LOGALI

Teoría

ABAP Cloud Developer Extensibility – BAdI

SAP S/4HANA Cloud – Modelo de extensibilidad Clean Core

Contenido

1. ABAP Cloud Developer Extensibility – BAdI

1.1. ABAP Repository Tree – Objetos Liberados

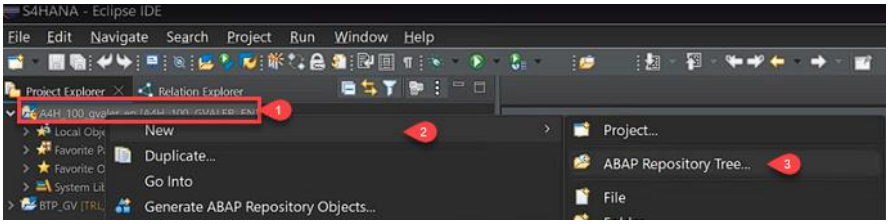
El ABAP Repository Tree es una herramienta en el entorno de desarrollo ABAP de SAP que permite organizar y visualizar los objetos liberados, como APIs y BAdIs (Business Add-Ins). Estos objetos liberados son componentes estándar de SAP que pueden ser extendidos o modificados sin cambiar el código fuente original, facilitando la personalización y actualización del sistema. Los BAdIs permiten implementar mejoras orientadas a objetos, proporcionando flexibilidad y compatibilidad con futuras actualizaciones.

La configuración del ABAP Repository Tree en Eclipse permite crear jerarquías que facilitan la identificación y análisis de estos objetos, permitiendo a los desarrolladores y consultores funcionales trabajar en equipo para agregar o modificar funcionalidades de acuerdo con los procesos de negocio específicos del cliente.

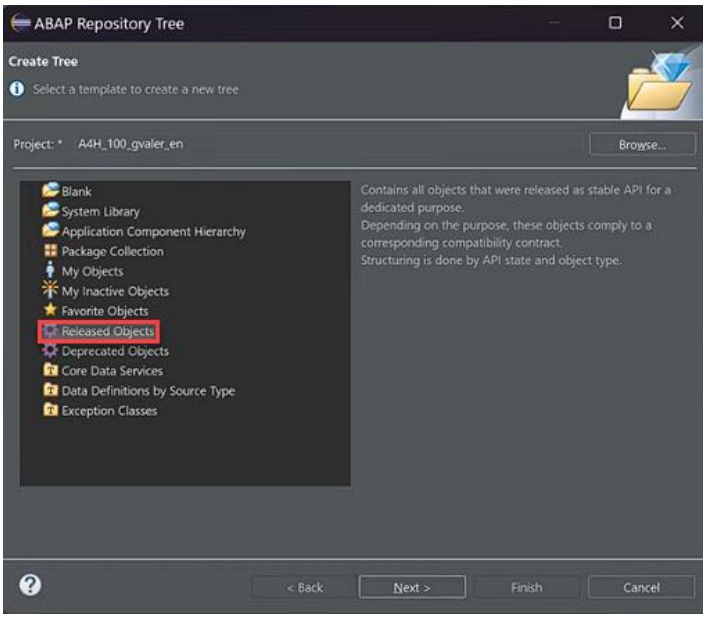
Organización del Entorno de Desarrollo:

Utilizando la herramienta Eclipse, se puede agregar un árbol del repositorio ABAP al entorno de trabajo. Para identificar los objetos liberados que son relevantes para aplicar cambios. De la siguiente manera:

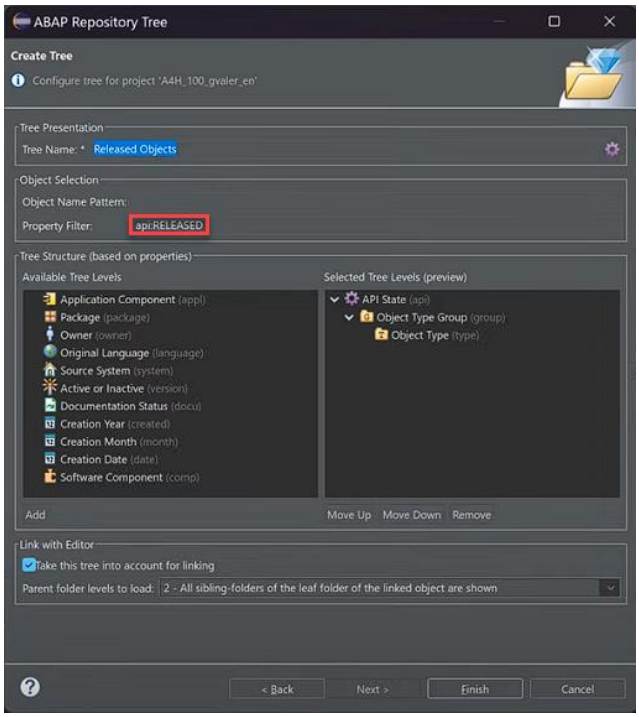
- Hacer clic derecho en el nombre del proyecto y seleccionar la opción "New" para crear un "ABAP Repository Tree".



Seleccionar la opción “Released Objects” que corresponden a los objetos o las APIs liberadas para establecer la jerarquía del árbol. Y continuar con el proceso presionando el botón “Next”.

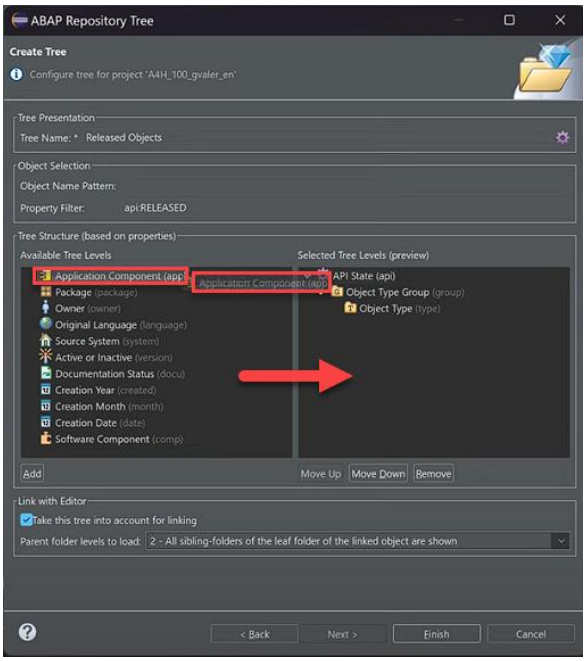


En la siguiente ventana se configura el árbol de la forma en que se representará con filtros como el estado de la API, entre muchos otros los grupos de tipos de objetos. Previamente filtrados al haber seleccionado la opción “Released Objects” y representado en la ventana con “api.Released” en el campo Property Filter.

