfaces de componentes, a nivel de un diseño por componentes, se plasma en el documento **InformacionTecnicaDesarrollo.eap**, dentro de la sección **04_Componentes**.
2. Cuando se desea representar un componente, dentro de los diagramas de componentes y clases, éste se representa como un componente y, se le aplica un color de fondo de acuerdo al código de colores especificado al inicio de este documento.
3. El nivel máximo de detalle de un diagrama de componentes será a nivel de componente **Component** o nivel de paquete **Package**, siempre y cuando exista un diseño detallado. En caso contrario, el diagrama de componentes deberá ser muy detallado y se podrán utilizar todos los siguientes elementos de diseño. Por ejemplo: **Component, Package, Artifact, Document Artifact, Object y Port**.
4. Cada elemento agregado, dentro de un diagrama de componentes, deberá contener en sus propiedades: Nombre, Descripción, Autor, Lenguaje y Fase (correspondiente con las fases del ciclo de vida).
5. La comunicación entre componentes debe ser realizada como una **dependencia**. Para conocer como comunicar dos componentes como una dependencia, consulte la sección [9.4. Asociación Dirigida de Dependencia entre Elementos](#).



Universidad Tecnológica de León
Modelo de Madurez de la Capacidad Integrado - CMMI
CRITERIOS Y LINEAMIENTOS PARA EL DISEÑO DE INTERFACES

6. Toda relación (comunicación) entre componentes, representa una interface de componentes. Por lo tanto, debe contener un nombre descriptivo.
7. A su vez, un componente (padre) que contiene a otros componentes (hijos), se define colocando los componentes hijos dentro del componente padre.
8. Toda relación (comunicación) entre interfaces de sistema deberá estar documentada. Esta documentación deberá agregarse en las propiedades del elemento de relación que comunica a dos interfaces de sistema. La documentación implica la descripción de los mensajes que serán enviados entre dichas interfaces, siguiendo la estructura definida a continuación:

Envía:

[NombreDelComponenteA]: [Parámetro1], [Parámetro2], ..., [ParámetroN]

Responde:

[NombreDelComponenteB]: [Mensaje1] ó [Mensaje2], ó ..., ó [MensajeN]

Nota: Es posible reemplazar la definición individual de cada parámetro por una descripción del tipo de comunicación que se realiza entre ambas interfaces.

9. En la estructura anterior, deberá reemplazar los elementos *[NombreComponenteA]* y *[NombreComponenteB]*, con los nombres de los componentes. Así mismo, deberá reemplazar los nombres de los parámetros y mensajes por sus nombres correspondientes.

La Figura 5.1 muestra un ejemplo de definición de comunicación entre interfaces.



Universidad Tecnológica de León

Modelo de Madurez de la Capacidad Integrado - CMMI

CRITERIOS Y LINEAMIENTOS PARA EL DISEÑO DE INTERFACES

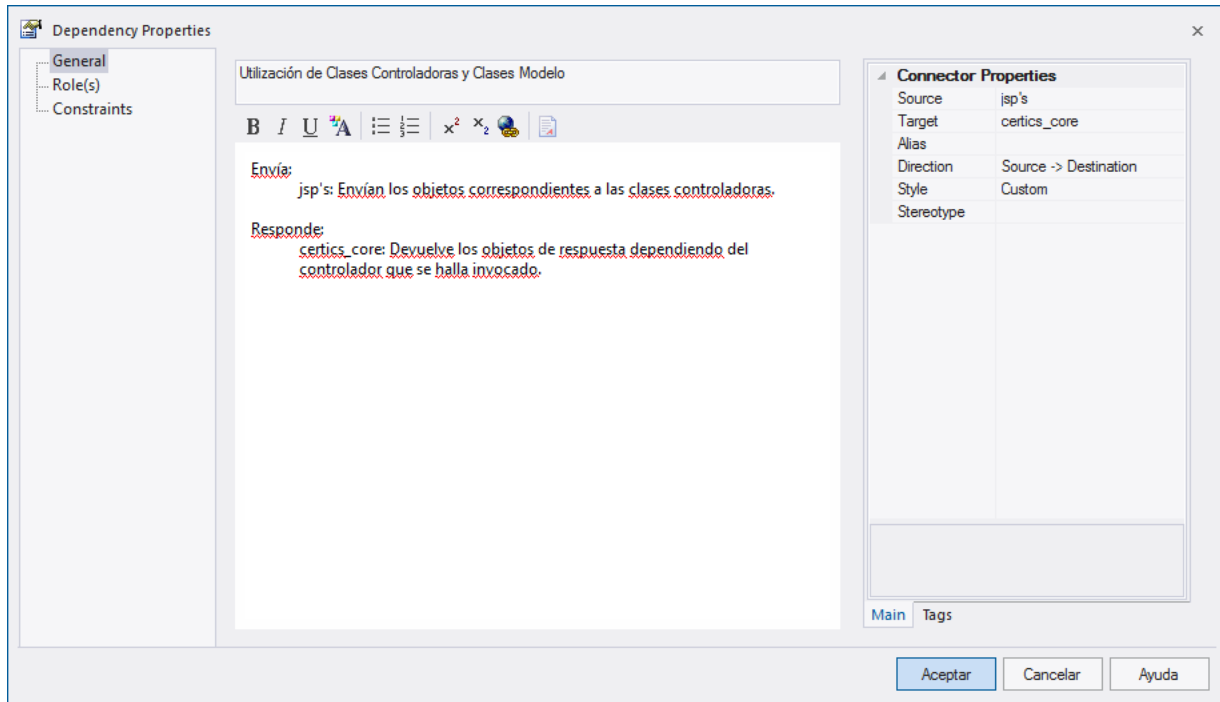


Figura 5.1. Ejemplo de diseño correcto de interfaces de sistema.

La Figura 5.2 muestra un diseño de componentes correcto, en el cual se aplican los criterios antes establecidos, hasta un nivel de detalle de componentes.

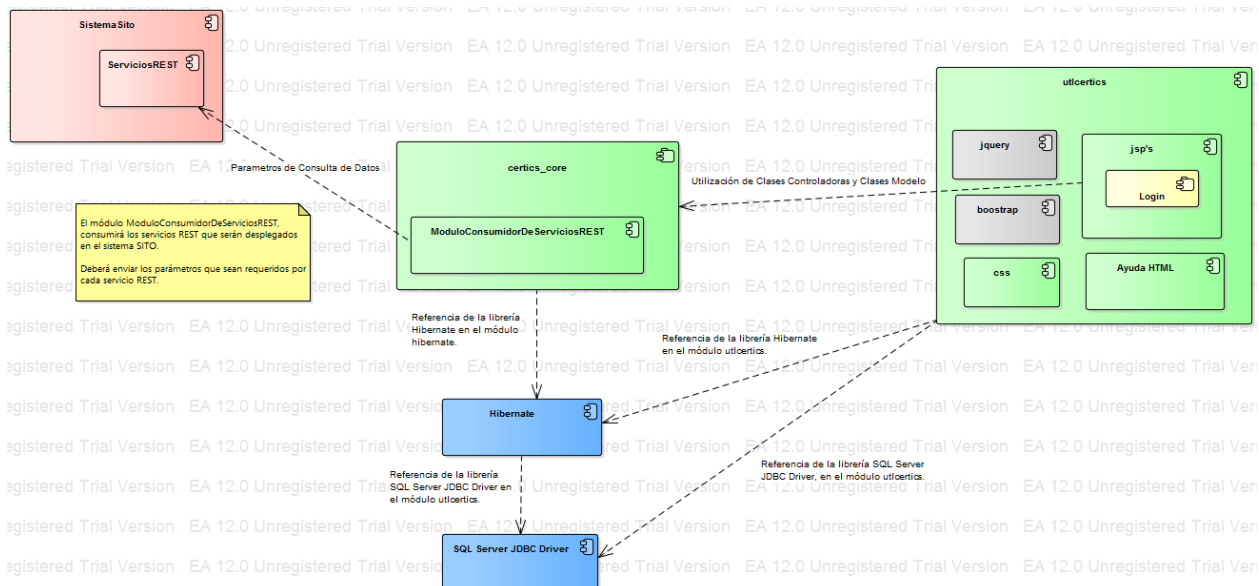


Figura 5.2. Ejemplo de diseño de un diagrama de componentes de alto nivel hasta un nivel de detalle de Componentes.

La Figura 5.3 muestra un diseño de componentes correcto, en el cual se aplican los criterios antes establecidos, hasta un nivel de detalle de paquetes.

