...

```
64 draft table zrap_dbook_tm3
```

...

```
72   association _Travel { with draft; }
```

Listagem 3: Funcionalidade draft em ABAP

2º - Depois cria-se a tabela *draft*, ou seja a tabela que vai ter as viagens e as reservas como rascunho. Esta tabela nova criada é referida no código acima como `draft table zrap_dtrav_tm` para a viagem e `draft table zrap_dbook_tm` para as reservas.

3º - A seguir, o *draft* ainda não tem nenhuma projeção por isso é necessário criar na classe *ZC_RAP_TRAVEL_tm:*

...

```
2 use draft;
```

...

```
10   use association _Booking { create; with draft; }
```

...

```
23   use association _Travel { with draft; }
```

Listagem 4: Funcionalidade draft em ABAP, projection.

4º - Finalmente na classe principal *ZBP_I_RAP_TRAVEL* no método `get_authorizations`, aqui a funcionalidade adiciona duas ações predefinidas de preparação e de edição da funcionalidade, ao *business object* (primeiro retângulo) e é o que permite tornar mais um controle de autorização.

```
METHOD get_authorizations.
  DATA: has_before_image    TYPE abap_bool,
        is_update_requested TYPE abap_bool,
        is_delete_requested TYPE abap_bool,
        update_granted      TYPE abap_bool,
        delete_granted      TYPE abap_bool.

  DATA: failed_travel LIKE LINE OF failed-travel.

  " Read the existing travels
  READ ENTITIES OF zi_rap_travel_tm3 IN LOCAL MODE
    ENTITY Travel
      FIELDS ( TravelStatus ) WITH CORRESPONDING #( keys )
    RESULT DATA(travels)
    FAILED failed.

  CHECK travels IS NOT INITIAL.

*   In this example the authorization is defined based on the Activity + Travel Status
*   For the Travel Status we need the before-image from the database. We perform this for active (is_draft=00)
*as well as for drafts (is_draft=01) as we can't distinguish between edit or new drafts
  SELECT FROM zrap_atrav_tm3
    FIELDS travel_uuid,overall_status
    FOR ALL ENTRIES IN @travels
    WHERE travel_uuid EQ @travels-TravelUUID
    ORDER BY PRIMARY KEY
    INTO TABLE @DATA(travels_before_image).

  is_update_requested = COND #( WHEN requested_authorizations-%update              = if_abap_behv=>mk-on OR
                                     requested_authorizations-%action-acceptTravel = if_abap_behv=>mk-on OR
                                     requested_authorizations-%action-rejectTravel = if_abap_behv=>mk-on OR
                                     requested_authorizations-%action-Prepare      = if_abap_behv=>mk-on OR
                                     requested_authorizations-%action-Edit         = if_abap_behv=>mk-on OR
                                     requested_authorizations-%assoc-_Booking      = if_abap_behv=>mk-on
                                THEN abap_true ELSE abap_false ).

  is_delete_requested = COND #( WHEN requested_authorizations-%delete = if_abap_behv=>mk-on
                                THEN abap_true ELSE abap_false ).
```

```
  LOOP AT travels INTO DATA(travel).
    update_granted = delete_granted = abap_false.

    READ TABLE travels_before_image INTO DATA(travel_before_image)
     WITH KEY travel_uuid = travel-TravelUUID BINARY SEARCH.
    has_before_image = COND #( WHEN sy-subrc = 0 THEN abap_true ELSE abap_false ).

    IF is_update_requested = abap_true.
      " Edit of an existing record -> check update authorization
      IF has_before_image = abap_true.
        update_granted = is_update_granted( has_before_image = has_before_image  overall_status = travel_before_image-overall_status ).
        IF update_granted = abap_false.
          APPEND VALUE #( %tky        = travel-%tky
                          %msg        = NEW zcm_rap_tm3( severity = if_abap_behv_message=>severity-error
                                                         textid   = zcm_rap_tm3=>unauthorized )
                        ) TO reported-travel.
        ENDIF.
        " Creation of a new record -> check create authorization
      ELSE.
        update_granted = is_create_granted( ).
        IF update_granted = abap_false.
          APPEND VALUE #( %tky        = travel-%tky
                          %msg        = NEW zcm_rap_tm3( severity = if_abap_behv_message=>severity-error
                                                         textid   = zcm_rap_tm3=>unauthorized )
                        ) TO reported-travel.
        ENDIF.
      ENDIF.
    ENDIF.

    IF is_delete_requested = abap_true.
      delete_granted = is_delete_granted( has_before_image = has_before_image  overall_status = travel_before_image-overall_status ).
      IF delete_granted = abap_false.
        APPEND VALUE #( %tky        = travel-%tky
                        %msg        = NEW zcm_rap_tm3( severity = if_abap_behv_message=>severity-error
                                                       textid   = zcm_rap_tm3=>unauthorized )
                      ) TO reported-travel.
      ENDIF.
    ENDIF.
```

```
387
388       IF is_delete_requested = abap_true.
389         delete_granted = is_delete_granted( has_before_image = has_before_image  overall_status = travel_before_image-overall_status ).
390         IF delete_granted = abap_false.
391           APPEND VALUE #( %tky        = travel-%tky
392                           %msg        = NEW zcm_rap_tm3( severity = if_abap_behv_message=>severity-error
393                                                          textid   = zcm_rap_tm3=>unauthorized )
394                         ) TO reported-travel.
395         ENDIF.
396       ENDIF.
397
398       APPEND VALUE #( %tky = travel-%tky
399
400                       %update              = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
401                       %action-acceptTravel = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
402                       %action-rejectTravel = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
403                       %action-Prepare      = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
404                       %action-Edit         = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
405                       %assoc-_Booking      = COND #( WHEN update_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
406
407                       %delete              = COND #( WHEN delete_granted = abap_true THEN if_abap_behv=>auth-allowed ELSE if_abap_behv=>auth-unauthorized )
408                     )
409         TO result.
410     ENDLOOP.
411
412   ENDMETHOD.
413
```