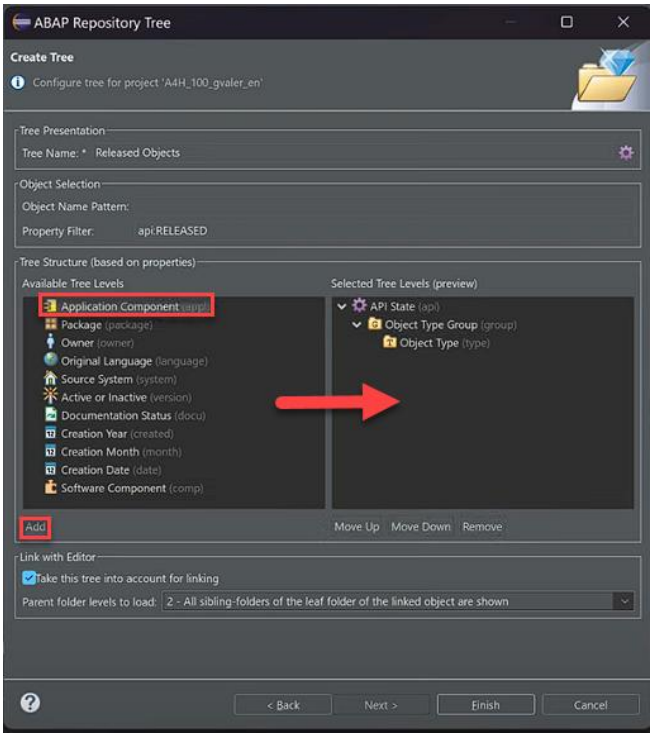


La sección a la derecha con el título “Selected Tree Levels (Preview)” corresponde a los componentes de la aplicación que se mostrarán en el árbol del repositorio.

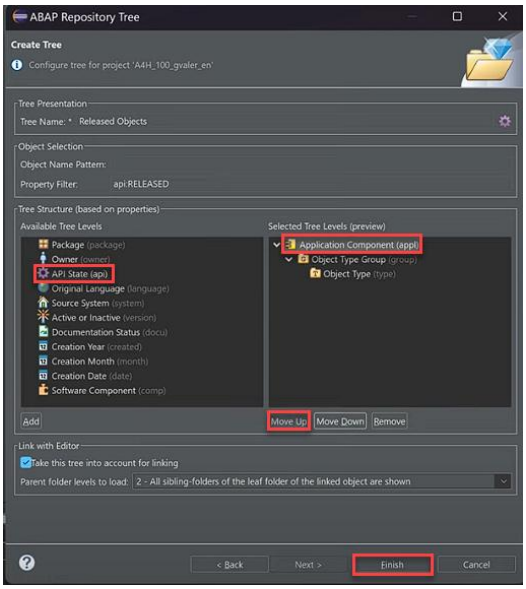
Donde es posible administrar los componentes de la aplicación al arrastrar de la sección de la izquierda a la derecha para que sean visibles y ocultar componentes al contrario.



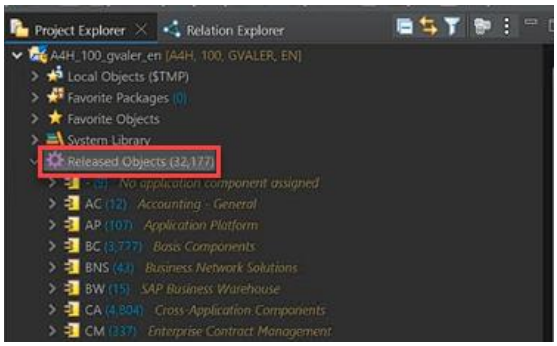
También es posible mover componentes de la aplicación a la parte superior de la jerarquía utilizando el botón "Move Up". En este caso es necesario habilitar el componente **Application Component (appl)**, por lo que se moverá a la sección de la izquierda y se subirá al inicio del árbol.



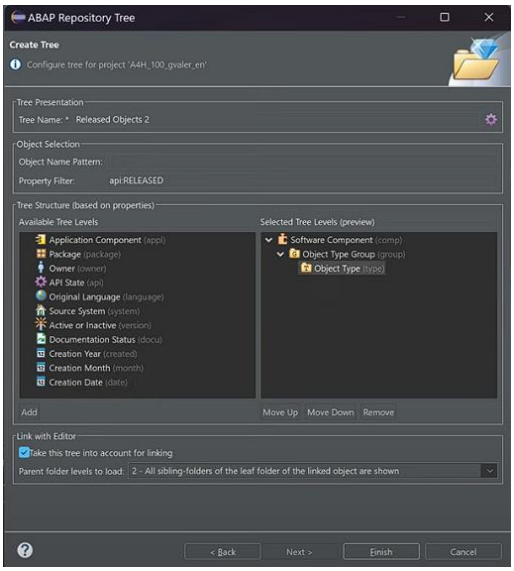
También se pueden ocultar componentes no necesarios al seleccionarlos y presionar el botón “Remove” y ajustar la jerarquía para una fácil visualización y análisis. Para finalizar el proceso con “Finish”.



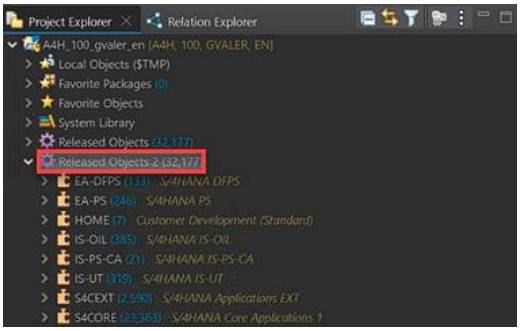
Mostrándose en el editor en una carpeta al final del proyecto.



Cabe destacar que se pueden crear más árboles de repositorio habilitando componentes diferentes para un análisis diferente.



Mostrándose al final de las carpetas dentro del proyecto.

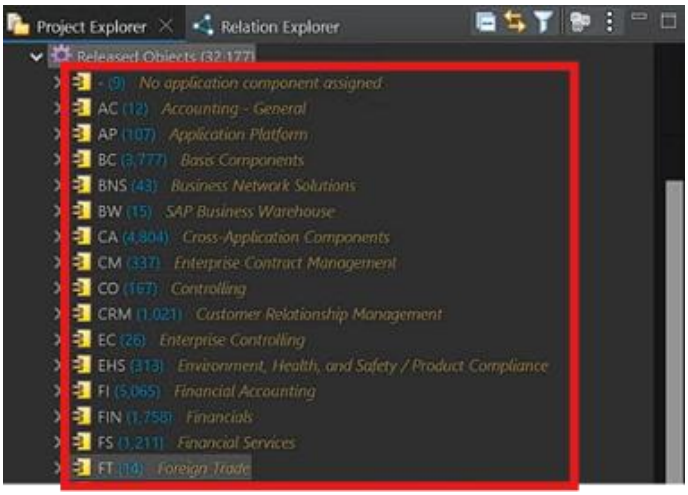


1.2. Puntos de extensión – Análisis

El análisis de puntos de extensión en SAP es un proceso que permite complementar la funcionalidad estándar de las aplicaciones SAP con lógica personalizada que requiere el negocio. Este proceso involucra la identificación de puntos específicos (enhancement spots) donde se pueden implementar mejoras sin alterar el código fuente estándar proporcionado por SAP.

