



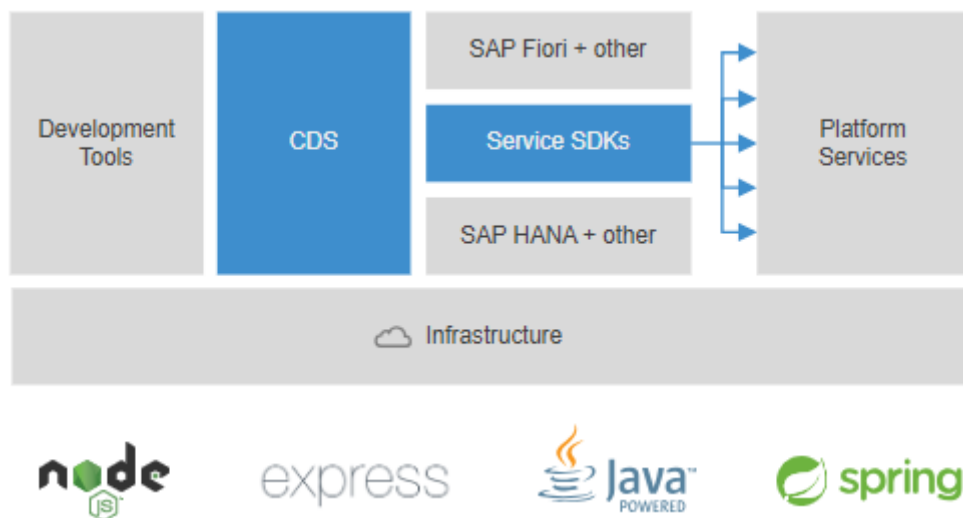
# Introdução ao CAP

Cloud Application Programming Model

Criado por: Gustavo Rhormens

Data: 04/07/2024

O CAP é uma estrutura de linguagens , bibliotecas e ferramentas para construir serviços e aplicativos de alto nível de complexidade. Sua estrutura apresenta uma combinação de tecnologias SAP e de código aberto amplamente adotadas, conforme destacado na figura abaixo:



Além das tecnologias de código aberto, o CAP acrescenta principalmente:

- Core Data Services (CDS) como nossa linguagem de modelagem universal para modelos de domínio e definições de serviço.
- SDKs de serviço e tempos de execução para Node.js e Java, oferecendo bibliotecas para implementar e consumir serviços, bem como implementações de provedores genéricos que atendem muitas solicitações automaticamente.

Portanto, o CAP é uma ferramenta SAP que revoluciona o modo de construir back-end services, mas mantendo uma linguagem de mercado, o que abre um espaço gigante para novas opções de solução, integração e modelagem, porém ainda assim mantém sua estrutura bem definida, garantindo fluxos automatizados que facilitam e agilizam o desenvolvimento.

## Estrutura de projeto

A estrutura básica do CAP é simples, podendo sofrer alterações e adição de componentes de acordo com a necessidade de cada projeto. Sua forma principal é a seguinte:

```
bookshop/      # Your project's root folder
├─ app/        # UI-related content
├─ srv/        # Service-related content
├─ db/         # Domain models and database-related content
├─ package.json # Configuration for cds + cds-dk
└─ readme.md   # A readme placeholder
```

