LOGALI

Teoría

CDS – Business Object

ABAP RESTful – Arquitectura Cloud

Contenido

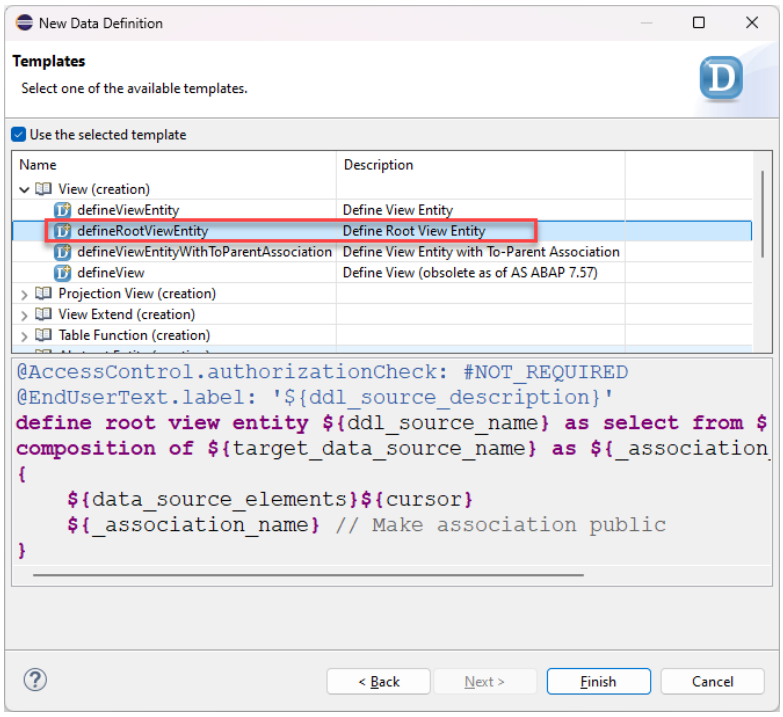
1. CDS – Business Object

1.1. Root Entity

La creación de entidades raíz (Root Entity) en SAP RAP (Restful ABAP Programming) es un paso esencial en el desarrollo de objetos de negocio (Business Object). Utilizando Core Data Services (CDS), se define una entidad con una estructura coherente que integra anotaciones, composiciones, y asociaciones necesarias para la gestión eficiente de los datos y facilita su interacción tanto para los usuarios finales como para los endpoints API.

Esta entidad se crea a partir de una entidad de persistencia y actúa como el nodo principal en la jerarquía de datos y es el punto de entrada para todas las relaciones de negocio en una aplicación RAP. La entidad raíz facilita la gestión centralizada de datos complejos, estableciendo relaciones de composición y jerarquía con otras entidades del sistema. Su correcta implementación asegura la integridad y eficiencia en la manipulación y acceso a los datos del negocio.

Para crear una entidad raíz desde cero se hace el mismo procedimiento para crear una vista CDS que se realiza a través de la carpeta de proyecto luego New en la opción **Other ABAP Repository Object** ubicar la carpeta **Business Services** y luego **Data Definition**. Con la diferencia que la plantilla que debe ser seleccionada es la plantilla **defineRootViewEntity** que se encuentra en la carpeta **View (creation).**



Tomando en cuenta que es recomendable utilizar la nomenclatura **z_r_entity_name**. Para identificar que será una entidad raíz.

Sintaxis:

```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS Root Entity'
define root view entity entity_name
 as select from data_source_name
 //composition of target_data_source_name as _AssociationName
 association [min..max] to entity_name as _AliasName on
_AliasName.ComponentName = $projection.ComponentName
…
{
 _AssociationName // Make association public
}
```

Puntos importantes a considerar para la implementación:

- Es necesario especificar como tabla de fuente de datos la tabla de persistencia.
- Luego insertar todos los componentes de dicha tabla.
- Agregar un alias a todos los componentes utilizando el formato **UpperCamelCase,** para seguir con la nomenclatura OData.
- Agregar las anotaciones pertinentes a los campos de auditoría. Todo esto para tener en la aplicación, después de haber definido los demás artefactos una vinculación automática de los valores que necesitan estos campos de auditoría. Dichas anotaciones son las siguientes:
  - **@Semantics.user.createdBy: true:** Esta anotación indica que el campo anotado contiene el ID del usuario que creó el registro de datos.
  - **@Semantics.systemDateTime.createdAt: true:** Esta anotación indica que el campo anotado contiene la fecha y hora en que se creó el registro de datos.
  - **@Semantics.user.localInstanceLastChangedBy: true:** Esta anotación indica que el campo anotado contiene el ID del usuario que hizo el último cambio en el registro de datos a nivel local.
- Agregar las anotaciones ETag pertinentes a los campos de auditoría para manejar el control de versiones y la verificación de datos entre servidores y clientes. Para los siguientes conceptos:
  - **Local ETag:** es un campo que se utiliza para implementar un control de concurrencia optimista verificando los cambios a nivel local antes de que se guarden los datos. Este campo se actualiza cada vez que se modifica una instancia de la entidad raíz. Su propósito principal es asegurar que las modificaciones locales no entren en conflicto con cambios realizados por otros usuarios o procesos mientras se trabaja en una instancia de datos. La anotación utilizada para este concepto es la siguiente:
    - **@Semantics.systemDateTime.localInstanceLast ChangedAt: true:** Esta anotación indica que el campo anotado contiene la fecha y hora en que se

realizó el último cambio en el registro de datos a nivel local.

- **Total ETag:** Es un campo que se utiliza para implementar un control de concurrencia optimista verificando los cambios a nivel global antes de que se guarden los datos. Este campo se actualiza cada vez que se modifica cualquier instancia activa de la entidad raíz. Su propósito es asegurar que las modificaciones realizadas en una instancia activa no entren en conflicto con cambios realizados en una instancia de datos en estado de borrador (draft) antes de que se convierta en activa. La anotación utilizada para este concepto es la siguiente:
  - **@Semantics.systemDateTime.lastChangedAt: true:** Esta anotación indica que el campo anotado contiene la fecha y hora en que se realizó el último cambio en el registro de datos.

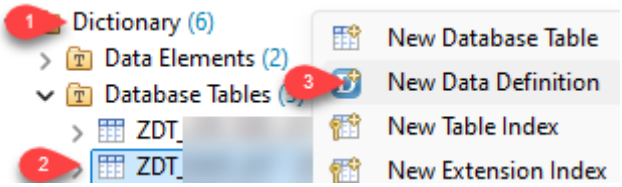
- Es necesario agregar las asociaciones necesarias para vincular la tabla raíz con otras entidades.
- En caso de agregar asociaciones a otras entidades:
  - Se debe especificar la cardinalidad. Aunque por defecto, se asigna una cardinalidad de cero a uno **[0..1]** en caso de no especificarla. Los diferentes tipos de cardinalidad son los siguientes:
    - **[1..1]:** Cada registro en la entidad origen está relacionado con exactamente un registro en la entidad destino.
    - **[0..1]:** Cada registro en la entidad origen puede estar relacionado con cero o un registro en la entidad destino.
    - **[1..*]:** Cada registro en la entidad origen está relacionado con uno o más registros en la entidad destino.
  - Publicar dichas asociaciones dentro de los componentes de la entidad para que se habilite el acceso a los componentes de las entidades relacionadas.

Ejemplo:

```
@AccessControl.authorizationCheck: #NOT_REQUIRED
```

```
@EndUserText.label: 'CDS Root Entity'
define root view entity entity_name
 as select from data_source_name
 //composition of target_data_source_name as _AssociationName
 association [min..max] to entity_name as _AliasName on
_AliasName.ComponentName = $projection.ComponentName
…
{
// components from database_source_name with alias
 components,
 // audit components
@Semantics.user.createdBy: true
local_created_by      as LocalCreatedBy,
@Semantics.systemDateTime.createdAt
local_created_at      as LocalCreatedAt,
@Semantics.user.localInstanceLastChangedBy
local_last_changed_by as LocalLastChangedBy,
@Semantics.systemDateTime.localInstanceLastChangedAt: true
local_last_changed_at as LocalLastChangedAt,
@Semantics.systemDateTime.lastChangedAt: true
last_changed_at       as LastChangedAt,
 // Publication of associations
 _AssociationName // Make association public
}
```