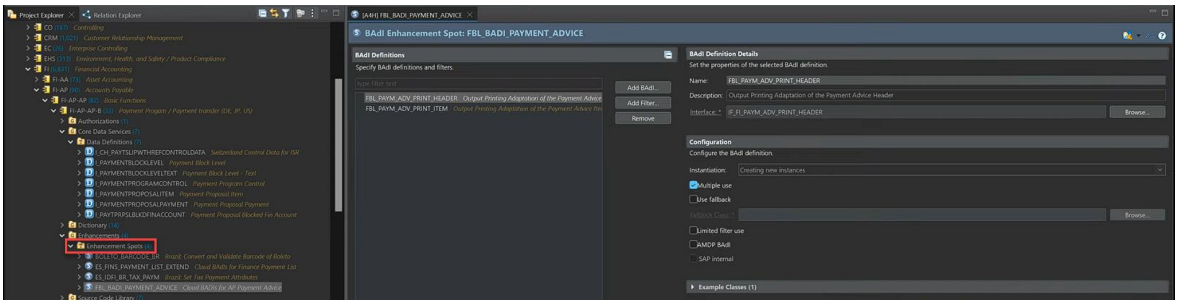
Identificar el componente de negocio:

En donde por lo general el equipo funcional debe proporcionar información detallada sobre el componente de negocio, el módulo o el proceso específico que necesita ser complementado por el equipo técnico. Esta comunicación es crucial para entender las necesidades del negocio y planificar adecuadamente las mejoras necesarias.



Identificación de Enhancement Spots

Una vez que se ha establecido la comunicación, el siguiente paso es identificar los enhancement spots disponibles. Estos puntos de extensión permiten implementar lógica personalizada en momentos específicos o "puertas" abiertas por SAP. Estos puntos son esenciales para permitir que el cliente ejecute código personalizado sin modificar la funcionalidad estándar.

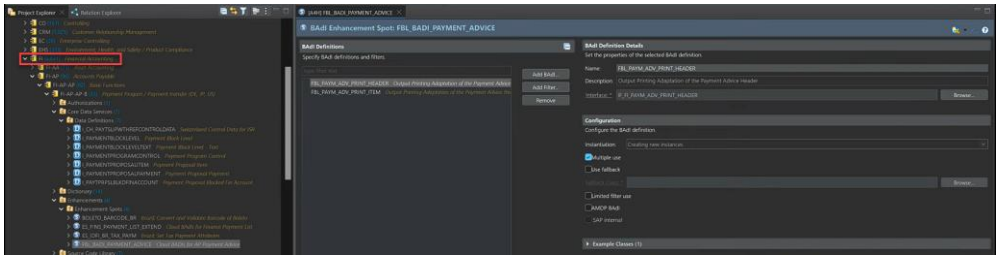


Organización del Entorno de Desarrollo

Para llevar a cabo el análisis, se debe organizar el entorno de desarrollo, utilizando el árbol del repositorio ABAP (ABAP Repository Tree). Este árbol permite identificar y organizar los objetos liberados (released objects) por módulo o componente de aplicación. Aquí se puede clasificar y desplegar los componentes para observar las posibilidades de extensión disponibles en módulos específicos como Financial Accounting (FI), Controlling (CO), logística, entre otros.

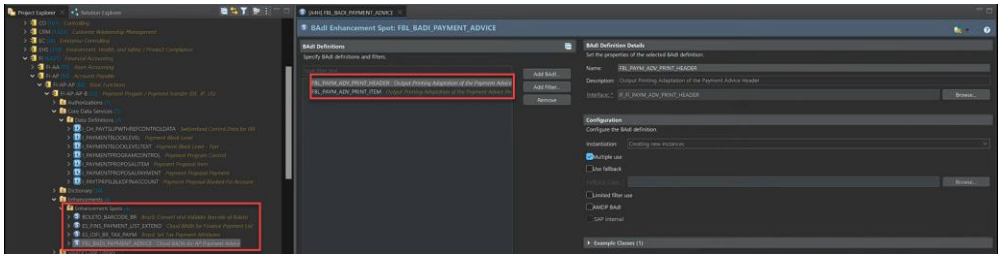
Análisis y Selección de Enhancement Spots

- **Módulos Específicos:** Se debe identificar el módulo específico en el árbol del repositorio, por ejemplo, Financial Accounting. Dentro de este módulo, se pueden observar las funcionalidades disponibles en categorías como basic functions entre otros.

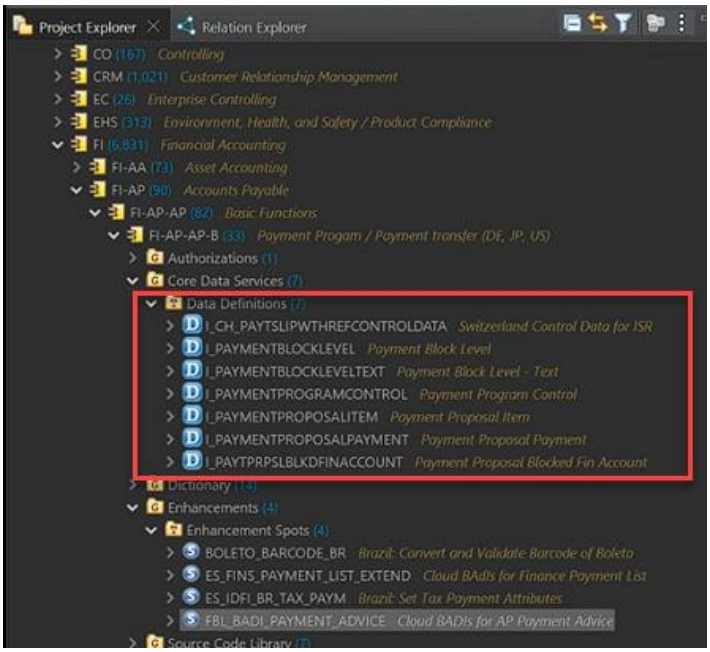


- **Identificación Detallada:** Se debe analizar cada funcionalidad de manera detallada. Para determinar cuál funcionalidad es la más adecuada para implementar la lógica personalizada.

- **Identificación de Enhancement Spots:** Una vez que se ha establecido la comunicación, el siguiente paso es identificar los enhancement spots disponibles. Estos puntos de extensión permiten implementar lógica personalizada en momentos específicos o "puertas" abiertas por SAP. Estos puntos son esenciales para permitir que el cliente ejecute código personalizado sin modificar la funcionalidad estándar.



- **Uso de APIs Liberadas y CDS:** Además de los enhancement spots, también se pueden utilizar objetos de modelado Core Data Services (CDS) disponibles. Estos objetos pueden ser de gran utilidad para recuperar información y complementar los reportes o cualquier otra funcionalidad necesaria.

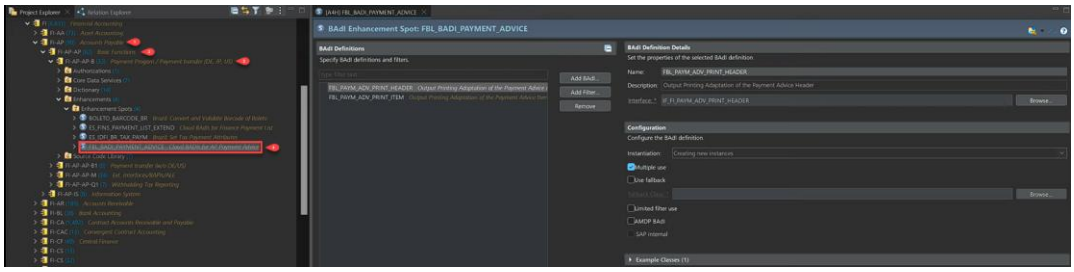


Un análisis minucioso y detallado es la clave para una implementación exitosa de las mejoras necesarias en las aplicaciones SAP. En la siguiente lección, se procederá con la implementación práctica del enhancement spot seleccionado.

