***Slide 3: O que é REStful?***

O modelo de programação ABAP RESTful é o desenvolvimento de serviços RESTful OData que podem ser consumidos por aplicações SAP Fiori ou qualquer tipo de cliente OData oferecendo suporte ao desenvolvimento desses mesmo serviços OData otimizados para SAP HANA para aplicações SAP Fiori com base no core data services (CDS) views, estas através da queries que, adicionando um behaviour definition para transational basics ficam a funcionar como as operações antigas do CRUD.

Também abrange processos analíticos, transacionais e de aplicações de pesquisa.

O modelo de programação ABAP RESTful foi integrado diretamente no core da linguagem ABAP, introduzindo novos tipos de objeto de desenvolvimento ABAP transportáveis ​​para behavior definitions, service definitions, and service bindings, bem como também uma nova Entity Manipulation Language (EML).

Existem dois tipos de cenários que resultam de dois tipos de aplicação: Read-only application que requer o cds data model e uma aplicação analítica ou de pesquisa de anotações especifica. Transational applications que é o outro cenário, adicionalmente ao que o read only necessita, requer a criação de um business object baseado num behavior definition e a implementação para manipular as operações CRUD (create, read, update and delete) como o business logic implementado na ABAP.

QUAIS AS 5 CARACTERISTICAS?

O foco da aplicação é o user experience e tem 5 principais caracteristicas: role-based, adaptive ( adptavel), simple, coherent (coerente), and delightful.

SAP Fiori foca se em dar o role-based application aos utilizadores, o que significa que a aplicação oferece apenas o que é relevante para o utilizador. Este corre no browser o que é bom porque não precisamos de ter uma aplicação especifica para isto.

Delightful retira o que não é relevante para o trabalho impendindo distrações e enganos.

Coherent é coerente entre si, com todos os dispositivos.

Simple se a app tem muitos botões e muitas coisas, torna-se confuso e leva a desinteresse. O objetivo é deixar o utilizador acabar o que as suas tarefas quando vem a app com poucos clicks. Isto é tudo possivel com uma UI simples.

Adaptive tem haver com adapta se a qualquer ecrã, sem ficar desformatado.

SAP Fiori e SAPUI5 suportam os pedidos com base on the RESTful protocol data. Com in-memory database, SAP HANA, Podendo oferecer novas caracteristicas e melhorar a perfomance significativamente. HANA oferece também a combinação entre as capacidades transacionais com as capacidades analiticas que podem ser integradas numa aplicação Fiori.

A aplicação também oferece testability, supportability, and documentability permitindo ao utilizador ter a melhor experência possivel.

***Slide 4: Arquitetura***

**CDS** – suporta o esboço das funcionalidades e da extensibilidade e as implementações “fora da caixa” para assumir todas as implementações técnicas das tarefas.

**Business objects** – é um termo comum para representar um artefato do mundo da vida real numa aplicação, por exemplo mais a frente vamos ver uma aplicação de viagens e bookings as mesmas, a viagem é um business object.

**Queries** – é a interface de conexão para o acesso somente de leitura à database nos serviços OData. Sendo usado basicamente para relatórios de lista ou relatórios analíticos para processar os dados.

**Business object projection** – No modelo de programação ABAP RESTful é uma abordagem nativa de ABAP para os projetos e referencia de um subconjunto do business object para um business service espercifico. A projeção permite o consumo de serviço flexível, bem como designs de serviço baseados em funções.

**Business service** -- No contexto do modelo de programação de aplicação ABAP RESTful, um Business service é um serviço RESTful que pode ser chamado por um consumidor. É definido expondo o seu data model em conjunto com os comportamentos associados. Isto é, é o conjunto do service definition and a service binding.

**Service definition** – descreve quais entidades CDS de um data model vão ser expostos para que um business service específico, por exemplo, a gestão de pedido das viagens, possa ser ativado.

**Service binding** – um objeto do Repositório ABAP usado para ligar um service definition com o protocolo de comunicação cliente-servidor, como Odata.

**ODATA** – OData é um protocol que oferece mutias funcionalidade, como por exemplo, powerful querying e com a SAP Fiori e UI5, podendo criar Fiori applications.

**Web api** – Um serviço OData que é exposto como uma API da Web vem sem nenhuma informação específica da IU nos metadados. Isto é a interface pública para qualquer cliente OData ter acesso ao serviço OData.

**Sap fiori UI service** – Cada front-end configuração, que se manifesta no objeto de desenvolvimento de back-end (por exemplo, anotações de IU), é exposto nos metadados do serviço. Isso significa que uma IU Fiori lê as informações nos metadados e cria a interface do utilizador correspondente para o serviço. Essas configurações da IU podem ser ampliadas e sobrescritas no SAP Web IDE.

**EML -> Entity Manipulation Language –** é uma parte da linguagem abap que é usada para controlar o business object behaviou no contexto de abap restfull programming model.