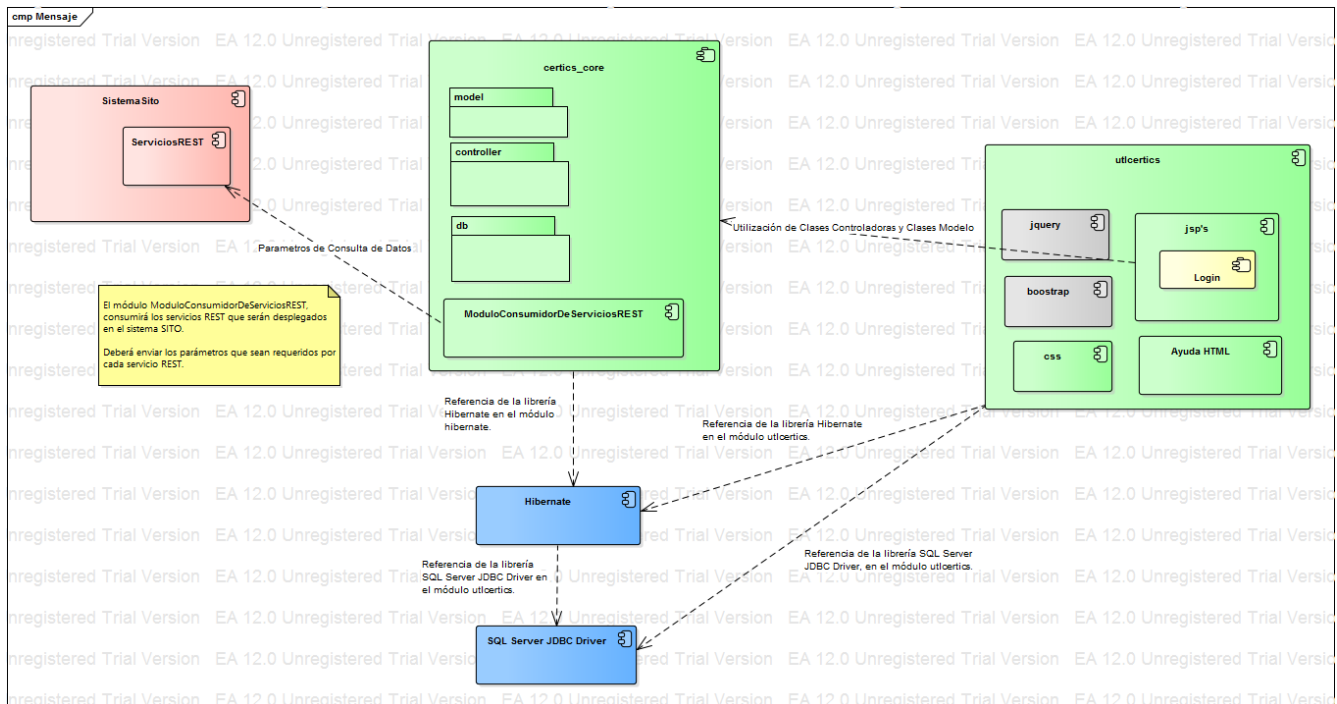


Figura 5.3. Ejemplo de diseño de un diagrama de componentes de alto nivel hasta un nivel de detalle de Paquetes.

Las figuras 5.2 y 5.3, muestran dos ejemplos del nivel de detalle en un diagrama de componentes de alto nivel. Ambos diseños son correctos. La diferencia es que en la Figura 6, el nivel de detalle llegó hasta la definición de paquetes.

6. INTERFACES DE COMPONENTES – DIAGRAMA DE CLASES

A continuación se describen los lineamientos para el diseño de Interfaces de Componentes que son definidos en la sección **06_DiagramaClases**, dentro del documento **InformacionTecnicaDesarrollo.eap**.

1. El diseño de las interfaces de componentes, a nivel de clases se plasma en el documento **InformacionTecnicaDesarrollo.eap**, dentro de la sección **06_DiagramaClases**. A este diseño, se le conoce también como **Diseño Detallado**.



Universidad Tecnológica de León
Modelo de Madurez de la Capacidad Integrado - CMMI
CRITERIOS Y LINEAMIENTOS PARA EL DISEÑO DE INTERFACES

2. Cuando se desea representar un componente, dentro de los diagramas de componentes y clases, éste se representa como un componente y, se le aplica un color de fondo para indicar su estado de reusabilidad de acuerdo al código de colores establecido al inicio de este documento.
3. Cuando se desea representar un componente a nivel de clase, este se representa como un elemento de tipo **Class** y se le aplica un fondo de color, de acuerdo con el código de colores establecido al inicio de este documento.
4. Adicionalmente, al definir una clase asegúrese de describir:
 - d) Una descripción del funcionamiento general de la clase.
 - e) El nombre del autor de la misma.
 - f) El lenguaje de programación.
 - g) La Fase en la que se definió dicho elemento: Inicio, Definición Global, Planeación Global, Análisis y Diseño por Iteración, Construcción por Iteración, Pruebas por Iteración, Implementación por Iteración, Cierre por Iteración o Cierre Global.

La Figura 6.1 indica donde colocar la información antes descrita.

Figura 6.1. Información que debe contener una clase al definirla.

5. La comunicación entre componentes a nivel de clase se clasifica en dos tipos: *uso* y *herencia*. El arquitecto deberá decidir, en base a su propio diseño, la más adecuada considerando los siguientes dos supuestos:

- a) **Uso (*Directed Association*):** Utilice este tipo de comunicación entre clases cuando, por ejemplo, una clase ClaseB *hace uso* ó *necesita* de una clase ClaseA. En este caso, utilice una asociación dirigida de ClaseB hacia ClaseA. Por ejemplo, en la Figura 6.2, la clase Alumno, hace uso de la Clase persona:

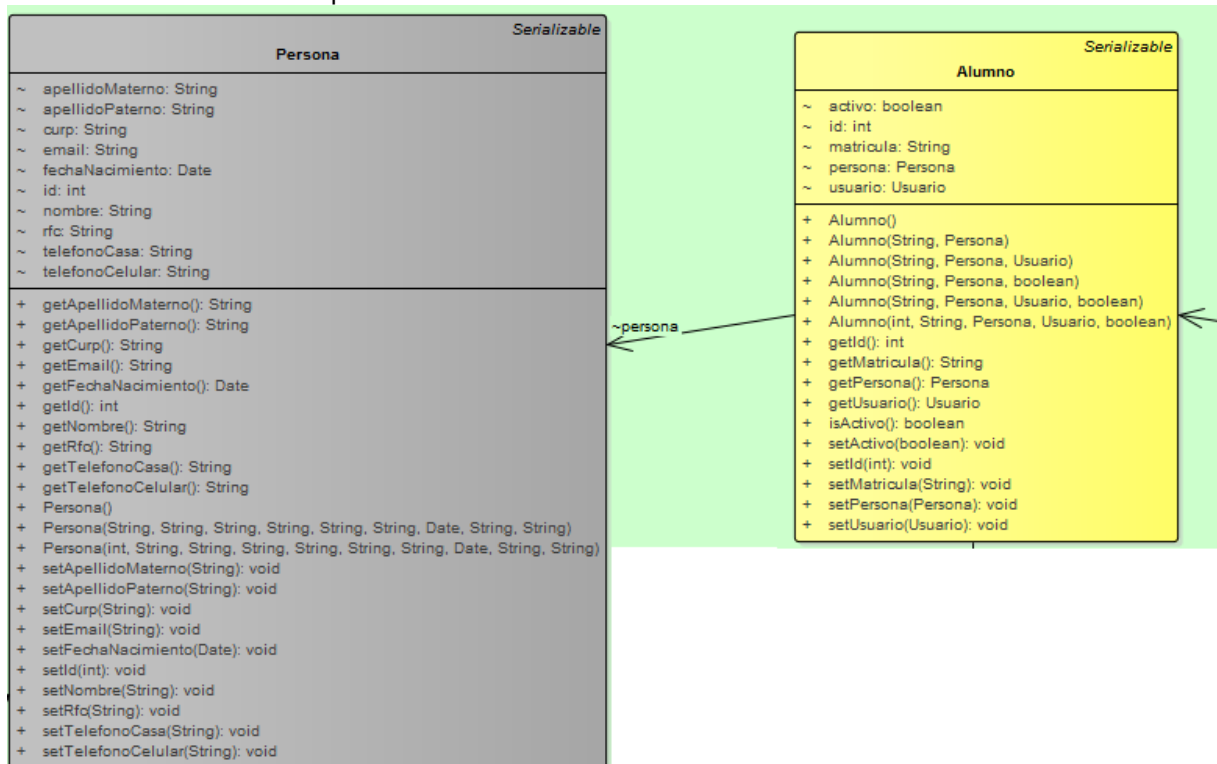


Figura 6.2. Comunicación entre dos clases. La clase Alumno utiliza a la clase Persona.