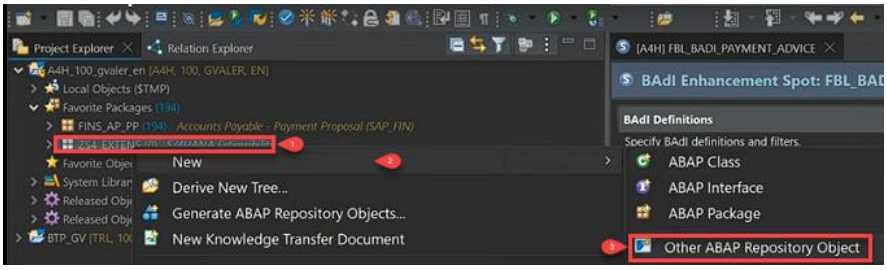
1.3. BAdI Enhancement Implementation

La implementación de mejoras en SAP a través de los Business Add-Ins (BAdIs) es un proceso que permite complementar la funcionalidad estándar de SAP con lógica personalizada según las necesidades específicas del negocio. Este proceso se realiza mediante puntos de extensión conocidos como enhancement spots, que permiten insertar lógica personalizada sin alterar el código fuente estándar.

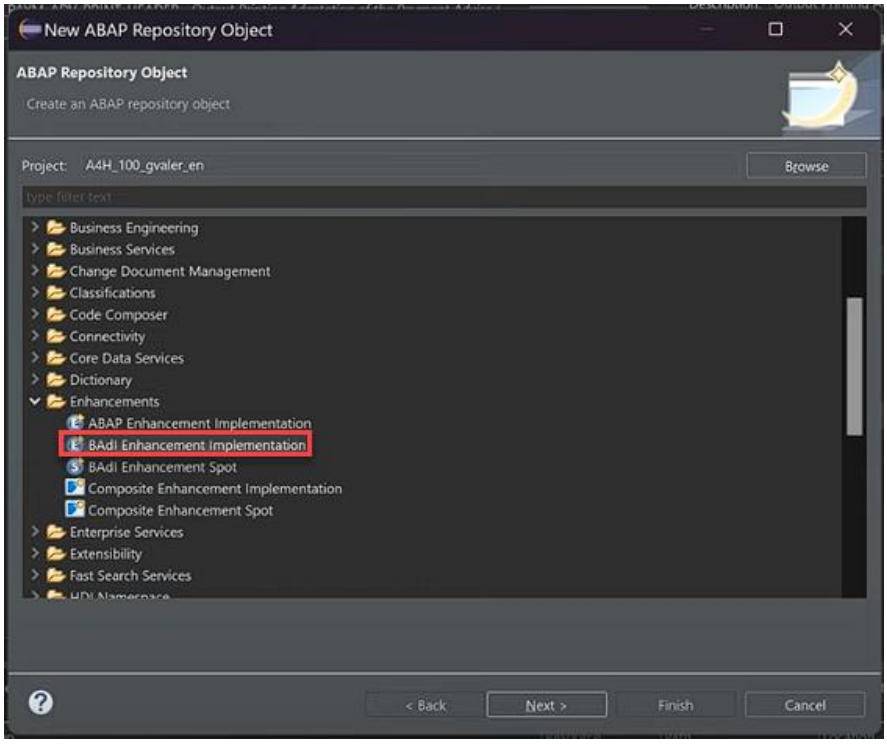
Después del análisis para decidir qué enhancement spot utilizar se obtiene el nombre. Se debe crear la ampliación de la implementación de la BAdI para ajustarla a la necesidades requeridas. En este caso, se selecciona el enhancement spot relacionado con Payment Advice en el módulo de Financial Accounting (FI), específicamente en Accounts Payable -> Payment Program…-> “FBL_BAdI_PAYMENT_ADVICE” .



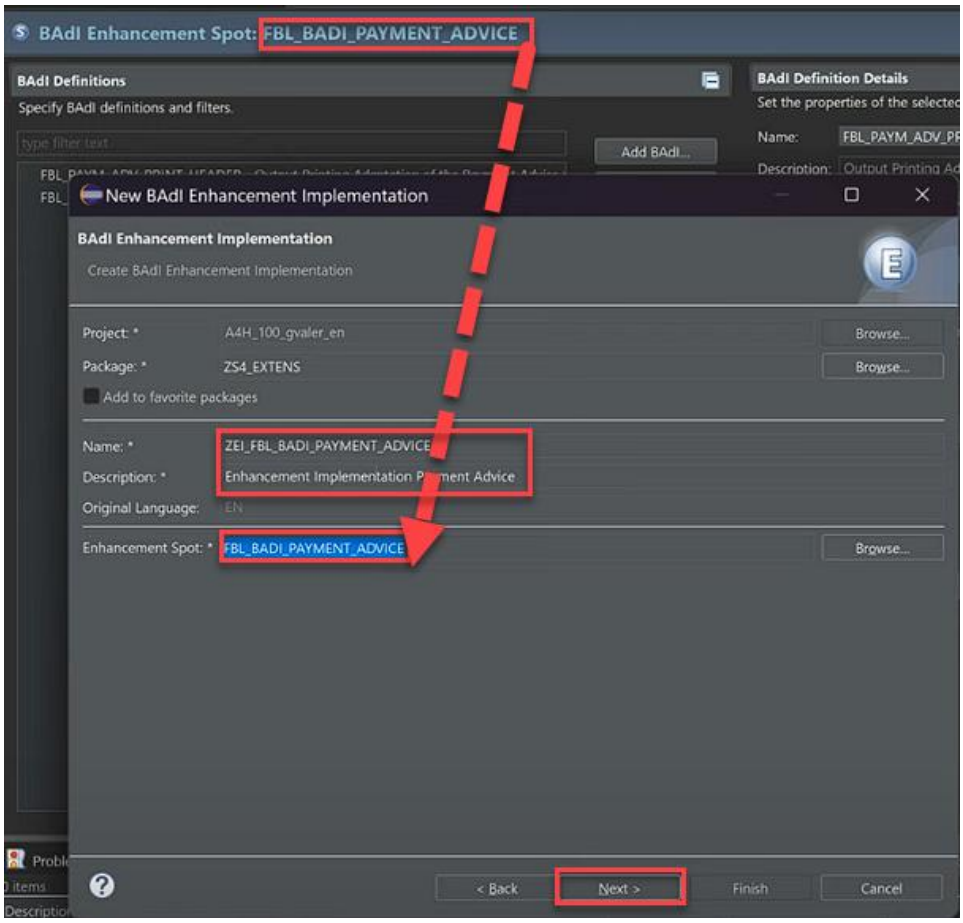
Se utiliza un paquete de desarrollo específico para implementar el enhancement spot seleccionado. Para este proceso al hacer clic derecho y seleccionar New->Other ABAP Repository Object.



