

estabelecer a conjunção de um campo com as suas unidades de texto descritivo independente da lingua. Também definimos o *value help*₂₀ para os campos Agency ID, Customer ID, and Carrier ID usando a anotação *@Consumption*, e redirecionamos os nós de composition da viagem e da reserva para as entidades correspondentes da camada de projeção do business object. Com as projections views, podemos expôr apenas aqueles elementos que são relevantes para o serviço específico, incluindo a desnormalização do modelo de dados subjacente. O *value help*, a pesquisa e a semântica da UI também podem ser definidos.

A seguir, temos que especificar a view da projeção do BO como a entidade root, e fornecer um alias para a projection view da viagem que já está atribuída. Todas as views, elementos e associações são inseridos automaticamente na lista de projeção.

Nesta fase, vamos adicionar algumas semânticas específicas do serviço ao modelo de dados projetado, porque permite que a projection view seja aprimorada com extensões de metadados separadas e porque permite a pesquisa usando as anotações *@Metadata.allowExtensions* e *@Search.searchable*.

Relativamente à lista de projeções, a anotação *@Search.DefaultSearchElement* ativa a pesquisa das colunas TravelID, AgencyID e CustomerID. A anotação *@ObjectModel.text.element* especifica os elementos adicionados anteriormente AgencyName e CustomerName como descrições para os elementos AgencyID e CustomerID, respectivamente. A anotação *@Consumption.ValueHelpDefinition* define o value help para os elementos de visualização AgencyID, CustomerID e CurrencyCode. Aqui, o nome da entidade CDS que atua como um value help, é o nome do elemento que está vinculado ao elemento local que tem que ser especificado. É possível completar automaticamente ao especificar a entidade do value help de destino.

Vamos também especificar o elemento currency como o campo de referência para os campos de currency BookingFee e TotalPrice, usando o elemento annotation *@Semantics.amount.currency*. Manteremos todas as associações expostas para caso seja necessário usar, mas redirecionaremos o BO “filho” da entidade booking para a visualização de projeção do BO Booking. A anotação *@ObjectModel.text.element* serve para fornecer uma descrição aos elementos CustomerId e CarrierId. Para o CustomerID, o CarrierID, o ConnectionID e o CurrencyCode os *value help* são definidos de formas diferentes. Para o *value help* do ConnectionID temos que adicionar uma condição de ligação extra, que é definida para retornar valores do *value help* que já foram guardados para os elementos de visualização locais CarrierID, FlightDate, FlightPrice e Currency (linha 23 do booking). O campo de visualização CurrencyCode é especificado como o campo de referência para o campo de currency FlightPrice (linha 31 booking).