***Slide 5: Evolução***

*Com o ABAP versão 7.4, a SAP começou a otimizar muito a plataforma ABAP para a sua database em memória SAP HANA. Neste contexto, ABAP Core Data Services (CDS) foi introduzido para oferecer a infraestrutura de modelagem de dados de última geração e acesso a dados. Os modelos de dados baseados em CDS são modelos de dados semanticamente ricos que podem ser usados ​​para todos os domínios de aplicações. Além disso, o SAP Gateway tornou-se parte integrante do AS ABAP com a versão 7.4 e o Business Object Processing Framework (BOPF) também entrou na plataforma ABAP com a versão 7.5.*

A SAP decidiu oferecer suporte a uma comunicação RESTful entre back-end e front-end. Combinado com a decisão de aplicar uma nova experiência do utilizador baseada em funções e de nível de consumidor usando um paradigma de design de IU moderno.

A SAP Fiori Elements permite a construção em blocos de reutilizacao e esquemas mais completos para haver uma maior produtividade no desenvolvimento de UI para os padroes de aplicações mais comuns

Odata como protocolo baseado em REST, incluindo metadados, que permite uma abordagem baseada em modelos e permite o desenvolvimento de UI baseado em SAP Fiori Elements the ABAP Platform for the business logic ABAP Development Tools in Eclipse e SAP Web IDE para desenvolvimento de backend e de frontend respetivamente.

Levando a arranjar uma maneira eficiente e padronizada de construir todos os tipos de aplicações

otimizadas para a SAP HANA, ou seja, aplicações transacionais, analíticas e de pesquisa. Promovendo

também melhores qualidades de produto de suporte, como trabalho contínuo, extensibilidade e

escalabilidade ao nivel da cloud. Outro aspectoimportante foi permitir uma fácil integração das pessoas

que desenvolvem as aplicações sem ter que ler toneladas de documentação primeiro.

***Slide 6: stategical long term solution***

Nós queremos que isto seja uma solução para o futuro, isto é que não nos obrigue daqui a 5 anos mudar tudo de novo ou adaptar grande parte das aplicações a uma nova. Assim, acrescentando ao que já disse anteriormente sobre a abap restful programming model, nós podemos utilizar greenfield developments. E o que isto é? São aplicações que se desenvolve a partir do zero, sem a reutilização de pedaços de código. A essa possibilidade chamamos a managed runtime. Ou seja, a partir do nada, podemos construir uma interface do utilizador completa com suporte para criar, ler, atualizar, excluir (CRUD operations). Deixando as determinações, validações e autorizações para quem está a desenvolver a app.

Uma aplicação que utiliza pedaços de código, chama se uma brownfield development, onde já é unmanaged runtime. Mas temos um buffer transacional onde implementamos as operações CRUD (create, read, update, delete).

O que estamos a ver nesta imagem é o business object que é dividido em tres partes: interaction phase, quando por exemplo carregamos num botão. Essa informação é escrita no transational buffer. Isto é uma aplicação stateless (por causa da comunicação do nosso ambiente cloud) e por isso não podemos estar a guardar constantemente, entao o draft torna se um recurso prórpio que é o intermédio entre o stateless da cloud e o estado natural da app de precisar de guardar as informações(save sequence).

***Slide 7: Vantagens e desvantagens***

Ler o que está ali e na desvantagens dizer que:

- DESVANTAGENS:

Algumas restrições como por exemplo a nivel numérico; a numeração é atrasada e por isso não é compativel com o tipo de implementação managed e tmbem algumas ações estão restringidas, como por exemplo a primary key nao pode usar valores com mais 117 caracteres ou por exemplo nao se pode usar estruturas de dicionário ABAP como parametro de acao de importacao ou exportação.

- O valor do filtro não é enviado ao back-end.

***Slide 8: Development flow***

Passando agora à parte de como é que é isto funciona mais esquemáticamente.

Começamos pelo modelo de dados com a database armazenada no dicionario ABAP e criada pela database da SAP HANA.

***Slide 9: CDS***

Depois temos o CDS que independentemente se a aplicação é do tipo read only ou transacional, a próxima etapa é esta. As cds views dão nos uma visão de como são as estruturas da tabela de base de dados e as suas relações.

***Slide 10: service consumption***

De seguida, para transformar um modelo de dados CDS num business object, o modelo CDS precisa de definir uma hierarquia. Com o modelo de programação ABAP RESTful, a definição da hierarquia foi totalmente integrada no core da linguagem ABAP (substituindo o uso de anotações no BOPF). Para adicionar o comportamento real do business object (operações CRUD e ações personalizadas), é necessária uma behavior definition com uma behavior implementation correspondente.

O behavior definition é o novo objecto que vai definir que ações é que cada business object. A implementação dessas operações é controlada pelo behavior implementation.

O nosso modelo de dados apenas se encontra na ABAP stack. Para ser usado pela SAP Fiori app ou outro Odata client, nós temos que expor o Odata service.

Antigamente fazia se uma processo complicado que envolvia a criação de um projeto de serviço SAP Gateway, mais o mapeamento das entitdades CDS para entidades Odata e uma implementação do serviço Gateway usando uma transaction especifica.