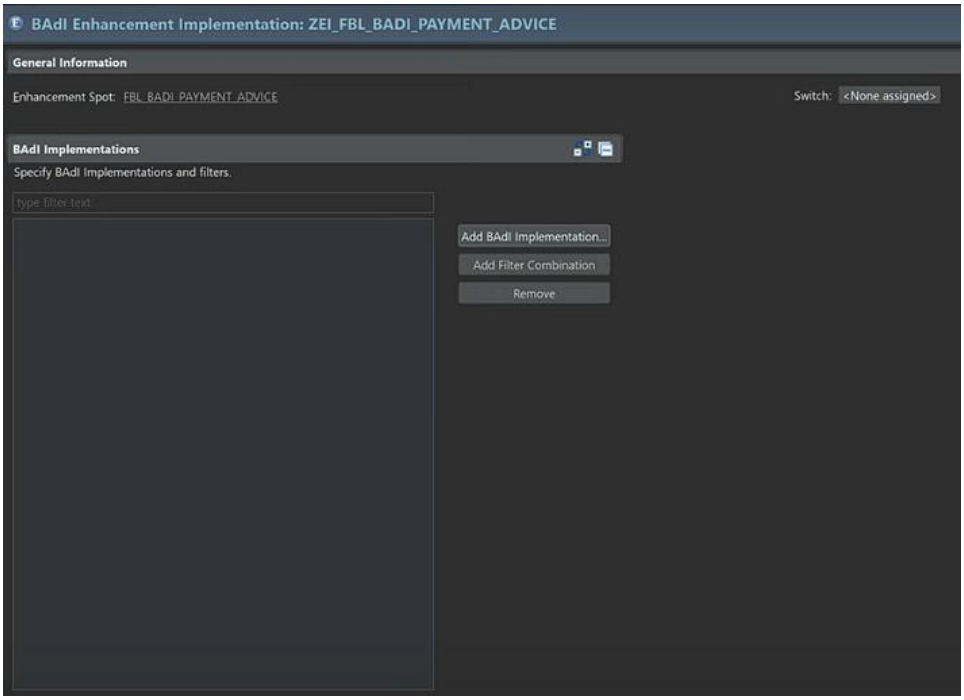
Y seleccionar la opción “BAdI Enhancement Implementation”.



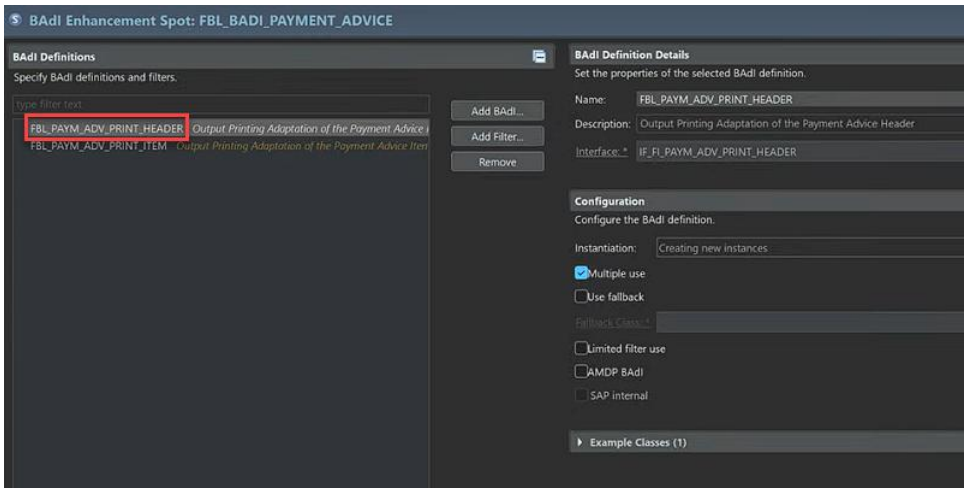
Se especifica el nombre del enhancement spot y se especifica un nombre y descripción para la ampliación de la implementación de la BAdI.



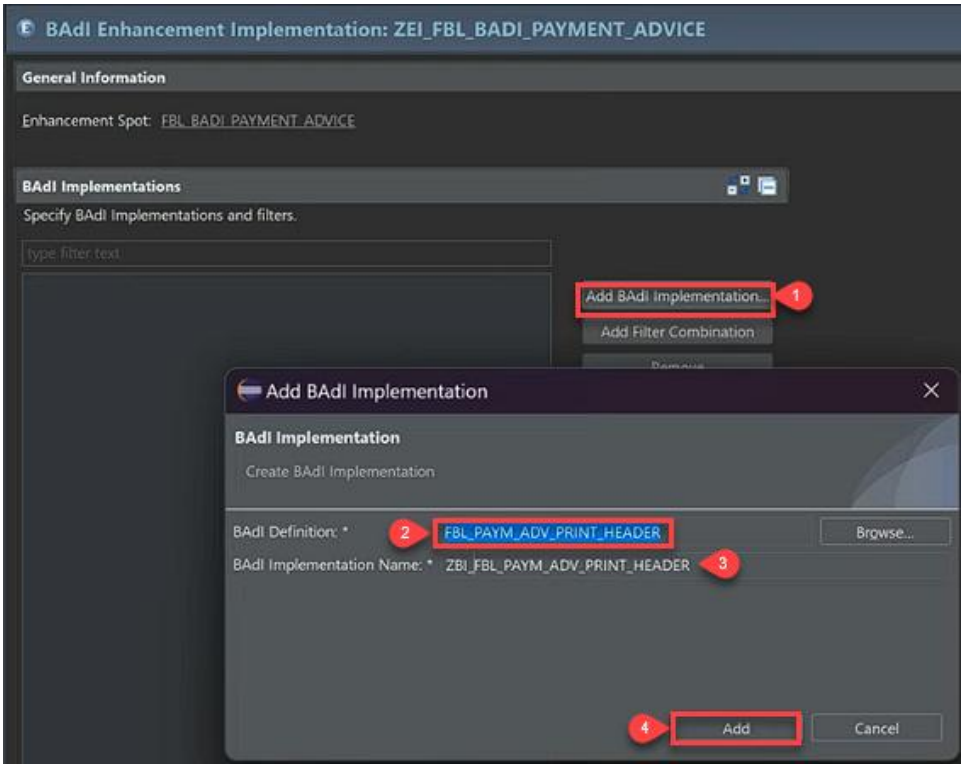
Mostrando la siguiente ventana.



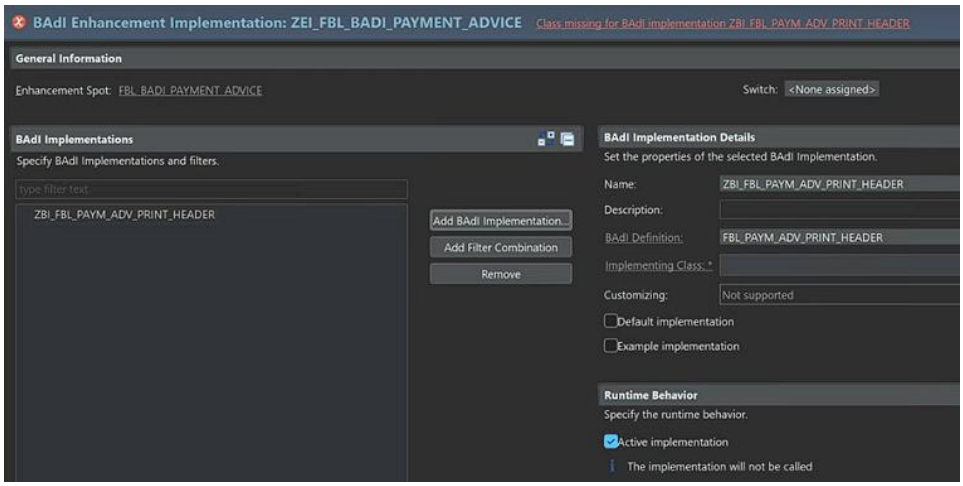
En la definición de la BAdI se puede observar que existen 2 definiciones en este caso se utilizará la primera “FBL_PAYM_ADV_PRINT_HEADER”.



Y en la ampliación se agrega presionando el botón “Add BAdI Implementation”. Para luego crear un nombre para su implementación.