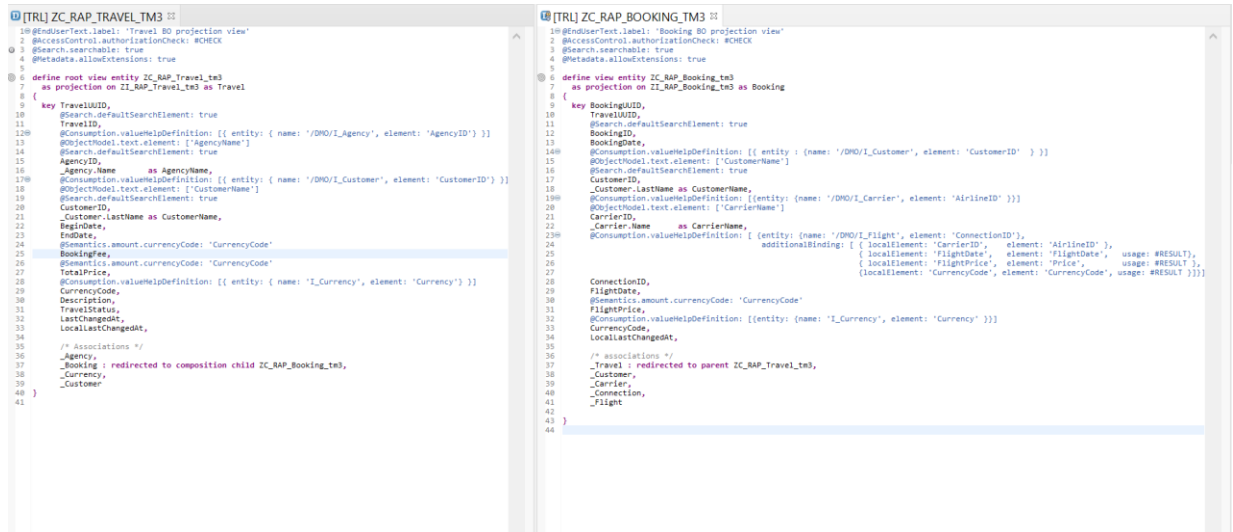


Figura 11: Projection views.

De modo a enriquecer ainda mais a projeção do modelo de dados CDS com a semântica da UI vamos usar extensões de metadados CDS (MDEs), (figura 12). As extensões dos metadados do CDS MDEs permitem que se melhore sem alterações no modelos de dados. Com o CDS, podemos adicionar semânticas para diferentes domínios para uma entidade CDS existente usando anotações, normalmente, anotações de UI. Ao usar CDS MDEs, conseguimos uma separação do modelo de dados com todos os business objects da semântica da UI. Essa separação de interesses permite uma melhor gestão de mudança simplificada sem modificação. Graças à abordagem por causa das camadas de CDS MDEs, podemos definir mais de uma extensão de metadados CDS para uma entidade CDS. Ficamos assim com duas extensões de metadados CDS: um MDE CDS para a view de viagem projeção e outro para a projeção de reserva.

Cada extensão de metadados é atribuída a uma camada, como *CORE#*, *PARTNER#* ou *CUSTOMER#*. A camada determina a prioridade da avaliação. Então com a *view* diversificada e os campos das anotações da UI específicos, podemos ver o nome da entidade CDS que está enriquecida. Nesta altura, temos a anotação *@Metadata.layer*, o nome da projection view do BO após a instrução *ANNOTATE VIEW* e uma entrada fictícia entre as chaves. Devemos especificar o *#CORE* como a camada de metadados. Quando existem várias extensões de metadados definidas para uma determinada entidade CDS, a layer determina a prioridade dos metadados. *#CORE* tem a prioridade mais baixa e *#CUSTOMER* a prioridade mais alta. Em cima devemos definir também algumas informações do cabeçalho, como o nome do tipo e o título. Neste caso, os dados de viagem apresentados serão classificados em ordem decrescente pelo elemento *TravelID* na lista, por isso, nas chaves, estamos a usar *@UI.facet* anotações para definir a pesquisa para a página do objeto e do seu layout. A página de objetos de viagem tem duas facetas: a referência de identificação das entidades de viagens e a referência do item da linha da entidade do booking, com a *composition_booking* especificada como elemento de destino.

A seguir, devemos especificar uma posição e possivelmente um rótulo para cada elemento usando as respectivas anotações de elemento *@UI*. O *@UI.lineItem* é a anotação usada para especificar as informações de layout de cada elemento mostrado como uma coluna na lista. A anotação *@UI.identification* é usada para especificar as informações de layout de cada elemento mostrado na secção de identificação da página do objeto. E a anotação *@UI.selectionField* é usada para permitir que

um elemento seja selecionado na funcionalidade filtro. Com a anotação `@UI.hidden: true`, evitamos que os elementos sejam exibidos na UI ou nas configurações. Os elementos `TravelID`, `AgencyID` e `CustomerID` estão disponíveis para serem selecionado nos filtro e também mostrados na lista de viagens.