Listagem 5: Funcionalidade draft em ABAP, código.

Conclusão

Neste anexo foi feita uma análise prática de como se faz uma aplicação SAP ABAP RESTfull programming model. Relativamente à configuração do ambiente foi utilizada uma licença trial com a duração de um mês, tendo-se verificado que para projectos longos o sistema pode levar à perda do projecto desenvolvido, pois não é possível guardar noutro meio. Na criação das tabelas utilizaram-se dados de demonstração o que para um exemplo de aplicação facilita a criação das mesmas.

O modelo de dados core data services (cds) é o que permite definir a estrutura da aplicação, pois é a base da mesma, preparando-a para os próximos passos, como é o caso do Odata. Na definição do Odata faz-se a ligação entre o eclipse e a SAP Fiori Web.

Para finalizar, na classe de implementação dos métodos observam-se as implementações das funcionalidades na aplicação, que é o que corresponde ao *back-end*. A ABAP RESTfull programming model é uma solução que tem muito potencial e a característica que sobressai é a simplicidade relativamente ao APEX, e que apesar de ainda ser recente tem bons pontos a adotar. É uma solução que para usufruir de todas as suas potencialidades requer já alguns conhecimentos da linguagem ABAP.

Anexos


```
managed;
with draft;

define behavior for ZI_RAP_Travel_tm3 alias Travel
implementation in class zbp_i_rap_travel_tm3 unique
persistent table zrap_atrav_tm3
draft table zrap_dtrav_tm3
lock master total etag LastChangedAt
authorization master ( instance )
etag master LocalLastChangedAt
{
  create;
  update;
  delete;
  association _Booking { create; with draft; }

  field ( numbering : managed, readonly ) TravelUUID;
  field ( readonly  ) TravelId, TotalPrice, TravelStatus;
  field ( readonly  ) LastChangedAt, LastChangedBy, CreatedAt, CreatedBy, LocalLastChangedAt;
  field ( mandatory ) AgencyID, CustomerID;

  action ( features : instance ) acceptTravel result [1] $self;
  action ( features : instance ) rejectTravel result [1] $self;
  internal action recalcTotalPrice;

  determination setInitialStatus    on modify { create; }
  determination calculateTotalPrice on modify { field BookingFee, CurrencyCode; }
  determination calculateTravelID   on save   { create; }

  validation validateAgency   on save { field AgencyID; create; }
  validation validateCustomer on save { field CustomerID; create; }
  validation validateDates    on save { field BeginDate, EndDate; create; }

  draft determine action Prepare  {
    validation validateAgency;
    validation validateCustomer;
    validation validateDates;
  }

  mapping for zrap_atrav_tm3
  {
    TravelUUID         = travel_uuid;
    TravelID           = travel_id;
    AgencyID           = agency_id;
    CustomerID         = customer_id;
    BeginDate          = begin_date;
    EndDate            = end_date;
    BookingFee         = booking_fee;
    TotalPrice         = total_price;
    CurrencyCode       = currency_code;
    Description        = description;
    TravelStatus       = overall_status;
    CreatedBy          = created_by;
    CreatedAt          = created_at;
    LastChangedBy      = last_changed_by;
    LastChangedAt      = last_changed_at;
    LocalLastChangedAt = local_last_changed_at;
  }
}

define behavior for ZI_RAP_Booking_tm3 alias Booking
implementation in class zbp_i_rap_booking_tm3 unique
persistent table zrap_abook_tm3
draft table zrap_dbook_tm3
lock dependent by _Travel
authorization dependent by _Travel
etag master LocalLastChangedAt
{
  update;
  delete;

  association _Travel { with draft; }

  field ( numbering : managed, readonly ) BookingUUID;
  field ( readonly ) TravelUUID, BookingId;
  field ( readonly ) CreatedBy, LastChangedBy, LocalLastChangedAt;