Y guardar los cambios para posteriormente continuar con la creación de clase de implementación y la activación final para completar la integración de la lógica personalizada.

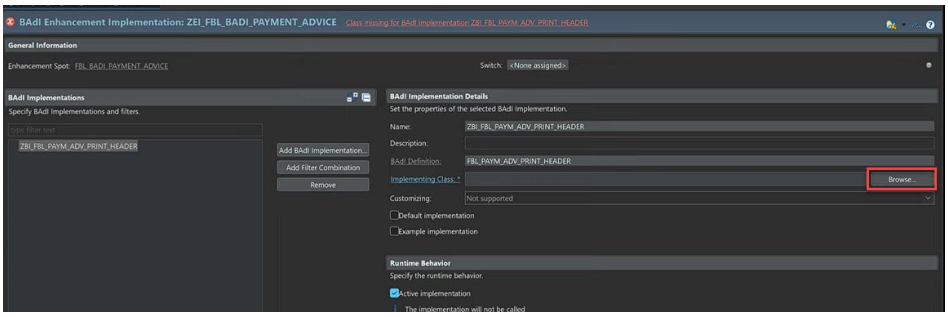


Al refrescar los cambios en el paquete se puede ver la ampliación de la implementación de la BAdI.

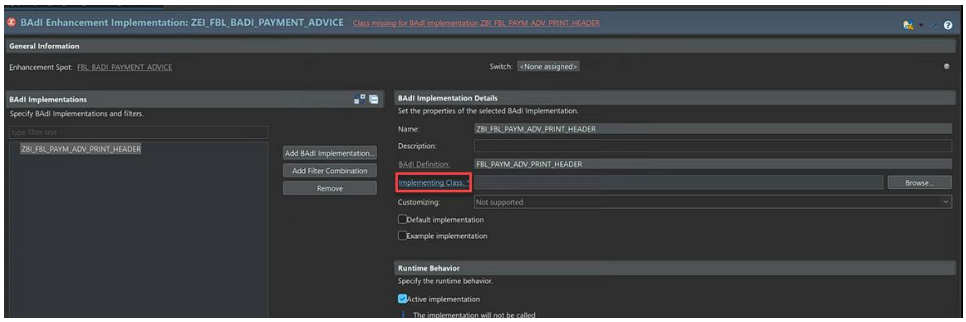
1.4. Clase de Implementación

La Clase de Implementación en SAP es un componente crucial en el proceso de personalización del sistema SAP mediante el uso de Business Add-Ins (BAdIs). Este proceso permite integrar lógica personalizada en puntos específicos del sistema estándar de SAP, conocidos como enhancement spots.

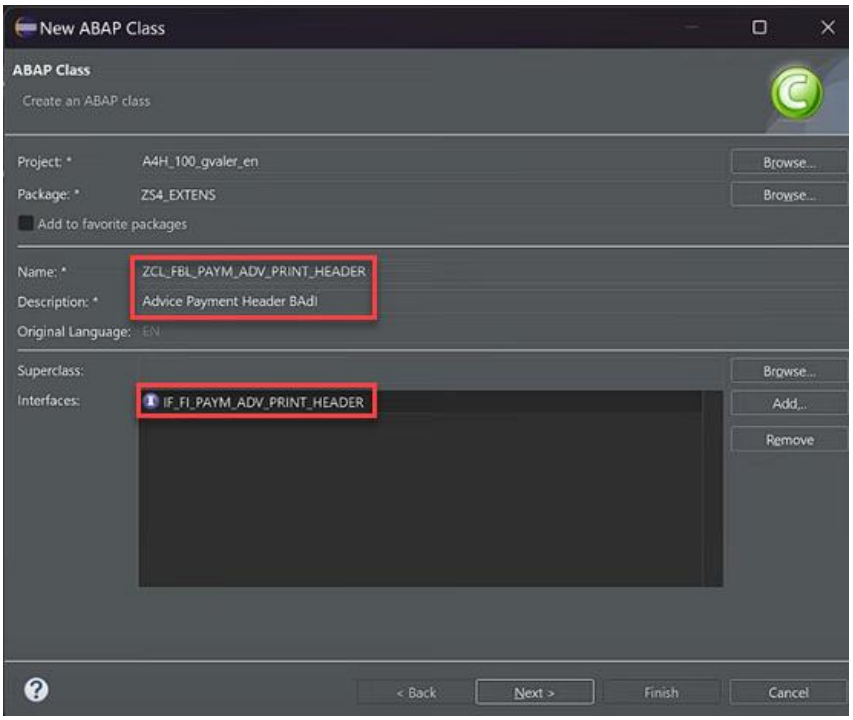
El proceso comienza desde la ampliación de la implementación de la BAdI creada anteriormente para crear o asignar una clase de implementación. En donde al presionar el botón “Browse”, se puede asignar una clase de implementación que se encuentre disponible en el sistema.



De no ser el caso al presionar el campo “Implementation Class” se abrirá una ventana para realizar la creación de la clase de implementación.

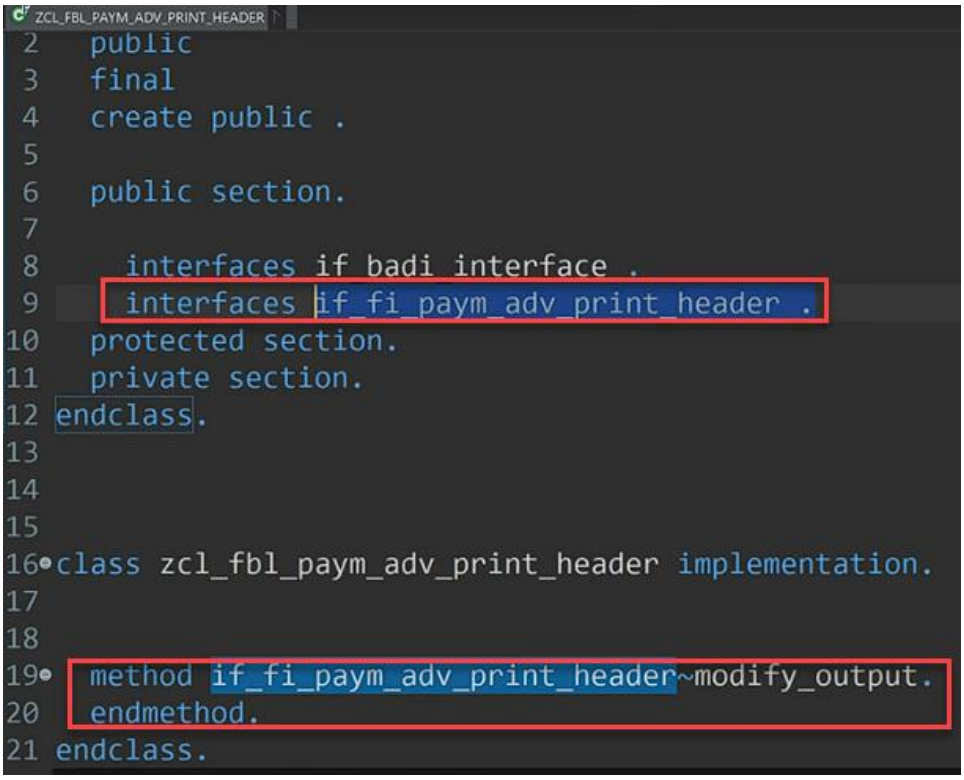


Lo interesante de esta opción es que estará seleccionada la interfaz necesaria para la creación de la clase de implementación. Por lo que solo será necesario colocar un nombre para la clase y una descripción para luego asignar los objetos creados a una orden de transporte.