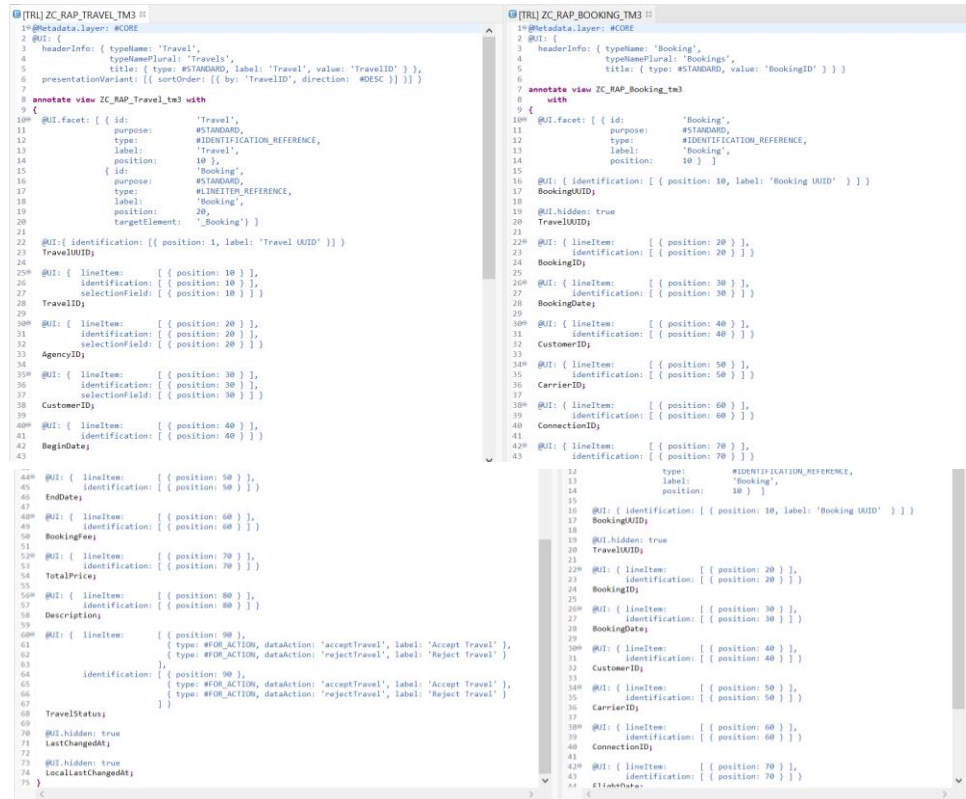


Figura 12: Metadados do CDS.

5º: Serviço Odata

O próximo passo é criar o serviço Odata e visualizar a aplicação. Passos a seguir: criar uma *service definition*₈ (figura 13) para especificar o scope do serviço *Odata*, e de seguida criar uma *service binding*₉ para ligar o *service definition* ao protocolo *OData* como um serviço de UI. O scope do serviço é definido expondo o que é relevante: cds views e os metadados.

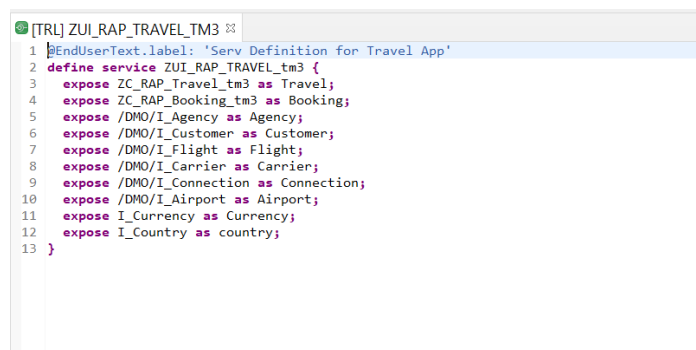


Figura 13: Service Definition.

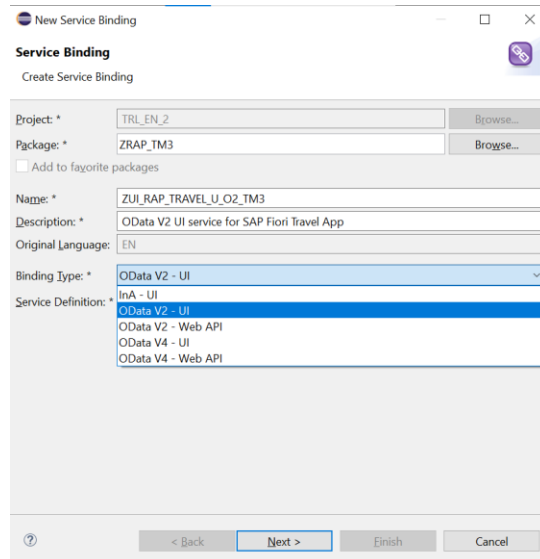


Figura 14: Criação do Service Binding. Atualmente apenas OData V2 é compatível, no entanto, OData V4 está planejado para ser usado.