Projeto/

|-- app/

|-- srv/

|-- db/

|-- package.json

|\_\_ readme.md

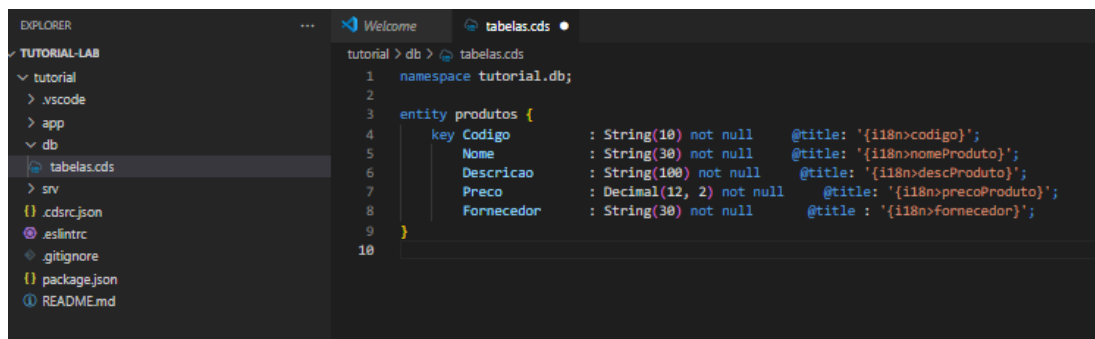
Essa estrutura é gerada automaticamente com os comandos “cds init nomeDoProjeto”, facilitando o processo de setup da aplicação de forma rápida e padronizada, seguindo as convenções de mercado e garantindo que ela funcione imediatamente, tornando o desenvolvimento mais rápido e seguro. Lembrando que o projeto pode ser desenvolvido na plataforma que preferir, como Visual Studio Code ou então dentro do ambiente BTP, com o BAS (Business Application Studio).

A estrutura do CAP é baseada em 3 pilares essenciais:

- DB é o repositório onde deveremos incluir as definições da estrutura das nossas tabelas,

declarando seu namespace, as entidades e suas colunas, definindo também o tipo de cada uma delas. Modelos de Domínio capturam aspectos estáticos de domínios como modelos de entidade-relacionamento bem conhecidos.

Exemplo:



```

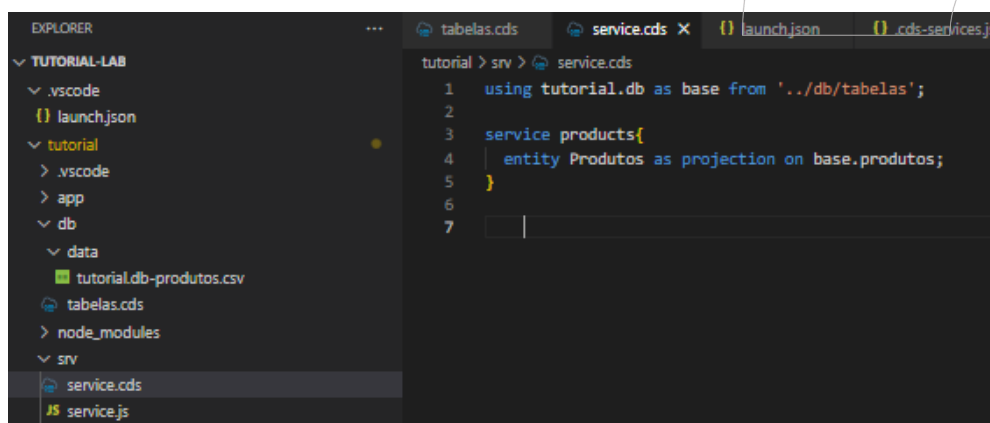
1 namespace tutorial.db;
2
3
4 entity produtos {
5   key Codigo      : String(10) not null @title: '{i18n>codigo}';
6   Nome           : String(30) not null @title: '{i18n>nomeProduto}';
7   Descricao      : String(100) not null @title: '{i18n>descProduto}';
8   Preco          : Decimal(12, 2) not null @title: '{i18n>precoProduto}';
9   Fornecedor     : String(30) not null @title: '{i18n>fornecedor}';
10 }

```

Aqui criamos um arquivo CDS com a estrutura das nossas tabelas, que no caso só possui uma entidade chamada “produtos”. Essa entidade tem 5 campos, sendo uma a sua chave.