Adicionalmente, puede especificar un nombre de uso de la asociación. En el ejemplo de la Figura 6.2, el *nombre de uso* se estipuló como *~persona*, sin embargo, esto no es obligatorio, ya que algunas veces, cuando se definen clases por ingeniería inversa, Enterprise Architect genera ese valor por defecto. Sin embargo, su definición u omisión no afecta el diseño del producto. Para conocer más acerca de cómo relacionar dos clases con una asociación de uso, consulte la sección [9.5. Asociación Dirigida de Uso entre Clases](#).

- b) **Herencia (*Realization*):** Utilice este tipo de relación entre clases cuando por ejemplo, una clase ClaseB, hereda de una clase Clase A. En este caso, utilice una asociación dirigida de ClaseB hacia ClaseA. Para conocer más acerca de cómo relacionar dos clases con una asociación de herencia, consulte la sección [9.6. Asociación de Herencia entre Clases](#).

La Figura 6.3 muestra un ejemplo de la relación (comunicación) entre dos clases donde, una hereda de la otra.

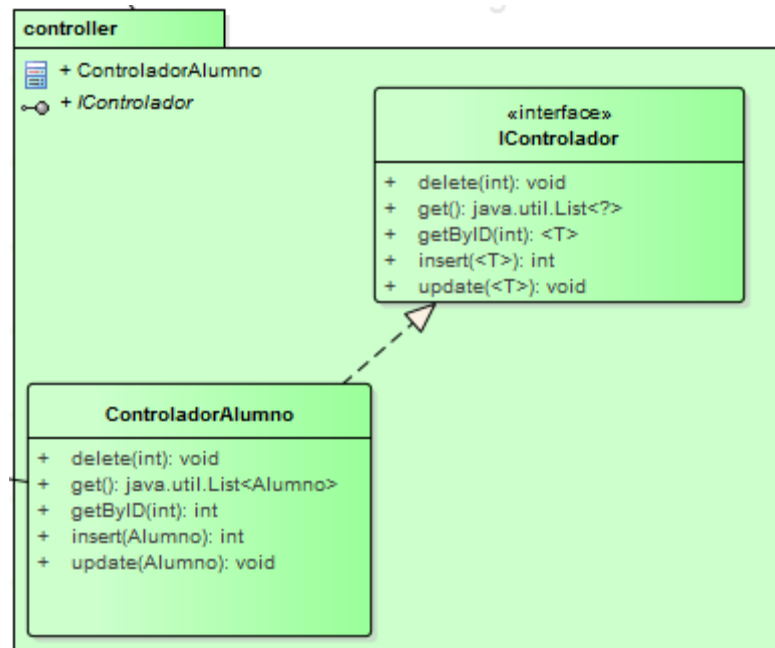


Figura 6.3. Relación de herencia entre dos clases. La clase ControladorAlumno hereda de la clase-interface IControlador.

6. Toda relación (comunicación) entre componentes de tipo clase deberá estar documentada. Esta documentación se hace mediante la descripción de la funcionalidad de cada clase así como cada uno de sus métodos públicos, incluyendo, una descripción del valor de retorno y cada uno de los parámetros correspondientes a cada método.

La Figura 6.4 contiene un ejemplo de la descripción del funcionamiento de un método público.

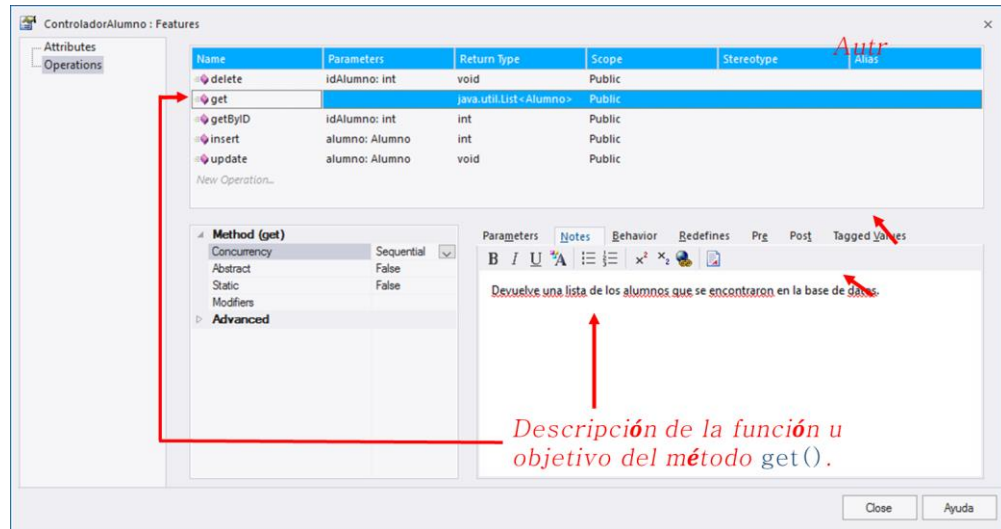


Figura 6.4. Descripción de la funcionalidad de un método público.

De igual forma, la Figura 6.5 muestra la descripción de un parámetro (alumno) correspondiente a un método (insert).

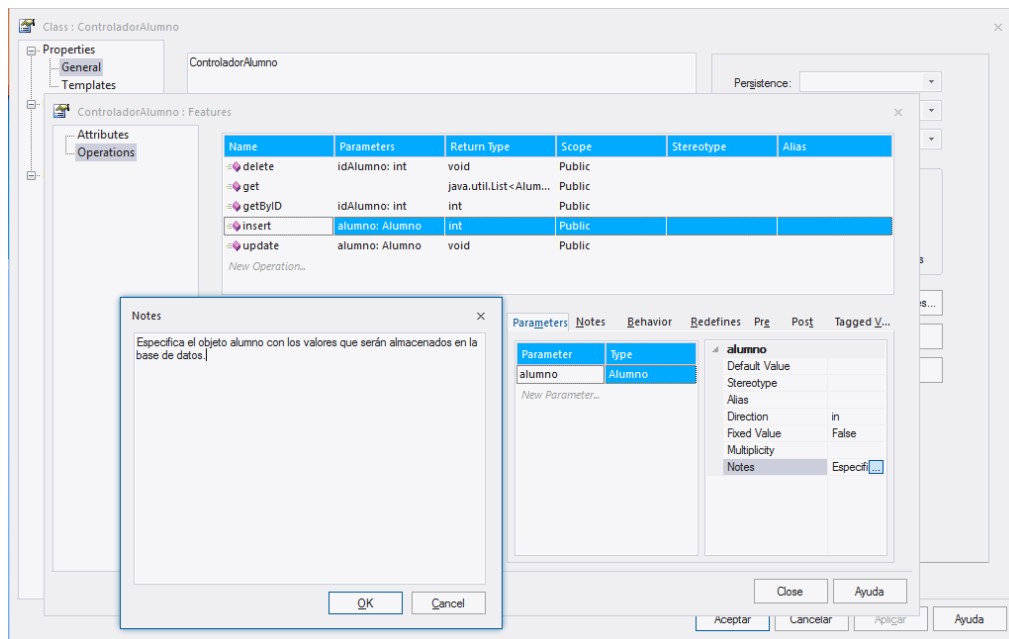


Figura 6.5. Descripción de un parámetro, correspondiente a un método.

7. Diseño de Base de Datos

El diseño del diagrama de bases de datos es muy similar al diagrama de clases. Si el desarrollo de su producto implica el uso de una base de datos relacional, deberá definirla dentro del documento InformaciónTecnicaDesarrollo.eap, siguiendo los siguientes criterios y lineamientos:

1. Cada tabla deberá contener una descripción.
2. Cada tabla deberá contener el nombre del autor de la misma.
3. Aplicar el código de colores a cada tabla definida en la base de datos.
4. Puede utilizar elementos de tipo *Boundary* (delimitador) para marcar zonas o partes de la base de datos y diferenciarlas de otras.

La Figura 7.1 muestra un ejemplo de la definición correcta de una tabla.