No *service binding* (figura 15) temos o service URL, os conjuntos de entidades expostas e as associações. Se clicarmos no link do service URL podemos ver os metadados do serviço no browser.

Para podermos ver a aplicação selecionamos a “Travel” e clicamos duas vezes ou clicamos com o botão direito, e aparece a opção abrir com Fiori Elements app.

A aplicação é aberta no browser e podemos ver as colunas de pesquisa: Agency ID e Customer ID que estão de acordo com as anotações de pesquisa definidas na projection view do BO de viagem. Podemos filtrar como por exemplo, pelo ID da agência, podemos usar o *value help*, podemos personalizar a lista adicionando, movendo e removendo colunas através da personalização das configurações. Por exemplo, podemos adicionar o *UUID* e movê-lo para a primeira posição. E podemos também facilmente restaurar o layout padrão.

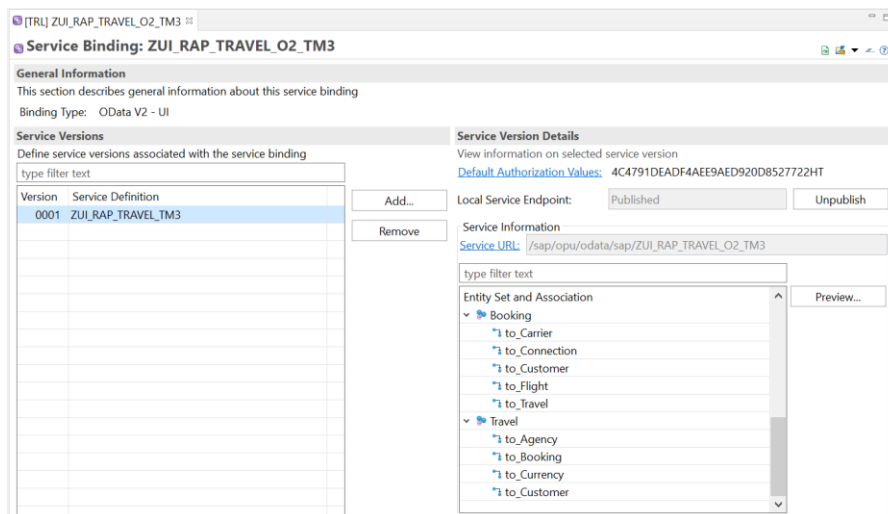


Figura 15: Service Binding.

6º : Implementação das autorizações básicas

Para finalizar, vamos implementar uma autorização básica para dados de acesso usando funções CDS para o business objects de viagens. O *ABAP CDS* fornece uma linguagem de controle de dados, DCL, para definir o acesso de autorização aos dados usando uma função CDS. A autorização clássica usada na plataforma ABAP requer verificações explícitas de autoridade codificada nos dados selecionados, que são armazenados em tabelas internas na camada de aplicação. O CDS DCL, em vez disso, oferece uma alternativa declarativa com base em verificações de autorização implícitas que ocorrem durante as tentativas de acesso à entidade CDS. As funções do CDS podem ser definidas usando condições PFCG, condições literais, condições do utilizador e condições de herança. Devem ser definidas explicitamente para cada entidade CDS, quando necessário. Fazem parte do modelo de dados e, portanto, também são enviadas para a base de dados para que apenas os dados autorizados sejam devolvidos ao ler um modelo de dados CDS.