Mas com o ABAP RESTful programming model, a definição de que visualizações CDS expor como também as entidades OData foi integrada à linguagem ABAP na forma de uma service definition. Dentro de uma service definition, podemos, de maneira declarativa através de uma sintaxe semelhante ao do CDS, definir quais entidades CDS expor com nosso serviço OData sem a necessidade de transações ou ferramentas SAP GUI adicionais.

Com o service binding, podemos expor um modelo da aplicação ligando com o service binding a um protocolo de comunicação cliente-servidor específico, como Odata.

O tipo de ligação do service binding atualmente oferece suporte a dois cenários: um serviço de UI ou uma API da web. Um serviço de UI é um serviço OData que permite adicionar uma UI de elementos SAP Fiori ou outros clientes UI ao serviço, ao mesmo tempo que incorpora anotações de UI, enquanto uma API da web é um serviço OData puro usado para extensões imprevistas do software SAP.

***Slide 11: cds behaviour***

Nesta fase Aqui vamos criar o business object defintion para as viagens e para os bookings entities. Vamos definir o standard transational Business object behavior para cada entity. Depois vamos criar um business object behavior projection Project the transactional BO behavior para cada entity, o behavior projection é quem projeta os recursos transacionais a partir da behavior definition básico. Teremos o behavior definition projection projetando as operações: a criação, atualização e o delete.

***Slide 12: abap code***

Aqui basicamente o que fazemos é mostrar mensagens de erros quando não corresponde ao que é suposto, por exemplo:

***Slide 13: drafting enable***

Como já tinha falado anteriormente, em vez de estarmos sempre a gravar na database todos os pequenos movimentos, temos o draft que faz mais ou menos o papel de rascunho como por exemplo nas nossas caixas de email, quando estamos a escrever e ainda nao queremos mandar ou não passa mesmo de um rascunho, ele fica guardado lá para quando se tornar decisivo, usarmos ou não. Aqui acontece o mesmo, o draft permite nos estar a fazer uma reserva e por exemplo falta nos um dado no momento e so podemos completar mais tarde, ele guarda, sem nós criarmos.

***Slide 14: Tabelas***

Agora vou mostrar os resultados finais

Estão aqui as nossas base de dados da Travel e o do booking

***Slide 15: sap fiori element***

Aqui apresento o “aspecto” da app fiori element, podemos ver todas a viagens registadas, com o ID, quando começa, etc, depois podemos ver mais detalhadamente

***Slide 16: detalhadamente***

e carregar numa das viagens por exemplo e ver os bookings associados e se carregarmos num booking podemos ver o detalhe do mesmo.

***Slide 17:***

Aqui podemos ver que podemos pesquisar pelo nome ou pelo Id agency o nome, ou podemos se não soubermos bem o que estamos a procura podemo carregar ali no agency id e pesquisar através dos diversas caixinhas que estão aqui.

E Podemos selecionar um, por exemplo este do 70004.

***Slide 18:***

E vemos os resultados dessa pesquisa, depois também podemos mudar as configurações e escolher que parametros queremos que apareça, por exemplo que queremos o UUID, selecionamos e aparece assim depois no principal.

***Slide 19: create travel && Error messages***

Podemos criar, apagar e quando os dados são incorretos o que fazemos é mostrar mensagens de erros

***Slide 20: create booking***

Vamos criar agora um booking de uma viagem

***Slide 21: RASCUNHO ENABLE DRAFTING***

Como já tinha falado anteriormente, em vez de estarmos sempre a gravar na database todos os pequenos movimentos, temos o draft que faz mais ou menos o papel de rascunho como por exemplo nas nossas caixas de email, quando estamos a escrever e ainda nao queremos mandar ou não passa mesmo de um rascunho, ele fica guardado lá para quando se tornar decisivo, usarmos ou não. Aqui acontece o mesmo, o draft permite nos estar a fazer uma reserva e por exemplo falta nos um dado no momento e so podemos completar mais tarde, ele guarda, sem nós criarmos. Vou mostrar:

***Slide 22: Value help***

Muito rapidamente, nós queremos ter um sistema de ajuda, isto é nao sabemos bem o que estamos à procura, ou nao nos lembramos de algum dado. Como fazemos?

1º vamos fazer download do metadado dos ficheiro que contém o modelo de dados da entidade do serviço remoto OData. A seguir, vamos criar Service consumption model. Depois, vamos criar uma entidade CDS personalizada, aproveitando as entidades abstratas que estão a ser geradas no service consumption model que criamos.

De seguida, criamos uma entidade personalizada, onde teremos que implementar a classe ABAP onde tem a implementação de nossa query.

É aqui que é realizada a chamada para o serviço OData remoto. Depois de ter criado a entidade personalizada, podemos aprimorar o modelo de dados da nossa managed application, de modo que a entidade personalizada seja usada como a tal ajuda e usamos a validação de dados da agência, que também chamará o serviço OData remoto.

***Slide 23:***

Aqui podemos escolher o aspeto que queremos e que nos mais agrada e ver o value help que já referi e mostrei anteriormente também!