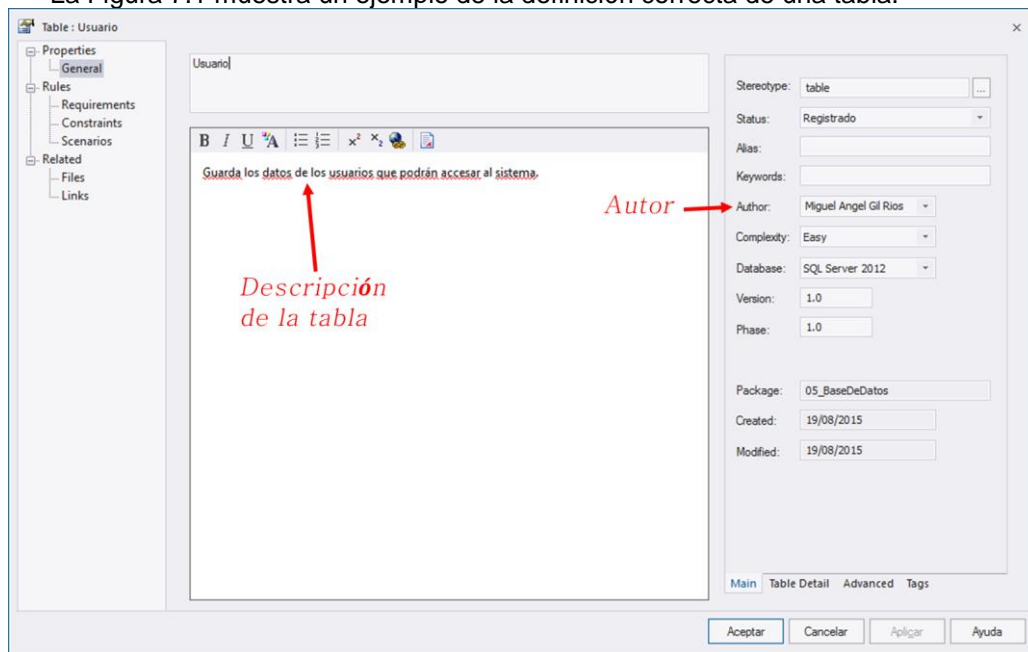


Figura 7.1. Definición correcta de una tabla.

5. A su vez, cada atributo de la tabla deberá contener una descripción del mismo. La Figura 7.2 muestra un ejemplo de la definición correcta de un atributo. En la Figura, el atributo se llama IdUsuario, junto con su descripción.

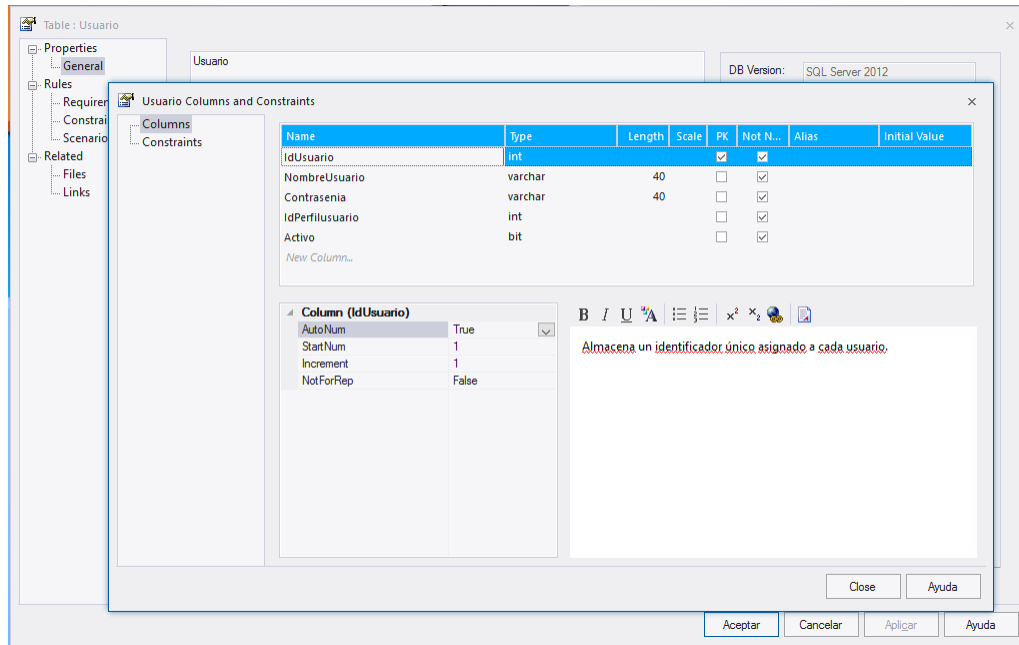


Figura 7.2. Definición correcta de los atributos de una tabla.

Finalmente, la Figura 7.3 muestra un diseño de base de datos utilizando un *Boundary* (delimitador) para marcar una sección de la misma.

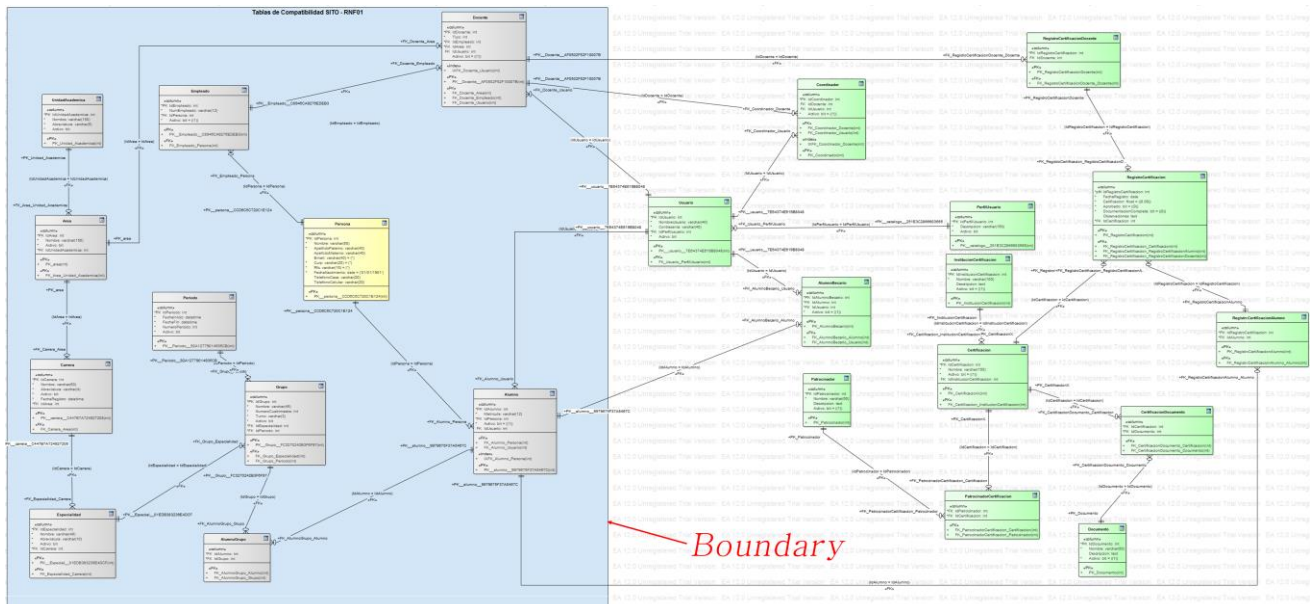


Figura 7.3. Diagrama de Base de Datos utilizando un delimitador (zona azul) para diferenciar dos secciones de la base de datos.



8. Cumplimiento de Requerimientos

Para asegurar que todos los requerimientos definidos, han sido considerados en todas las fases del diseño, deberá asociar cada requerimiento con cada uno de los elementos definidos en cada uno de sus diagramas: 01_ModeladoDeNegocio; 02_Requerimientos; 03_CasosDeUso; 04_Componentes; 05_BaseDeDatos; 06_DiagramaClases.

Nota: Como los diagramas 01_ModeladoDeNegocio y 03_CasosDeUso no siempre se definen, solo se asociarán requerimientos en los casos en los que sí se definan.

La realización de estas acciones es de importancia crítica, pues permitirá tener un rastreo bidireccional entre cada requerimiento y cada componente individual que asegurará su implementación.

8.1. Cumplimiento de Requerimientos en Diagrama de Casos de Uso

Deberá asociar cada requerimiento con los casos de uso que le correspondan.

Nota: Si un caso de uso, abarca la implementación y/o solución de un requerimiento funcional y todos sus requerimientos funcionales derivados, asociar solo el requerimiento principal. Por el contrario, si solo se cubrirán ciertos requerimientos derivados, asociar el requerimiento funcional y sus requerimientos funcionales derivados, respectivos.

8.2. Cumplimiento de Requerimientos en Diagrama de Componentes

En el diagrama de componentes, deberá asociar cada requerimiento con cada uno de los componentes y artefactos definidos en el diagrama, que cumplirán su implementación y/o solución.