As funções do CDS são definidas uma vez e usadas automaticamente em todo o lado. Ao aceder às entidades CDS, teremos definido um authorization object e duas funções de CDS para a entidade de viagem, uma na camada de modelação de dados e outra na camada de provisionamento de business service para a projeção do modelo de dados. A função do CDS na camada de modelação de dados será definida com condições literais e PFCG, e criaremos um objeto de autorização para o efeito. O *PFCG₂₈* é um transaction code cujo objetivo é manter administração para manage roles e authorization data. A segunda função CDS irá herdar a condição da camada subjacente. As regras de acesso consistirão numa condição literal para o elemento de visualização *CurrencyCode* e uma condição PFCG para o elemento de visualização *TravelStatus*. Para a definição do PFCG, criaremos um authorization object, incluindo campo de autorização e elemento de dados.

The screenshot shows the SAP Data Element configuration interface for 'ZOSTAT_TM3'. The 'Data Type Information' tab is active, showing the following fields:

- Category: Domain
- Type Name: DMO/OVERALL_STATUS
- Data Type: CHAR
- Length: 1

The 'Field Labels' tab is also visible, showing the following field labels and their maximum lengths:

Field Label	Maximum Length
Short: Status	10
Medium: Travel Status	20
Long: Travel Status	40
Heading: Travel Status	55

Figura 16: Criação do Data Element.

No campo Category temos que pôr “Domain” e inserir o nome do tipo no campo Type name. Nos Filed labels como mostra o screenshot, devemos preencher com “Status” e “Travel Status” (figura 17).

Authorization Field: ZOSTAT_TM3

General

Data Element: * ZOSTAT_TM3

Provide Search Help in Standard Maintenance Dialog

Check Table:

Used in authorization object

type filter text

Class	Object	Description
CPAE	ZOSTAT_TM3	Authorization object for travel status

What's next?

[Create a new Authorization Object and assign the Authorization Field to it](#)
[Assign the Authorization Field to an existing Authorization Object](#)

Figura 17: Authorization object.

Neste screenshot estamos a permitir adicionar, criar, alterar, fazer display e apagar (figura 18).

Authorization Object: ZOSTAT_TM3

General

Object Class: CPAE Object Class Description: SAP Cloud Platform ABAP Environment Objects

Authorization Fields

type filter text

Authorization Field	Description	Activity Field
ZOSTAT_TM3	Travel status	<input type="checkbox"/>
ACTVT	Activity	<input checked="" type="checkbox"/>
<Enter new value>		<input type="checkbox"/>

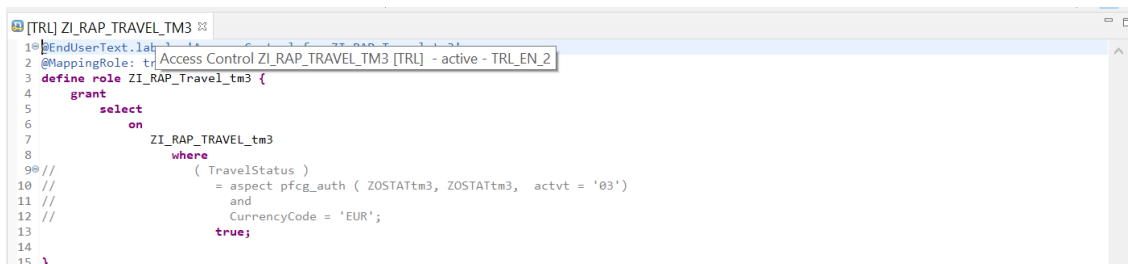
Permitted Activities

type filter text

Activity	Description	Access Category
01	Add or Create	Write
02	Change	Write
03	Display	Read
06	Delete	Write
<Enter new value>		

Figura 18: Authorization object, permissões das atividades criar, altrear, fazer display e apagar.

No screenshot da figura 19 estamos a definir o papel do CDS para a view do BO da viagem. Assim, vamos criar um novo controle de acesso. A anotação @MappingRole: true é definida no topo para atribuir a função CDS a todos os utilizadores, independentemente do cliente. O nome do CDS protegido é especificado após o select na declaração e as regras de acesso do utilizador são definidas na cláusula where. Em comentário está definido uma condição no elemento CurrencyCode. Apenas registos com o código de moeda EURO devem ser seleccionados. Está em comentário porque nós queremos ter o acesso total aos dados e para isso podemos adicionar uma condição ou “true” na cláusula where do controle de acesso para o BO de viagem.