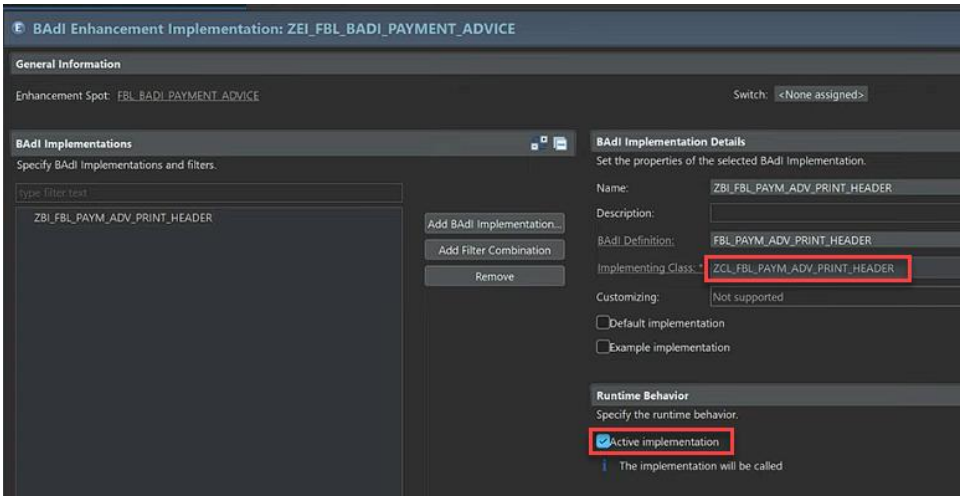


Se puede evidenciar que las interfaces if_badi_interace y if_fi_paym_adv_print_header. Han sido implementadas correctamente

por lo que la clase se puede activar para su vinculación con la ampliación de la implementación de la BAdI.



Al regresar a la implementación se puede ver que ha sido vinculada correctamente por lo que solo faltaría su activación. Otro punto importante es que se encuentra seleccionado el check “Active Implementation”, ya que esta opción asegura que la implementación sea llamada durante la ejecución del proceso estándar.



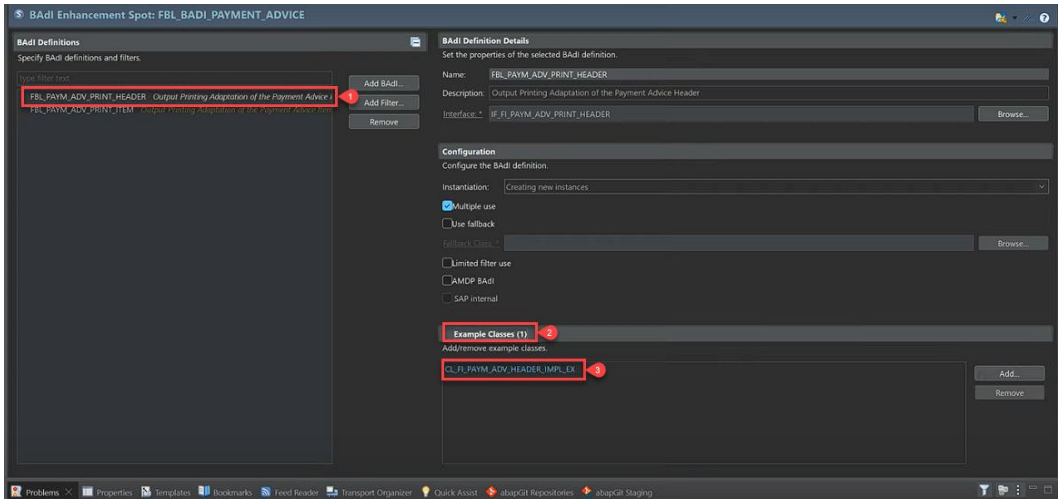
El proceso implica la creación y configuración de una clase de implementación que contiene la lógica de negocio personalizada. Esta clase se asocia a un enhancement spot y se activa para que pueda ser

llamada durante la ejecución del proceso estándar de SAP. Es esencial que esta clase implemente la interfaz correspondiente del BAdI para asegurar su correcta integración y funcionamiento.

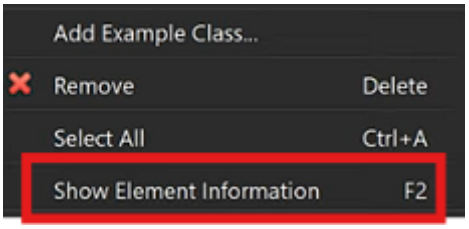
1.5. Modelo Lógico de Implementación

Se refiere al proceso de complementar y personalizar la funcionalidad estándar de las aplicaciones SAP mediante el uso de Business Add-Ins (BAdIs). Este proceso incluye la creación de una implementación de enhancement (enhancement implementation) y la definición de una clase de implementación que contenga el código necesario para realizar las modificaciones deseadas. La clase de implementación permite insertar lógica personalizada sin alterar el código fuente estándar. El proceso también abarca la activación y verificación de la implementación para asegurar que funcione correctamente durante la ejecución del sistema.

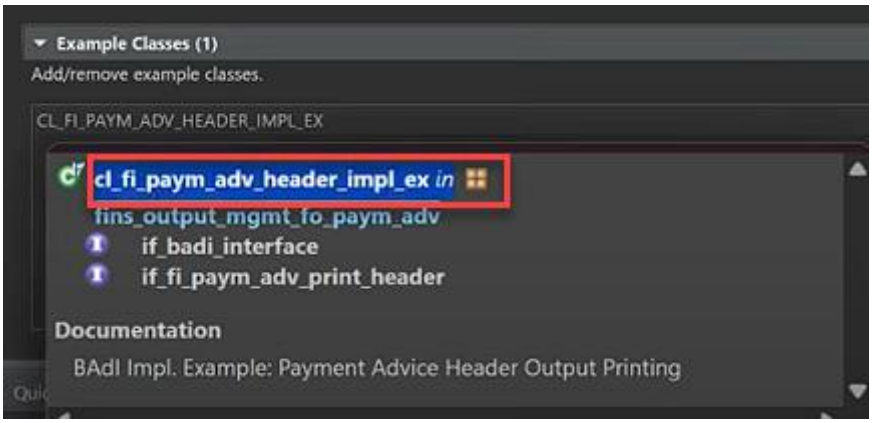
Nuevamente desde la ampliación de la implementación de la BAdI creada anteriormente, se puede seleccionar alguna de las definiciones para habilitar la sección “Example Classes”.