Nota: Si un componente abarca la implementación y/o solución de un requerimiento funcional y todos sus requerimientos funcionales derivados, asociar solo el requerimiento principal. Por el contrario, si solo se cubrirán ciertos requerimientos derivados, asociar el requerimiento funcional y sus requerimientos funcionales derivados, respectivos.

8.3. Cumplimiento de Requerimientos en Base de Datos

En el diagrama de base de datos, deberá asociar cada requerimiento con cada una de las tablas involucradas en el cumplimiento de dicho requerimiento.

8.4. Cumplimiento de Requerimientos en Diagrama de Clases

En el diagrama de clases, deberá asociar cada requerimiento con cada una de las clases definidas, que cumplirán su implementación y/o solución.

Nota: Si una clase abarca la implementación y/o solución de un requerimiento funcional y todos sus requerimientos funcionales derivados, asociar solo el requerimiento principal. Por el contrario, si solo se cubrirán ciertos requerimientos derivados, asociar el requerimiento funcional y sus requerimientos funcionales derivados, respectivos.

Al realizar y cumplir con los pasos descritos anteriormente, se podrá asegurar un rastreo bidireccional, en conjunto con el documento de requerimientos (hoja matriz rastreo). Esta práctica es importante porque le ayudará a conocer cómo va transformando y plasmando cada requerimiento en el producto final asegurándose, de no dejar ninguno sin cumplir.

9. GUÍA DE DISEÑO

A continuación, se encuentra una guía concisa para describir los pasos necesarios y, que le ayudarán a cumplir con los criterios de diseño de interfaces descritos anteriormente.

9.1. Agregar Requerimientos

Para agregar un nuevo requerimiento, abra el menú contextual (clic con el botón derecho del mouse) dentro de un área libre del diagrama de requerimientos y, seleccione la opción *New Element or Connector* > *Requirement*. La Figura 9.1 muestra un ejemplo.

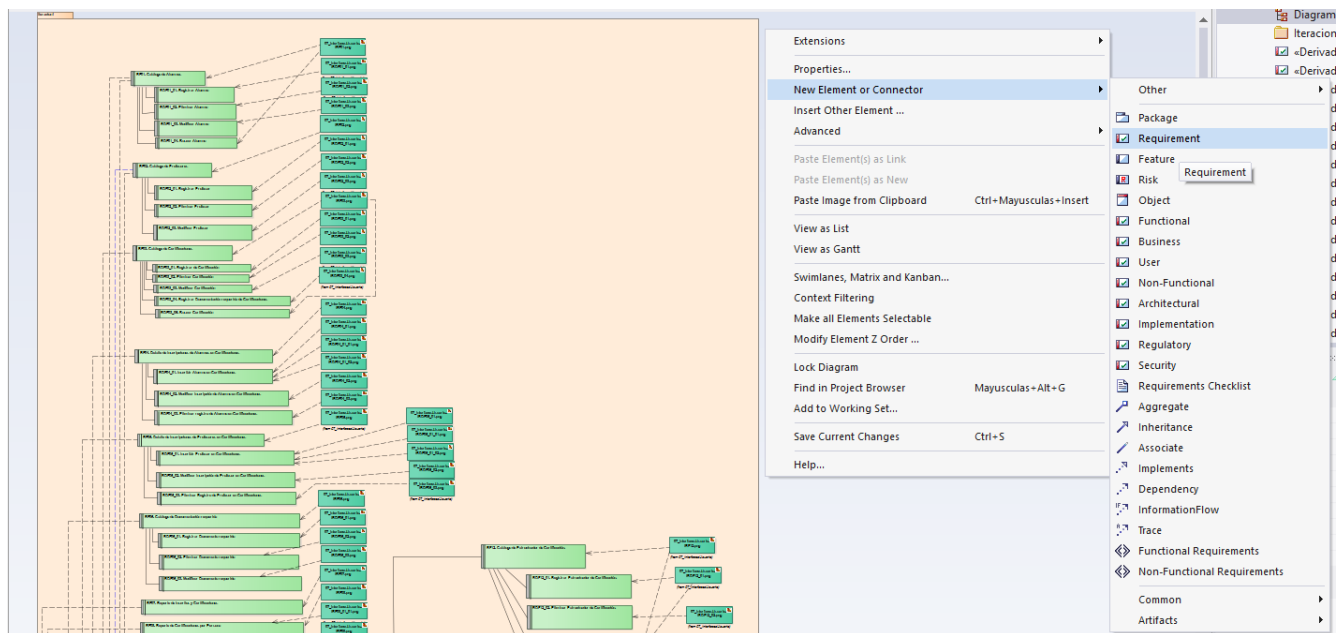


Figura 9.1. Menú Contextual para agregar un nuevo requerimiento.

9.2. Agregar interfaces de usuario como artefactos

Para agregar interfaces de usuario como artefactos siga los siguientes pasos:

1. Asegúrese de tener sus archivos de interface de usuario en formato de imagen JPG o PNG.
2. Vaya al diagrama *DiagramaGeneral* que se encuentra dentro de la sección *07_InterfacesUsuario*, en el documento *InformacionTecnicaDesarrollo.eap*.
3. Arrastre su archivo de imagen, que corresponde a la interface de usuario, dentro del área del diagrama de interfaces de usuario.

4. Cuando Enterprise Architect detecta la acción anterior, mostrará un menú contextual con varias opciones. Seleccione la opción *Artifact (Internal)*. En ese momento su interface de usuario será agregada dentro de su documento como un artefacto interno.

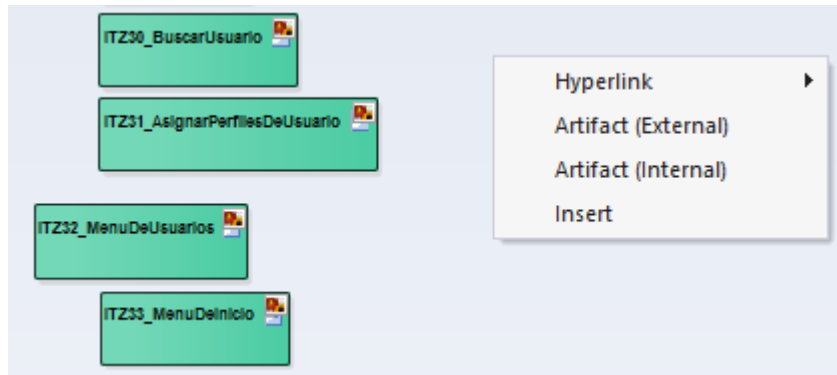


Figura 9.2. Inserción de una interface de usuario al diagrama de interfaces de usuario.

5. Agregue un nombre a su interface de acuerdo con el estándar de nombramiento de interfaces de usuario, dentro del documento *EstandaresConfiguracion_X.X.xlsx*. Por ejemplo: *ITZ01_CatalogoAlumnos.png*.

9.3. Asociación de interfaces de usuario con requerimientos

Para asociar interfaces de usuario con requerimientos siga los siguientes pasos:

1. Abra su diagrama de requerimientos *DiagramaGeneral*, que se encuentra en la sección 02_Requerimientos.
2. Expanda la sección *07_InterfacesUsuario*.
3. Arrastre cada interface de usuario de la sección de interfaces de usuario al diagrama de requerimientos.
4. Al realizar la acción anterior, se desplegará un cuadro de diálogo. Seleccione la opción *Drop as Link*, La Figura 9.3 muestra un ejemplo.

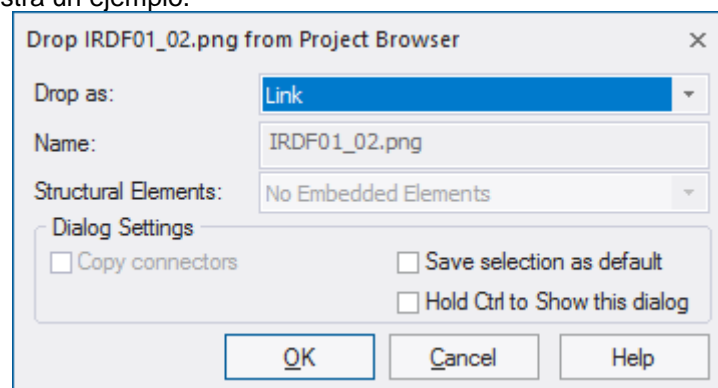


Figura 9.3. Diálogo que aparece al insertar una interface de usuario en el diagrama de requerimientos.