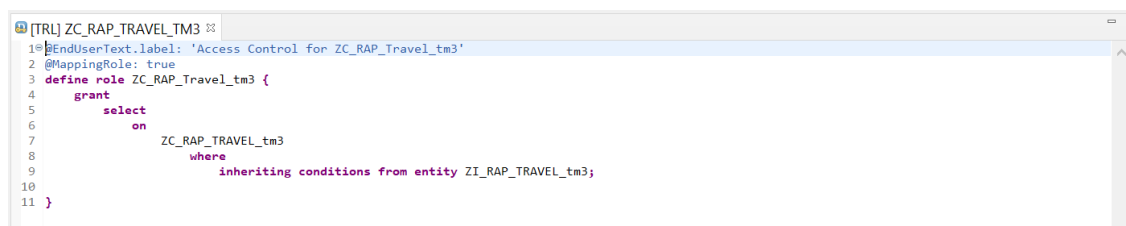
```

1 @MappingRole: true
2 @MappingRole: true
3 define role ZI_RAP_Travel_tm3 {
4   grant
5     select
6       on
7         ZI_RAP_TRAVEL_tm3
8         where
9           // ( TravelStatus )
10          // = aspect pfcg_auth ( ZOSTATtm3, ZOSTATtm3, actvt = '03' )
11          // and
12          // CurrencyCode = 'EUR';
13          true;
14 }
15

```

Figura 19: Definição do papel do CDS para a view do BO viagem.

Conforme já explicado, uma função CDS deve ser definida explicitamente para cada entidade CDS. Não há herança implícita de regras de acesso. Portanto, vamos agora definir as regras de acesso para a visão de projeção do BO de viagem (figura 20). A anotação @MappingRole: true é definida na parte superior. O nome do CDS é especificado após a instrução grant select on, e, na cláusula da condição where, podemos definir a entidade CDS da qual as condições devem ser herdadas.



```

1 @MappingRole: true
2 @MappingRole: true
3 define role ZC_RAP_Travel_tm3 {
4   grant
5     select
6       on
7         ZC_RAP_TRAVEL_tm3
8         where
9           inheriting conditions from entity ZI_RAP_TRAVEL_tm3;
10 }
11

```

Figura 20: Definição das regras de acesso para a visão de projeção do BO viagem.

A próxima fase é ativar os recursos transacionais da nossa aplicação, criando o *behavior definition* do business object: neste teremos a definição básica do comportamento, que define o nosso business object das viagens *managed₂₁* com o criar, atualizar e apagar, bem como as associações necessárias. É muito importante que o nome do *behavior definition* tenha o mesmo nome da root view do CDS. Neste anexo apenas estou a implementar a aplicação do tipo managed.

A primeira coisa que definimos é o alias (figura 21) para a viagem e a entidade de reserva. Depois, especificamos a persistência dando os nomes das tabelas da base de dados para ambas as entidades. Isso permite o runtime managed de execução para executar as operações criar, alterar e apagar diretamente nas nossas tabelas de base de dados. O próximo passo é especificar o master lock para a entidade root. Para isso, adicionamos a linha 8 "*lock master₂₂*". O nó “filho” da reserva torna-se dependente do lock e faz uso da associação definida na CDS view para permitir ao transacional fazer a associação de viagens explicitamente listadas na entidade de reserva. Como este é um cenário baseado em UUID, queremos que o runtime managed forneça uma chave quando novas instâncias são criadas. Para isso, precisamos de