- Dentro da pasta SRV, utilizamos os arquivos CDS e JS para expor os serviços e tratá-los da forma que for necessário, respectivamente. Os arquivos CDS nos dão a possibilidade de estabelecer a conexão com as tabelas criadas, de forma que podemos fazer a projeção das entidades baseada nas tabelas, associar e/ou compor dados a partir de outras entidades, construindo uma estrutura capaz de controlar os dados da melhor maneira. Já os arquivos JS têm a função de armazenar os fluxos necessários ao projeto com eventos e métodos capazes de controlar as ações da aplicação em momentos diferentes, como handlers de manipulação das chamadas antes, durante e após sua execução.

Exemplo:

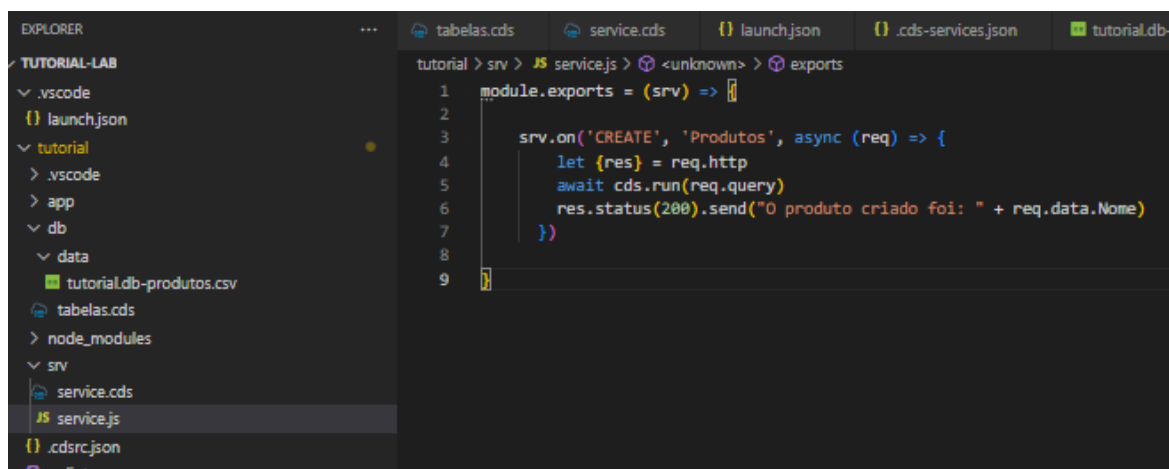


```

1 using tutorial.db as base from '../db/tabelas';
2
3
4 service products{
5   entity Produtos as projection on base.produtos;
6 }
7

```

No arquivo CDS criamos a entidade “Produtos” para projetar os dados da tabela “produtos”, a qual já havia sido criada anteriormente.



```

1 module.exports = (srv) => {
2
3   srv.on('CREATE', 'Produtos', async (req) => {
4     let {res} = req.http
5     await cds.run(req.query)
6     res.status(200).send("O produto criado foi: " + req.data.Nome)
7   })
8 }
9

```

Dentro do arquivo JS foi gerado um handler que será acionado no momento de execução da chamada, tratando a resposta que será devolvida ao solicitante do serviço. Note que o handler está configurado para ser acionado somente para os métodos POST da entidade “Produtos”. Isso mostra como o CAP dá total liberdade para gerenciar seus serviços da forma e momento que achar melhor.