Sin embargo, existe otra forma de crear la entidad raíz, que permite utilizar como base de consulta una tabla de base de datos, reduciendo así el tiempo dedicado al mapeo de los componentes de la entidad. Para esto, es necesario ubicar la carpeta **Dictionary** dentro del proyecto y buscar la tabla de base de datos requerida dentro de la carpeta **Database Tables**. Una vez localizada la tabla de persistencia, se hace clic derecho y se selecciona la opción **New Data Definition** y seleccionar la plantilla **defineRootViewEntity** que se encuentra en la carpeta **View (creation).**

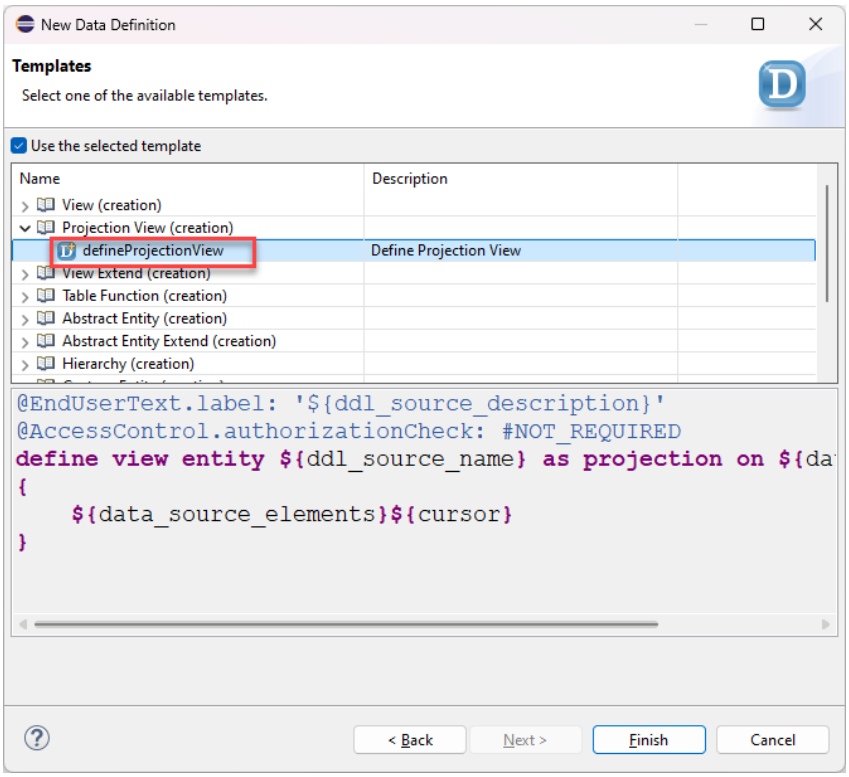


1.2. Interface Entity

Una entidad de interfaz (Interface Entity) o también conocida como proyección de contrato transaccional interfaz define una vista que expone solo las columnas relevantes para una interfaz transaccional específica. Esta técnica facilita la separación entre la lógica de negocio y la presentación de datos, lo que permite la creación de APIs y aplicaciones con acceso controlado a los datos.

Esta entidad permite que los desarrolladores amplíen y mejoren los objetos de negocio sin alterar la lógica subyacente. Basada en la entidad raíz (Root Entity), la entidad de interfaz integra anotaciones y proyecciones necesarias para gestionar datos de manera eficiente y coherente. Aunque no es necesario su implementación en un diseño RAP personalizado, su comprensión es clave para aprovechar las capacidades de extensibilidad ofrecidas por SAP RAP.

Para crear una proyección se realiza el mismo procedimiento para crear una entidad o vista CDS a través de la carpeta de proyecto luego New en la opción **Other ABAP Repository Object** ubicar la carpeta **Core Data Services** y luego seleccionar **Data Definition** y seleccionar la plantilla **defineProjectionView** que se encuentra en la carpeta **Projection View (creation).**



Tomando en cuenta que es recomendable utilizar la nomenclatura **z_i_entity_name**. Para identificar que será una entidad de interfaz.