fazer a numeração TravelUUID managed e somente de leitura. O mesmo é necessário para a entidade de reserva.

```

@TRL ZI_RAP_TRAVEL_TM3
1 managed;
2 with draft;
3
4@define behavior for ZI_RAP_Travel_tm3 alias Travel
5 implementation in class zbp_i_rap_travel_tm3 unique
6 persistent table zrap_atrav_tm3
7 draft table zrap_dtrav_tm3
8 lock master total etag LastChangedAt
9 authorization master ( instance )
10 etag master LocalLastChangedAt
11 {
12   create;
13   update;
14   delete;
15   association _Booking { create; with draft; }
16
17   field ( numbering : managed, readonly ) TravelUUID;
18   field ( readonly ) TravelID, TotalPrice, TravelStatus;
19   field ( readonly ) LastChangedAt, LastChangedBy, CreatedAt, CreatedBy, LocalLastChangedAt;
20   field ( mandatory ) AgencyID, CustomerID;
21
22   action ( features : instance ) acceptTravel result [1] $self;
23   action ( features : instance ) rejectTravel result [1] $self;
24   internal action recalcTotalPrice;
25
26   determination setInitialStatus on modify { create; }
27   determination calculateTotalPrice on modify { field BookingFee, CurrencyCode; }
28   determination calculateTravelID on save { create; }
29
30   validation validateAgency on save { field AgencyID; create; }
31   validation validateCustomer on save { field CustomerID; create; }
32   validation validateDates on save { field BeginDate, EndDate; create; }
33
34   draft determine action Prepare {
35     validation validateAgency;
36     validation validateCustomer;
37     validation validateDates;
38   }
39
40@ mapping for zrap_atrav_tm3
41 {
42   TravelUUID = travel_uuid;
43   TravelID = travel_id;
44   AgencyID = agency_id;
45   CustomerID = customer_id;
46   BeginDate = begin_date;
47   EndDate = end_date;
48   BookingFee = booking_fee;
49   TotalPrice = total_price;
50   CurrencyCode = currency_code;
51   Description = description;
52   TravelStatus = overall_status;
53   CreatedBy = created_by;
54   CreatedAt = created_at;
55   LastChangedBy = last_changed_by;
56   LastChangedAt = last_changed_at;
57   LocalLastChangedAt = local_last_changed_at;
58 }
59 }
60
61@define behavior for ZI_RAP_Booking_tm3 alias Booking
62 implementation in class zbp_i_rap_booking_tm3 unique
63 persistent table zrap_abook_tm3
64 draft table zrap_dbook_tm3
65 lock dependent by _Travel
66 authorization dependent by _Travel
67 etag master LocalLastChangedAt
68 {
69   update;
70   delete;
71
72   association _Travel { with draft; }
73
74   field ( numbering : managed, readonly ) BookingUUID;
75   field ( readonly ) TravelUUID, BookingID;
76   field ( readonly ) CreatedBy, LastChangedBy, LocalLastChangedAt;
77
78   determination calculateBookingID on modify { create; }
79   determination calculateTotalPrice on modify { field FlightPrice, CurrencyCode; }
80
81@ mapping for zrap_abook_tm3
82 {
83   BookingUUID = booking_uuid;
84   TravelUUID = travel_uuid;
85   BookingID = booking_id;
86   BookingDate = booking_date;
87   CustomerID = customer_id;
88   CarrierID = carrier_id;
89   ConnectionID = connection_id;
90   FlightDate = flight_date;
91   FlightPrice = flight_price;
92   CurrencyCode = currency_code;
93   CreatedBy = created_by;
94   LastChangedBy = last_changed_by;
95   LocalLastChangedAt = local_last_changed_at;
96 }
97 }

```

Figura 21: Ativar dos recursos transacionais.

Vai aparecer um aviso indicando que o TravelUUID na entidade de reserva deve ser definido para somente leitura, pois é usado na condição de associação. A solução para o problema é definir o master *Etag*₂₃ em ambas as entidades. O outro aviso que aparece diz respeito às informações de mapeamento que faltam. Como fornecemos aliases nas visualizações de CDS da interface para os nomes dos elementos, precisamos de dizer ao framework como mapear os nomes dos elementos no modelo de dados CDS para os campos da tabela correspondente. O mesmo precisa de ser feito para a entidade de reserva.