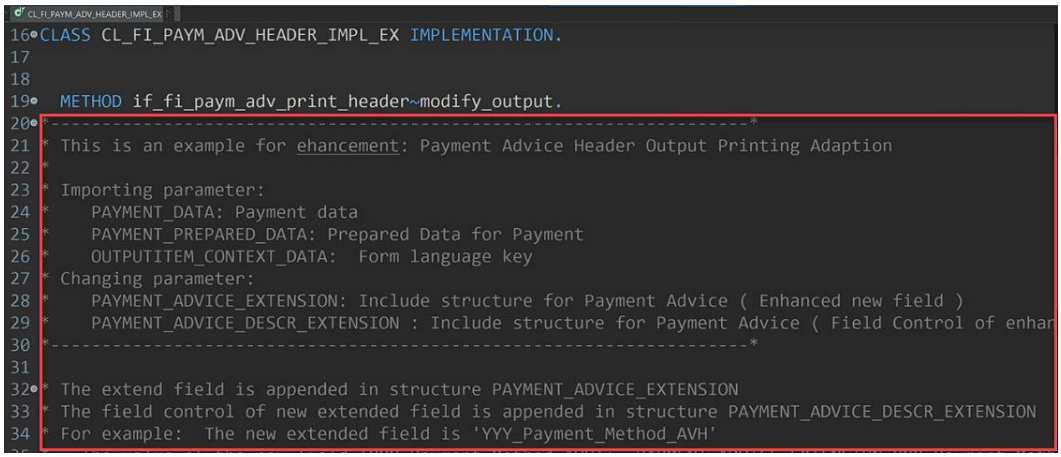
En donde es posible al hacer clic derecho en el nombre de la case de ejemplo para mostrar un el menú de opciones.



Y al seleccionar la opción “Show Element Information”, se puede visualizar la documentación de la clase y también es posible copiar el nombre.

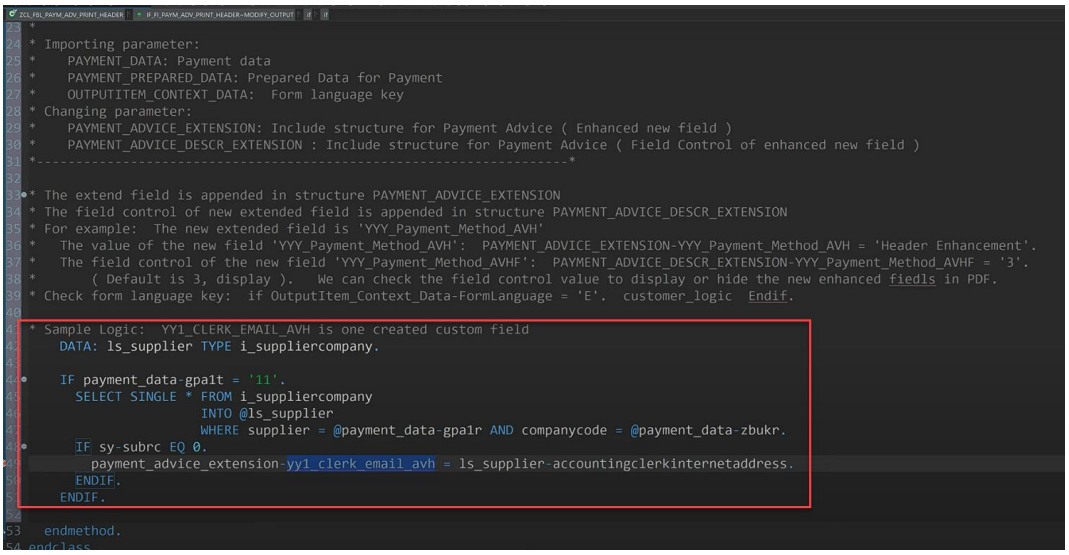


Para abrirla por medio del ABAP Development Object. En donde en primera instancia el código de ejemplo se encuentra comentado.



```
16 CLASS CL_FI_PAYM_ADV_HEADER_IMPL_EX IMPLEMENTATION.
17
18
19   METHOD if_fi_paym_adv_print_header~modify_output.
20 *---------------------------------------------------------------------*
21 * This is an example for ehancement: Payment Advice Header Output Printing Adaption
22 *
23 * Importing parameter:
24 *    PAYMENT_DATA: Payment data
25 *    PAYMENT_PREPARED_DATA: Prepared Data for Payment
26 *    OUTPUTITEM_CONTEXT_DATA:  Form language key
27 * Changing parameter:
28 *    PAYMENT_ADVICE_EXTENSION: Include structure for Payment Advice ( Enhanced new field )
29 *    PAYMENT_ADVICE_DESCR_EXTENSION : Include structure for Payment Advice ( Field Control of enhan
30 *---------------------------------------------------------------------*
31
32 * The extend field is appended in structure PAYMENT_ADVICE_EXTENSION
33 * The field control of new extended field is appended in structure PAYMENT_ADVICE_DESCR_EXTENSION
34 * For example:  The new extended field is 'YYY_Payment_Method_AVH'
```

El código de ejemplo se puede copiar y utilizar en la clase de implementación de la ampliación de la BAdI, adaptándolo según las necesidades específicas del negocio. Esto implica comprender y modificar el código de ejemplo, lo cual puede incluir la eliminación de comentarios, la realización de consultas (SELECT) en entidades como Core Data Services (CDS) y la incorporación de campos personalizados (custom fields).



```
ZCL_FBL_PAYM_ADV_PRINT_HEADER  IF_FI_PAYM_ADV_PRINT_HEADER~MODIFY_OUTPUT  if  if
23  *
24  * Importing parameter:
25  *    PAYMENT_DATA: Payment data
26  *    PAYMENT_PREPARED_DATA: Prepared Data for Payment
27  *    OUTPUTITEM_CONTEXT_DATA:  Form language key
28  * Changing parameter:
29  *    PAYMENT_ADVICE_EXTENSION: Include structure for Payment Advice ( Enhanced new field )
30  *    PAYMENT_ADVICE_DESCR_EXTENSION : Include structure for Payment Advice ( Field Control of enhanced new field )
31  *----------------------------------------------------------------------*
32
33  * The extend field is appended in structure PAYMENT_ADVICE_EXTENSION
34  * The field control of new extended field is appended in structure PAYMENT_ADVICE_DESCR_EXTENSION
35  * For example:  The new extended field is 'YYY_Payment_Method_AVH'
36  *   The value of the new field 'YYY_Payment_Method_AVH':  PAYMENT_ADVICE_EXTENSION-YYY_Payment_Method_AVH = 'Header Enhancement'.
37  *   The field control of the new field 'YYY_Payment_Method_AVHF':  PAYMENT_ADVICE_DESCR_EXTENSION-YYY_Payment_Method_AVHF = '3'.
38  *       ( Default is 3, display ).   We can check the field control value to display or hide the new enhanced fiedls in PDF.
39  * Check form language key:  if OutputItem_Context_Data-FormLanguage = 'E'.  customer_logic  Endif.
40
41  * Sample Logic:  YY1_CLERK_EMAIL_AVH is one created custom field
42    DATA: ls_supplier TYPE i_suppliercompany.
43
44    IF payment_data-gpa1t = '11'.
45      SELECT SINGLE * FROM i_suppliercompany
46                    INTO @ls_supplier
47                    WHERE supplier = @payment_data-gpa1r AND companycode = @payment_data-zbukr.
48      IF sy-subrc EQ 0.
49        payment_advice_extension-yy1_clerk_email_avh = ls_supplier-accountingclerkinternetaddress.
50      ENDIF.
51    ENDIF.
52
53  endmethod.
54 endclass
```

El modelo de lógica de implementación en SAP mediante BAdIs es un proceso técnico detallado que permite personalizar y extender la funcionalidad estándar del sistema. Desde la creación del enhancement implementation y la clase de implementación, pasando por la activación y verificación, hasta la depuración y pruebas, cada paso es esencial para asegurar una integración efectiva y sin problemas. El uso de ejemplos proporcionados por SAP facilita este proceso y proporciona una base clara para el desarrollo adicional.