Puntos importantes a considerar para la implementación:

- Es necesario especificar como proyección la entidad raíz que consulta la tabla de persistencia.
- Se debe agregar la palabra reservada **root,** en la declaración de la proyección quedando de la siguiente manera **define root view entity.**
- Comentar o eliminar en el encabezado de la declaración de la proyección la anotación **@Metadata.ignorePropagatedAnnotations: true,** para habilitar la herencia con la **entidad root** padre.
- Insertar solo los componentes y las notaciones que se necesiten utilizar de la entidad raíz. Utilizando las mismas anotaciones utilizados en los componentes de la entidad Raíz.
- Agregar el tipo de contrato **transactional_interface.** Para permitir la extensibilidad en el modelo de programación de aplicaciones ABAP RESTful. A través de las palabras reservadas **provider contract transactional_interface.**

Sintaxis:

```
@EndUserText.label: 'CDS Interface Projection'
@AccessControl.authorizationCheck: #NOT_REQUIRED
define root view entity entity_name
 provider contract transactional_interface
 as projection on data_source_name
{
// components from database_source_name with alias
 components,
 // audit components
@Semantics.systemDateTime.localInstanceLastChangedAt: true
local_last_changed_at as LocalLastChangedAt,
@Semantics.systemDateTime.lastChangedAt: true
last_changed_at      as LastChangedAt,
 // Publication of associations
 _AssociationName // Make association public
 }
```

Sin embargo, la forma más fácil de crear la entidad interfaz o proyección de contrato transaccional interfaz, es utilizando una entidad raíz que consulta la tabla de persistencia como base de la estructura de sus componentes. Es necesario ubicar la carpeta **Core Data Service** dentro del proyecto y buscar la entidad raíz requerida dentro de la carpeta **Data Definitions**. Una vez localizada la tabla de persistencia, se hace clic derecho y se selecciona la opción **Data Definition** y seleccionar la plantilla **defineProjectionView** que se encuentra en la carpeta **Projection View (creation).**

1.3. Consumption Entity

La entidad de consumo o también conocida como proyección de contrato transaccional Query, define una vista que expone solo las

columnas relevantes para una interfaz transaccional específica en el modelo de programación RAP (Restful ABAP Programming) de SAP. Y es responsable de presentar la información en la capa del consumidor, permitiendo que los datos sean accesibles a través de diversos protocolos asociados con el servicio. La proyección de entidades se basa en una entidad raíz (Root Entity), y se configura para facilitar la interacción y el consumo de datos en las aplicaciones.

Implementación:

Los pasos para la implementación de este tipo de entidad son prácticamente los mismos que para la proyección del tipo de contrato de interfaz, con la diferencia de que esta vez el tipo de contrato será query. Esto se especifica utilizando las palabras reservadas: **provider contract transactional_query.**

Además no es necesario comentar o eliminar del encabezado de la entidad la anotación **@Metadata.ignorePropagatedAnnotations: true**

Tomando en cuenta que es recomendable utilizar la nomenclatura **z_c_entity_name**. Para identificar que será una entidad de consumo.

Sintaxis:

```
@EndUserText.label: 'CDS Interface Projection'
@AccessControl.authorizationCheck: #NOT_REQUIRED
@Metadata.ignorePropagatedAnnotations: true
define root view entity entity_name
 provider contract transactional_query
 as projection on data_source_name
{
// components from database_source_name with alias
 components,
 // audit components
@Semantics.systemDateTime.localInstanceLastChangedAt: true
local_last_changed_at as LocalLastChangedAt,
@Semantics.systemDateTime.lastChangedAt: true
last_changed_at      as LastChangedAt,
 // Publication of associations
 _AssociationName // Make association public
```

```
}
```

1.4. Object Model – Text Element

Object Model es una anotación que se utiliza para definir y gestionar diversas propiedades avanzadas dentro de un modelo de datos. Esta anotación permite especificar aspectos cruciales como la calidad del servicio, la categoría y tamaño de los datos, claves representativas, entre otras capacidades adicionales que optimizan la búsqueda y el manejo de información en la base de datos.

Proyección de Textos:

La proyección de textos por medio de la anotación Object Model en la entidad de consumo tiene como objetivo vincular los componentes proyectados y sus descripciones de la entidad raíz. Estas descripciones están disponibles en las asociaciones proyectadas en la entidad raíz para ser utilizadas en la interfaz de usuario. Este enfoque evita la necesidad de mostrar columnas separadas para códigos y textos, proporcionando una representación más limpia y comprensible.