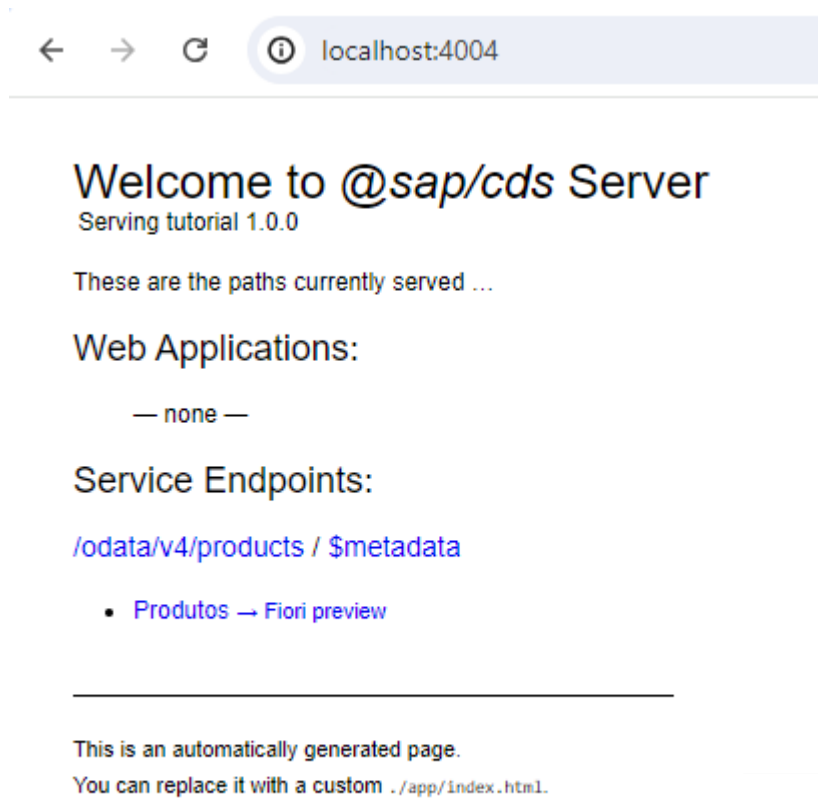
## Inicializando a aplicação

Para testar a aplicação, precisamos iniciá-la utilizando o comando “cds watch”. Esse comando dá o start na aplicação, disponibilizando o serviço em uma porta de servidor, que por padrão vem com a versão 4 do OData. Sem adicionar nenhum código de implementação do provedor, eles traduzem a solicitação OData em solicitações de banco de dados correspondentes e retornam os resultados como respostas OData.

Quando todas as definições de serviços estiverem prontas, o server (que já foi startado com o comando anterior) irá identificar essa estrutura e criar automaticamente uma base SQLite na memória.

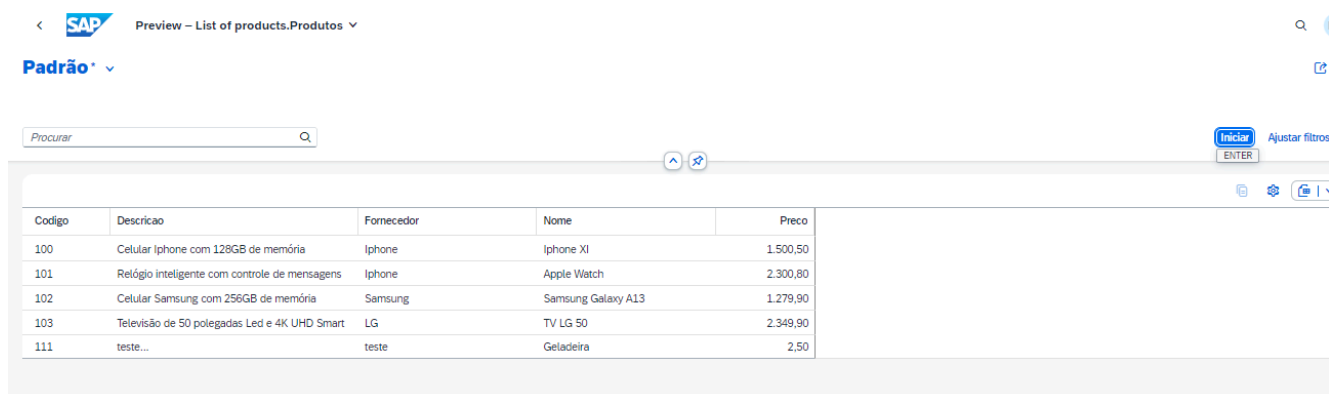
Opcionalmente, podemos testar e compilar modelos individualmente para verificar a validade e produzir uma saída analisada no formato CSN. Podemos executar, por exemplo, o comando cds “db/nome do arquivo.cds” para isso.