5. Realice una asociación de dependencia entre la interface de usuario y el requerimiento. La Figura 9.4 muestra un ejemplo de la asociación entre interfaces de usuario y requerimientos.

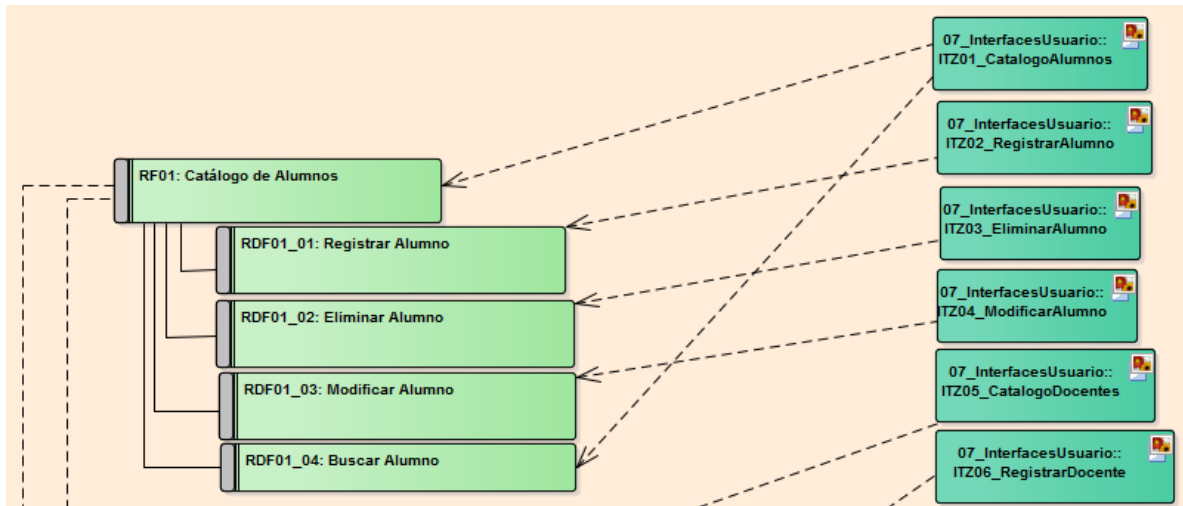


Figura 9.4. Asociación de dependencia entre interfaces de usuario y requerimientos.

9.4. Asociación Dirigida de Dependencia entre Elementos

La asociación dirigida entre elementos, es también una asociación de dependencia. En este tipo de asociación es de vital importancia, el sentido de las flechas (asociaciones). Por eso se le denomina *Asociación Dirigida*.

Para definir una asociación dirigida entre dos elementos, siga los siguientes pasos, tomando en cuenta el escenario descrito:

Se requiere asociar una interfaz de usuario que ha sido agregada al diagrama de requerimientos, con uno de los requerimientos.

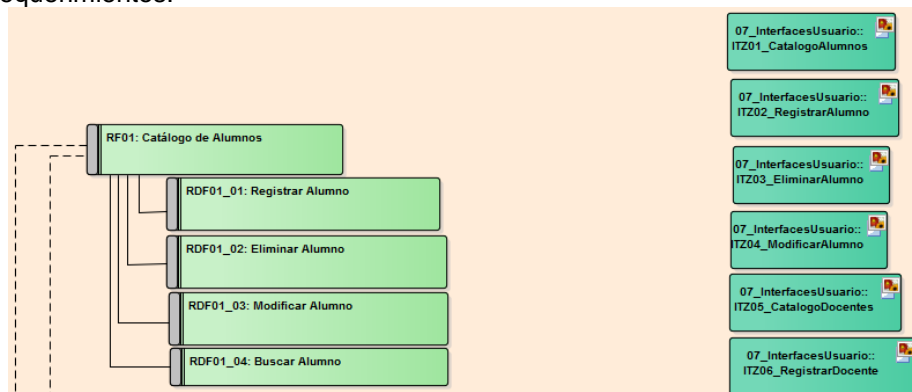


Figura 9.5. Definición de Requerimientos e Interfaces de Usuario sin relacionar.

1. Seleccione el **elemento dependiente**. En este caso, es la interfaz de usuario.

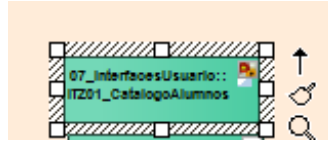
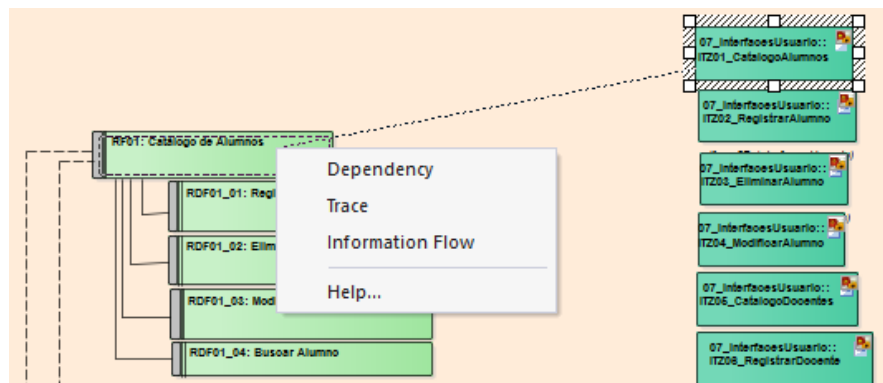


Figura 9.6. Selección del elemento dependiente que se desea relacionar. En este caso, una interface de usuario.

2. Sitúe el mouse en la esquina superior derecha, sobre la flecha, del elemento dependiente, haga clic y, sin soltarlo, arrastre hasta el requerimiento (elemento independiente) que desea relacionar y suelte el botón del mouse.
3. En ese momento, aparecerá un menú contextual indicando varias opciones de tipo de relación.



9.7. Menú contextual para seleccionar el tipo de relación entre elementos.

4. Seleccione la opción *Dependency* (dependencia). En ese momento, quedará definida una nueva asociación dirigida (relación de dependencia) entre dos elementos. La Figura 9.8 muestra los elementos asociados de forma dependiente. Para el caso de este ejemplo, el elemento ITZ01_CatalogoAlumnos, depende del elemento RF01: Catálogo de Alumnos.

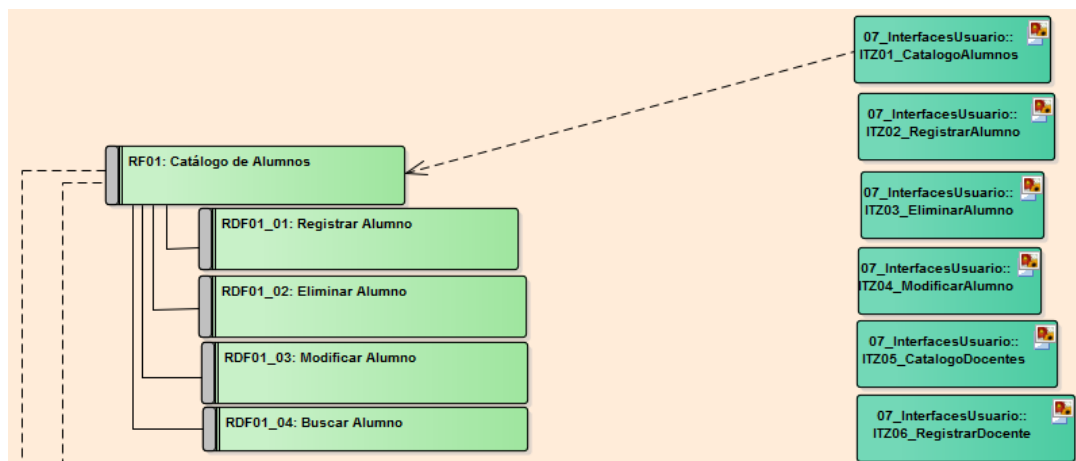


Figura 9.8. Asociación de una interface de usuario con un requerimiento. La interface de usuario depende del requerimiento.

La asociación dirigida puede estar presente, además del diagrama de requerimientos, en el diagrama de componentes y en el diagrama de clases.

9.5. Asociación Dirigida de Uso entre Clases

La asociación dirigida de uso entre clases se realiza de manera muy similar a la relación de dependencia, mostrada en el punto 9.5. Por lo tanto, siga los mismos pasos que en el punto 9.4 y, al aparecer el menú contextual, seleccione la opción *Use*.