Sintaxis

```
@ObjectModel.text.element: [ 'AliasAssociationCompName' ]
component,
_AssociationName.Component  as AliasAssociationCompName,
```

Se pueden vincular múltiples textos a un componente dentro de la misma anotación agregando cada uno de los alias de la asociación separados por una coma.

Ejemplo:

```
@ObjectModel.text.element: [ 'AliasAssociationCompName',
'AliasAssociationCompName2' ]
    component,
```

```
  _AssociationName.Component  as AliasAssociationCompName,
  _AssociationName.Component2  as AliasAssociationCompName2,
```

Ejemplo de la implementación:

```
@EndUserText.label: 'CDS Interface Projection'
@AccessControl.authorizationCheck: #NOT_REQUIRED
@Metadata.ignorePropagatedAnnotations: true
define root view entity entity_name
 provider contract transactional_query
 as projection on data_source_name
{
// components from database_source_name with alias
  @ObjectModel.text.element: [ 'AliasAssociationCompName' ]
  component,
  _AssociationName.Component  as AliasAssociationCompName,
 // audit components
@Semantics.systemDateTime.localInstanceLastChangedAt: true
local_last_changed_at as LocalLastChangedAt,
@Semantics.systemDateTime.lastChangedAt: true
last_changed_at      as LastChangedAt,
 // Publication of associations
 _AssociationName // Make association public
 }
```

Limitaciones en la Capa de Proyección:

- No todas las transformaciones de datos son posibles en la capa de proyección. Las operaciones complejas, como **concat** y **concat_with_space**, deben realizarse en la entidad raíz.
- La proyección debe centrarse en la representación visual y la vinculación de textos con códigos, mientras que las transformaciones de datos se manejan en capas anteriores.

1.5. Localized

La localización en SAP RAP (Restful ABAP Programming) es un proceso esencial que permite presentar los textos asociados a los datos en el idioma adecuado para el usuario final. Utilizando la anotación **localized**, los desarrolladores pueden gestionar

eficientemente la cardinalidad de los componentes de las proyecciones de consumo, asegurando que los textos se muestran en el idioma del usuario. Esta funcionalidad es clave para aplicaciones multilingües, ya que mejora el uso y la experiencia del usuario al presentar información contextualizada y accesible en su idioma preferido.

Implementación de la anotación:

Esta implementación se realiza agregando al menos un componente en la entidad de consumo o proyección de contrato transaccional Query. Con el objetivo de vincular los componentes de la proyección con las descripciones o textos de idioma correspondiente al usuario de la entidad raíz.

Además, es necesario manejar la cardinalidad de las asociaciones para asegurar que la proyección no devuelva un conjunto de resultados incorrectos.

Sintaxis:

```
_AssociationName._AssociationName2[1: Language = $session.system_language].component as AliasComponentName,
```

Ejemplo de implementación:

```
@EndUserText.label: 'CDS Interface Projection'
@AccessControl.authorizationCheck: #NOT_REQUIRED
@Metadata.ignorePropagatedAnnotations: true
define root view entity entity_name
 provider contract transactional_query
 as projection on data_source_name
{
// components from database_source_name with alias
  component,
   @ObjectModel.text.element: [ 'AliasAssociationCompName' ]
  component,
  _AssociationName.Component  as AliasAssociationCompName,
```

```
_AssociationName._AssociationName2.component          as
AliasComponentName : localized,

 // audit components
@Semantics.systemDateTime.localInstanceLastChangedAt: true
local_last_changed_at as LocalLastChangedAt,
@Semantics.systemDateTime.lastChangedAt: true
last_changed_at      as LastChangedAt,
 // Publication of associations
 _AssociationName // Make association public
}
```

Existe una variante agregando la instrucción **localized.** Permitiendo que el sistema inyecte automáticamente la cardinalidad y el valor del campo de texto del idioma del usuario en la proyección, reduciendo la cardinalidad y mejorando la eficiencia del código. Ya que el sistema identifica la entidad con los campos de texto de idioma que se requiere vincular. Estos textos están disponibles en algunas de las asociaciones proyectadas de la entidad raíz y, en ciertos casos, los textos de idioma pueden estar a su vez en una asociación dentro de la asociación vinculada a la entidad raíz. Estos campos de textos de idioma se identifican dentro de una entidad en los componentes de lenguaje y textos que utilicen la anotación **@Semantics.language: true** y **@Semantics.text: true,** respectivamente dentro de la misma entidad asociada. Y al ser vinculados en la entidad de consumo dichos textos pueden ser utilizados en la interfaz de usuario.