```

```
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 }
```

Teresa Monteiro, 52597, MIEI FCT

02/03/2021

Glossário

Service Instances[1] - No contexto do SAP Cloud, é a definição de um cliente OAuth.

Transport request[2] - é uma espécie de *Container / Collection*[3] de mudanças que são feitas no sistema de desenvolvimento.

Container / Collection[3] - Um Container SAP é um controle que acomoda outros controles. Ele gere esses controles de forma lógica numa coleção e fornece uma área física na qual eles são exibidos.

Core Data Service[4] (CDS) – Tem o objetivo de suportar o esboço das funcionalidades, da extensibilidade e as implementações "fora da caixa" para assumir todas as implementações técnicas das tarefas.

Business Object[5] (BO) - É um termo comum para representar um artefato do mundo da vida real numa aplicação. No caso do meu estágio serão as viagens e as reservas os BO.

Query[6] - É a interface de conexão para o acesso somente de leitura à base de dados nos serviços Odata, sendo usado basicamente para relatórios de lista ou relatórios analíticos para processar os dados;

Business service[7] - É o conjunto do *service definition*[8] e do *service binding*[9].

Service definition[8] - Descreve quais as entidades do core data service de um modelo de dados que vão ser expostas para um business service específico.

Service binding[9] - É um objeto do repositório ABAP usado para ligar um service definition com o protocolo de comunicação cliente-servidor, como *Odata*[10].

Odata[10] - É um protocolo que oferece muitas funcionalidades, como por exemplo, powerful querying, e com a SAP Fiori e UI5, podendo criar Fiori applications.

Web Api[11] - É um serviço OData que é exposto como uma API da Web que vem sem nenhuma informação específica da user interface (UI) nos metadados, isto é, a Web api é a interface pública para que qualquer cliente OData possa ter acesso ao serviço OData. Na SAP fiori UI service, a cada configuração front-end que se manifesta no objeto de desenvolvimento de back-end, são expostos nos metadados do serviço. Isso significa que uma UI Fiori lê as informações nos metadados e cria a interface do utilizador correspondente para o serviço. Essas configurações da UI podem ser ampliadas e sobrescritas no SAP Web IDE.

Sap Fiori[11] - É uma linguagem de design e abordagem de experiência do utilizador desenvolvida pela SAP para ser usada pela SAP, pelos seus clientes e parceiros em aplicações.

Namespace[12] – É o espaço de nomes que pode ser usado para objetos personalizados. Existe para diferenciar entre os padrões SAP e os que são desenvolvidos por o programador.

Database Migration Option[13] (/DMO/) – Simplifica e agiliza muito toda a migração SAP com um processo, uma ferramenta e um tempo de inatividade. O DMO permite que os utilizadores SAP atualizem um sistema SAP existente para uma versão superior e/ou podem migrar para a base de dados SAP HANA.

IF_OO_ADT_RUN[14] – Na execução da aplicação no eclipse, a classe que implementa if_oo_adt_classrun envia como um parâmetro para o serviço "ADT" como uma solicitação HTTP.