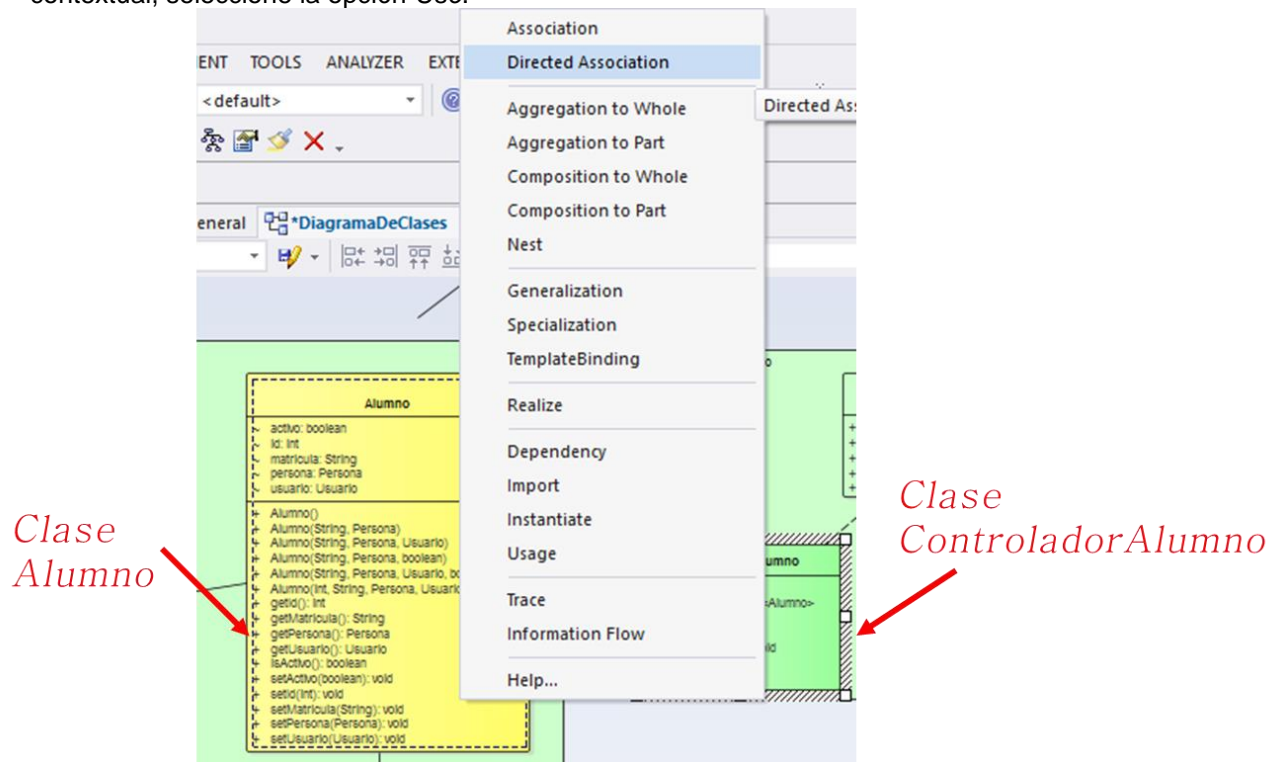


Figura 9.9. Ejemplo del menú contextual para relacionar dos clases. En ese caso, la clase ControladorAlumno, necesita o *usa* a la clase Alumno.

9.6. Asociación de Herencia entre Clases

La asociación de herencia entre clases, es muy similar a las asociaciones antes descritas. La Figura 9.10, muestra dos clases que requieren de una asociación de herencia: La clase ControladorAlumno hereda de la clase IControlador.

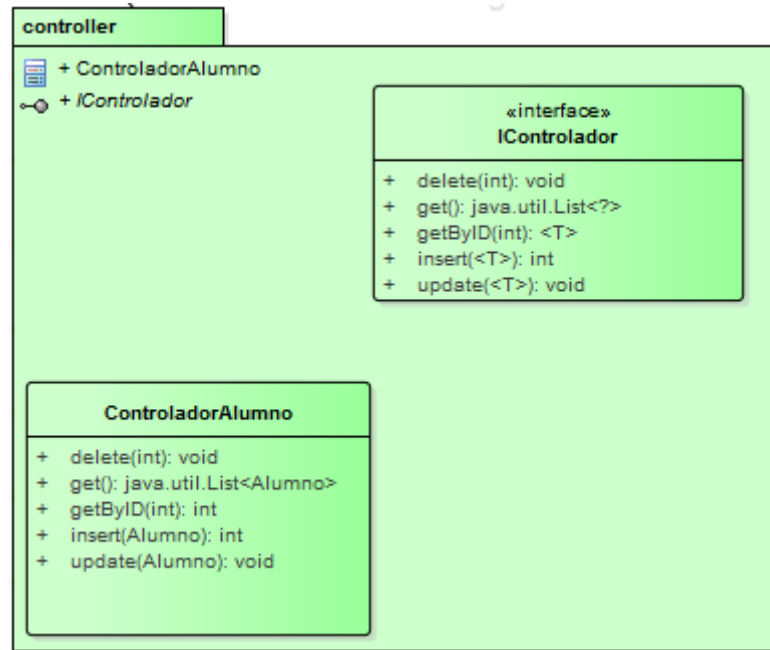


Figura 9.10. Ejemplo de dos clases que deben ser relacionadas con una asociación de herencia.

Para relacionar dos clases con una asociación de herencia, siga los siguientes pasos:

1. Seleccione la clase heredada. En este ejemplo, es la clase *ControladorAlumno*.
2. Arrastre el elemento de relación (la flecha que aparece en la parte superior derecha de la clase), a la clase padre, en este caso, a la clase *IControlador*.
3. En ese momento, aparecerá un menú contextual. Seleccione la opción *Realization*. La figura 9.11, muestra un ejemplo.

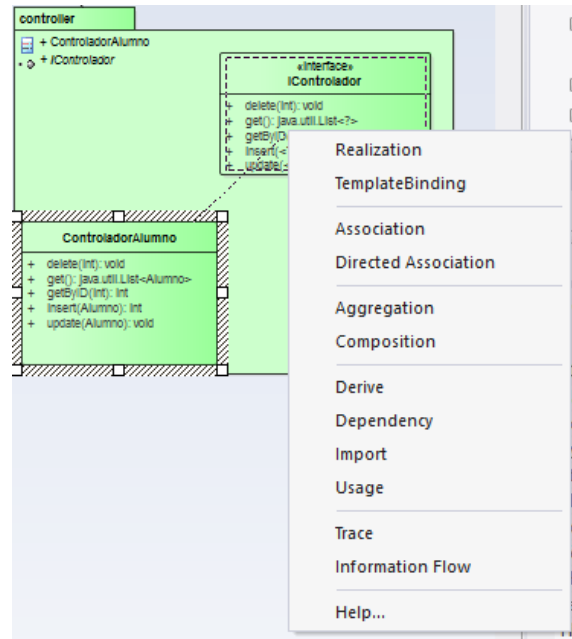


Figura 9.11. Asociación de herencia entre dos clases.

Una vez realizada la asociación, las clases quedan relacionadas con una relación de tipo herencia. La Figura 9.12 muestra un ejemplo.

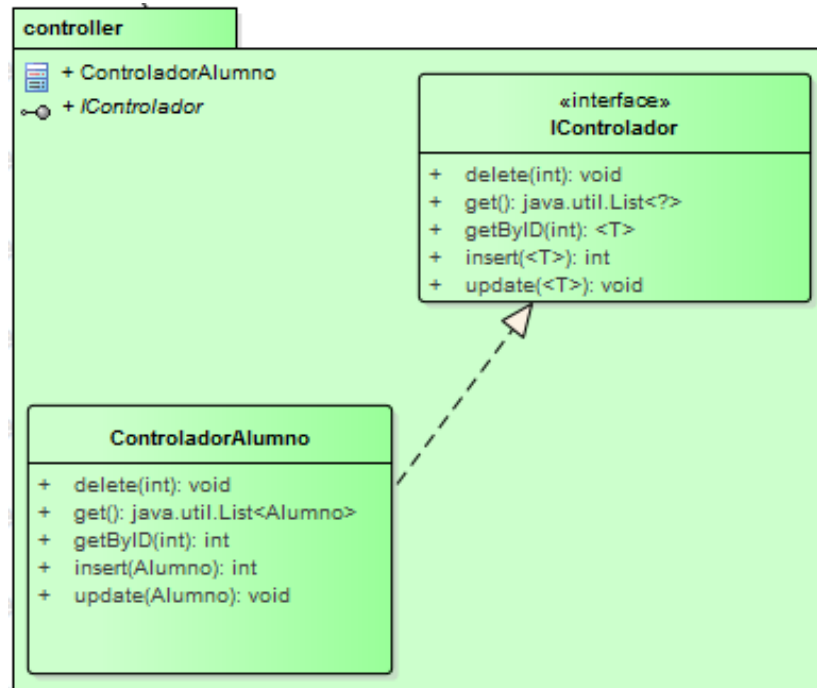


Figura 9.12. Ejemplo de dos clases relacionadas por asociación de herencia. En el ejemplo, la clase ControladorAlumno hereda de la clase IControlador.