Sintaxis:

```
_AssociationName._AssociationName2.component          as
AliasComponentName : localized,
```

Ejemplo de entidad con campos de texto de idioma:

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Overall Status Value Help'
@Metadata.ignorePropagatedAnnotations: true
@ObjectModel.usageType:{
```

```
serviceQuality: #A,
sizeCategory: #S,
dataClass: #MASTER
}
@ObjectModel.resultSet.sizeCategory: #XS
define view entity /DMO/I_Overall_Status_VH_Text
 as select from /dmo/oall_stat_t
 association [1..1] to /DMO/I_Overall_Status_VH as _OverallStatus
on $projection.OverallStatus = _OverallStatus.OverallStatus
{
    @ObjectModel.text.element: ['Text']
 key overall_status as OverallStatus,
    @Semantics.language: true
 key language      as Language,
    @Semantics.text: true
    text          as Text,
    _OverallStatus
}
```

1.6. Capacidad de búsqueda avanzada

La búsqueda avanzada en SAP HANA es una herramienta poderosa que mejora significativamente la capacidad de los usuarios para encontrar información relevante de manera rápida y precisa. A través de la configuración de anotaciones específicas y la optimización de umbrales de búsqueda borrosa, los desarrolladores pueden proporcionar una experiencia de usuario más eficiente y agradable. Esta funcionalidad es esencial para aplicaciones que manejan grandes volúmenes de datos y requieren búsquedas detalladas y accesibles.

Implementación de la capacidad de búsquedas avanzadas:

Para habilitar dichas búsquedas avanzadas es necesario configurar la entidad de consumo a nivel de cabecera con la anotación:

```
@Search.searchable: true
```

Para designar un campo específico como el elemento de búsqueda predeterminado. Es necesario identificar dichos componentes con la anotación.

Sintaxis:

@Search.defaultSearchElement: true

Al identificar una columna para el elemento de búsqueda o crear una columna técnica adicional. Manejará los tokens o fragmentos de texto para los criterios de búsqueda, optimizando el rendimiento de la base de datos. Esta columna facilita la búsqueda por diferentes variaciones de texto, asegurando que los resultados sean relevantes y precisos.

Búsqueda Borrosa (Fuzzy Search):

Otro concepto importante en las búsquedas avanzadas es la anotación de Búsqueda Borrosa (Fuzzy Search), que permite encontrar resultados que son similares al valor buscado, incluso si no coinciden exactamente. Configurando el nivel de tolerancia a través del **umbral de búsqueda**, el cual varía entre el rango de valores del 0.0 a 1.0, donde 1.0 representa una coincidencia exacta y valores más bajos permiten mayor tolerancia a errores. Este umbral ajustable permite personalizar la tolerancia del sistema a errores tipográficos o variaciones en la búsqueda.

Se recomienda establecer el umbral de búsqueda borrosa en alrededor de 0.8 para obtener un equilibrio entre resultados precisos y una mayor cantidad de coincidencias relevantes.

Sintaxis:

@Search.fuzzinessThreshold: decimal_number

Otro aspecto importante a mencionar es que es posible establecer un **orden de relevancia de los campos dentro de una búsqueda avanzada esto** se realiza a través de la anotación:

@Search.ranking: #Option

Al asignar un valor de ranking a un elemento de búsqueda, se determina qué tan importante es ese campo en comparación con otros cuando se devuelven los resultados de la búsqueda. Donde las posibles opciones de configuración son las siguientes:

- **#HIGHT:** El elemento es de alta relevancia; esto suele ser válido para los identificadores y sus descripciones.
- **#LOW:** Aunque el elemento es relevante para la búsqueda de estilo libre, un resultado en este elemento no tiene importancia real para la clasificación de un elemento del resultado.
- **#MEDIUM:** El elemento tiene una relevancia media; esto suele suceder con otros elementos importantes. Esta es la opción predeterminada.

Ejemplo de implementación:

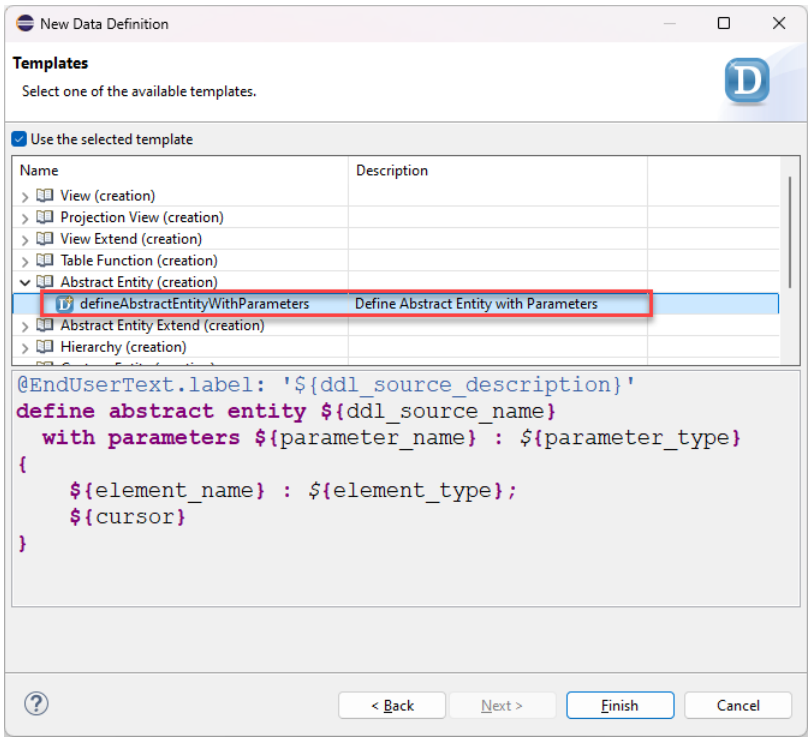
```
@EndUserText.label: 'CDS Interface Projection'
@AccessControl.authorizationCheck: #NOT_REQUIRED
@Metadata.ignorePropagatedAnnotations: true
@Search.searchable: true
define root view entity entity_name
 provider contract transactional_query
 as projection on data_source_name
{
// components from database_source_name with alias
  @Search.defaultSearchElement: true
  @Search.ranking: #Option
  @Search.fuzzinessThreshold: decimal_number
  component,
  @Search.defaultSearchElement: true
  @Search.ranking: #Option
  @Search.fuzzinessThreshold: decimal_number
  @ObjectModel.text.element: [ 'AliasAssociationCompName' ]
  component,
  _AssociationName.Component  as AliasAssociationCompName,
_AssociationName._AssociationName2.component                as
AliasComponentName : localized,

 // audit components
```

```
@Semantics.systemDateTime.localInstanceLastChangedAt: true
local_last_changed_at as LocalLastChangedAt,
@Semantics.systemDateTime.lastChangedAt: true
last_changed_at      as LastChangedAt,
 // Publication of associations
 _AssociationName // Make association public
 }
```

1.7. Abstract Entity

Una entidad abstracta en SAP RAP (Restful ABAP Programming) es una estructura utilizada para definir componentes sin seleccionar ni proyectar datos de otras fuentes. Estas entidades son similares a las estructuras en el diccionario de datos y se usan principalmente para definir elementos que se utilizarán en la interfaz de usuario para solicitar y manejar datos del usuario. Las entidades abstractas son esenciales para crear acciones en aplicaciones, como botones o formularios, donde se requiere la entrada de múltiples datos por parte del usuario.

Para crear una entidad abstracta se hace el mismo procedimiento para crear una vista CDS se realiza a través de la carpeta de proyecto luego New en la opción **Other ABAP Repository Object** ubicar la carpeta **Business Services** y luego **Data Definition**. Con la diferencia que la plantilla que debe ser seleccionada es la plantilla **defineAbstractEntityWithParameters** que se encuentra en la carpeta **Abstract Entity (creation).**



Tomando en cuenta que es recomendable utilizar la nomenclatura **z_ae_entity_name**. Para identificar que será una entidad abstracta.

Sintaxis:

```
@EndUserText.label: 'Abstract Entity'
define abstract entity abstract_entity_name
{
  @EndUserText.label: ‘Label’
  element_name : element_type;
…
}
```

También es posible agregar una etiqueta para el nombre del elemento utilizando la anotación **@EndUserText.label: ‘’**