View[15] - É criada combinando os dados de uma ou mais tabelas contendo informações sobre um objeto de uma aplicação e pode-se representar um subconjunto dos dados contidos numa tabela ou pode até juntar várias tabelas numa única tabela virtual.

Data Definition[16] – Define estruturas.

Projection view[17] - Fornece meios dentro de um serviço específico para definir projeções específicas do serviço.

Alias[18] - São nomes que os administradores do sistema definem para cada sistema que criam para que os componentes do portal possam fazer referência a esses sistemas. São um meio para recuperar as informações armazenadas em servidores de base de dados sem precisar saber o nome do servidor.

Semantics annotations[19] - Permitem a padronização da semântica que só tem impacto no lado do utilizador.

Value help[20] – Funcionalidade que permite encontrar a informação (dados) em falta.

Implementation managed[21] - Está associada a uma greenfield aplication, definida anteriormente, que tem o runtime managed.

Lock master[22] - O bloqueio é ativado quando o registo mestre é salvo.

Etag[23] - É usado para a função do processador, que pode ser modificada. Ao usar uma ETag mestre em todas as entidades, o processamento simultâneo é habilitado para o BO de viagem.

Key service[24] - é usada para conectar uma *instance service* à ABAP Development Tools (ADT).

Abap trial[25] – é uma parte da Cloud Foundry trial que pode ser acedido através do seguinte link https://cockpit.hanatrial.ondemand.com.

ABAP Class[26] - é a menor unidade de encapsulamento em objetos ABAP. Um método pode, portanto, usar todas as componentes de todas as instâncias da mesma classe, exceto as componentes da sua própria instância. Uma exceção a essa regra são as subclasses que não podem ter acesso aos componentes privados das superclasses, se não forem seus amigos.

Dicionário ABAP[27] - Permite centralizar a definição e a gestão de tipos como por exemplo, a criação de elementos de dados definidos pelo utilizador, estruturas e tipos de tabela, a definição de tabelas, índices e visualizações e a definição dos serviços que suportam o desenvolvimento do programa.

PFCG[28] – É um código transacioanl que administra a manutenção de funções para gerir funções e dados de autorização.

Savepoint[26] - https://docs.oracle.com/en/database/oracle/oracle-database/19/jjdbc/JDBC-standards-support.html#GUID-020AF373-6BF0-4BB3-B338-6F9609A223CA

Field level help[27] **-** https://docs.oracle.com/cd/E59116_01/doc.94/e58804/ch_locate_field_help.htm#WEABA219

https://docs.oracle.com/cd/E59116_01/doc.94/e58804/ch_locate_prog_lev_help.htm#WEABA203

Referências:

Database System Concepts, Seventh Edition, Abraham Silberschatz, Henry F. Korth, S.Sudar shan, LCCN 2018060474 | ISBN 9780078022159 (alk. paper) | ISBN 0078022150

https://apex.oracle.com/en/learn/documentation/ (02/03/2021)

https://open.sap.com/courses/cp13/overview (02/03/2021)

https://accenture.percipio.com/books/264e4dd8-1778-493d-81d7-a36b376ece9a (02/03/2021)

https://help.sap.com/doc/3750bcdf7b8045e18f1b759e6d2b000b/Cloud/en-US/ABAP_RESTful_Programming_Model_EN.pdf (02/03/2021)

https://blogs.sap.com/2019/02/08/evolution-of-the-abap-programming-model/#ClassicABAPProgramming (02/03/2021)

https://github.com/SAP-samples/abap-platform-rap-opensap (02/03/2021)

https://www.sapinsideronline.com/articles/a-developers-guide-to-the-abap-restful-programming-model/ (02/03/2021)

https://blogs.sap.com/2019/02/08/evolution-of-the-abap-programming-model/ (02/03/2021)

https://blogs.sap.com/2020/09/21/comparing-abap-restful-application-programming-rap-model-with-the-cloud-application-programming-cap-model/ (02/03/2021)

https://www.sapinsideronline.com/articles/a-developers-guide-to-the-abap-restful-programming-model/ (02/03/2021)

https://blogs.sap.com/2016/04/04/getting-started-abap-programming-model/ (02/03/2021)

https://help.sap.com/viewer/index (02/02/2021)

Teresa Monteiro, 52597, MIEI FCT

02/03/2021