9.7. Asociación de contención de Clases y Paquetes.

La asociación de contención de clases y paquetes es muy similar a las asociaciones dirigidas. La diferencia es que estas se realizan entre clases (elementos dependientes) y paquetes (elementos independientes). El procedimiento es similar al de una asociación, sin embargo, al aparecer el menú contextual de asociación, se debe seleccionar la opción *Nesting*.

La Figura 9.13 muestra un ejemplo del menú contextual de asociación con la opción *Nesting*.

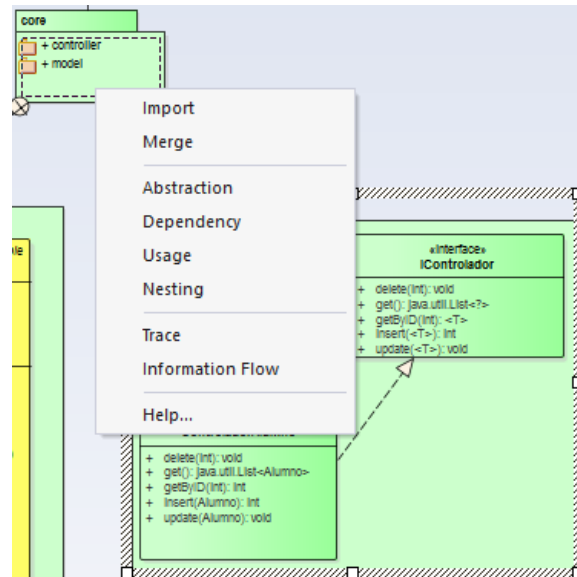


Figura 9.13. Menú contextual para asociar un paquete (controller) con otro (core).

Una vez que se selecciona la opción *Nesting*, los elementos quedarán con una relación de contención, como se muestra en la Figura 9.14.

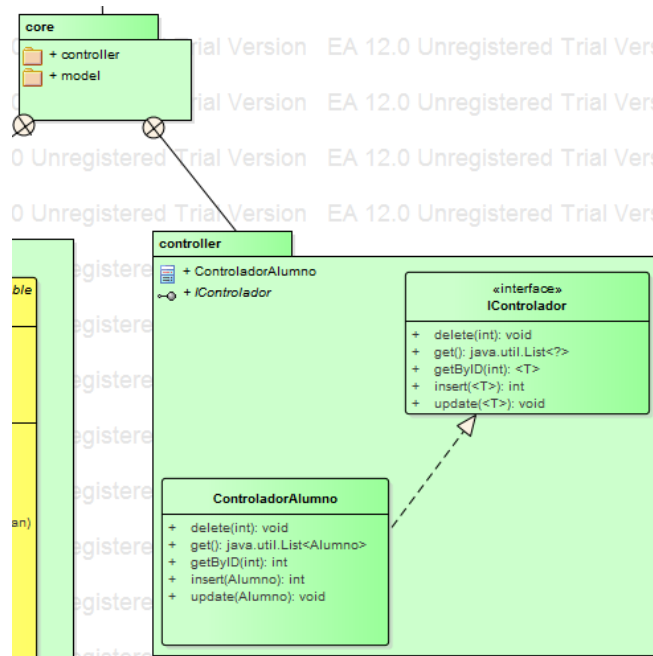


Figura 9.14. Asociación por contención de dos paquetes. En el ejemplo de la figura, el paquete *controller* está contenido dentro del paquete *core*.

9.8. Uso de un área delimitadora (Boundary) en diagramas.

El uso de áreas delimitadoras (boundaries) es útil cuando se desean distintas secciones dentro de un mismo diagrama.

Para agregar un área delimitadora dentro de un diagrama, siga los siguientes pasos:

1. Haga clic con el botón derecho del mouse sobre un área sin elementos de su diagrama.
2. En el menú contextual que aparece, seleccione la opción *Boundary*, que se encuentra en la ruta: *New Element or Connector > Common > Boundary*. La Figura 9.15 muestra la opción *Boundary*.

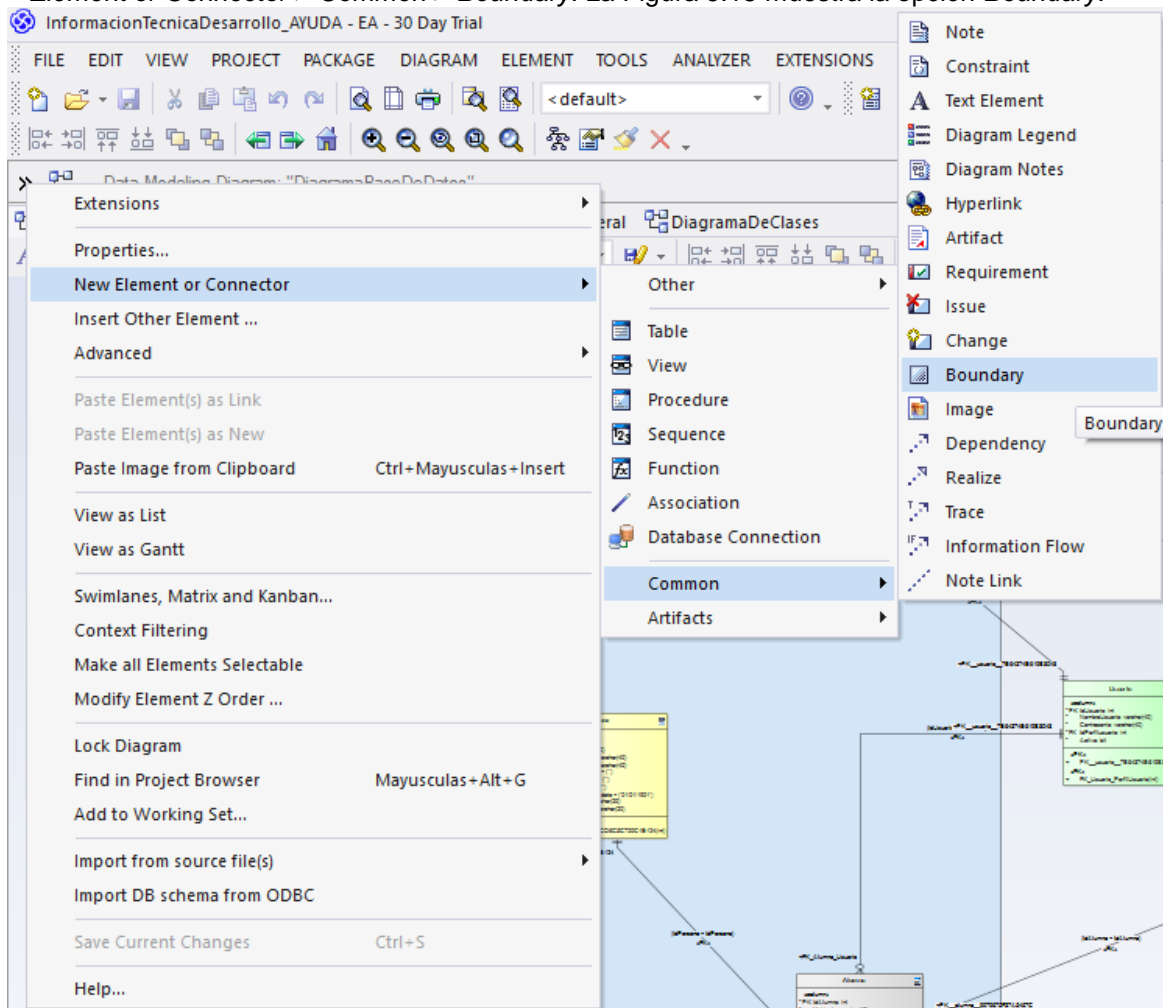


Figura 9.15. Opción Boundary para generar un área delimitadora.

3. Delimite el alcance del área delimitadora.

Es importante mencionar que el uso de áreas delimitadoras no afecta los diseños, la interpretación o el alcance del documento y su uso, es solamente visual para ayudar a comprender de forma rápida, diferentes áreas y partes de un mismo diagrama.