Acessando o localhost gerado para a aplicação, inicialmente poderá ver essa tela:



Aqui mostra os serviços e aplicações que definimos em nosso projeto. Nesse caso temos somente um serviço (products) e uma entidade visível (Produtos). Com o serviço startado temos as opções de visualizar os dados reais na base ou até mesmo visualizá-los através de

uma navegação visual, com auxílio do Fiori Elements:



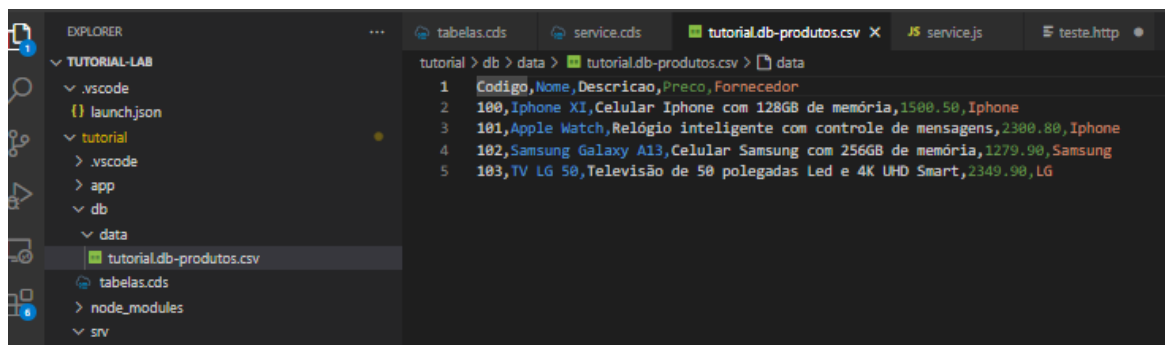
The screenshot shows the SAP Fiori Elements 'List of products' app. At the top, there's a header with the SAP logo, a breadcrumb 'Preview - List of products', and a 'Padrão' dropdown. Below the header is a search bar labeled 'Procurar' and a search icon. To the right of the search bar are buttons for 'Iniciar' (with an 'ENTER' label) and 'Ajustar filtros'. The main content area displays a table with the following data:

Codigo	Descricao	Fornecedor	Nome	Preco
100	Celular Iphone com 128GB de memória	Iphone	Iphone XI	1.500,50
101	Relógio inteligente com controle de mensagens	Iphone	Apple Watch	2.300,80
102	Celular Samsung com 256GB de memória	Samsung	Samsung Galaxy A13	1.279,90
103	Televisão de 50 polegadas Led e 4K UHD Smart	LG	TV LG 50	2.349,90
111	teste...	teste	Geladeira	2,50

## Trabalhando com dados

Tudo isso é possível graças à base gerada no momento em que a aplicação foi iniciada, conforme mostrado anteriormente, cds watch inicializa automaticamente um banco de dados SQLite em processo e na memória por padrão, isto é, a menos que seja informado o contrário. Embora isso não seja destinado ao uso produtivo, ele acelera drasticamente os tempos de desenvolvimento, essencialmente simulando seu banco de dados de destino, por exemplo, SAP HANA.

Contudo, para realização de testes locais é possível utilizar dados iniciais a partir de arquivos CSV gerados na pasta "DB". Espera-se que os nomes dos arquivos correspondam aos nomes totalmente qualificados das respectivas definições de entidade em seus modelos CDS, opcionalmente usando um traço, em vez de um ponto. O conteúdo desses arquivos é CSV padrão com os títulos das colunas correspondentes aos nomes dos elementos declarados. Porém tudo isso pode ser abreviado com a geração automática da estrutura dos dados iniciais utilizando o comando "cds add data". O arquivo com a nomenclatura baseada em sua entidade será gerado e você poderá incluir quais dados desejar.

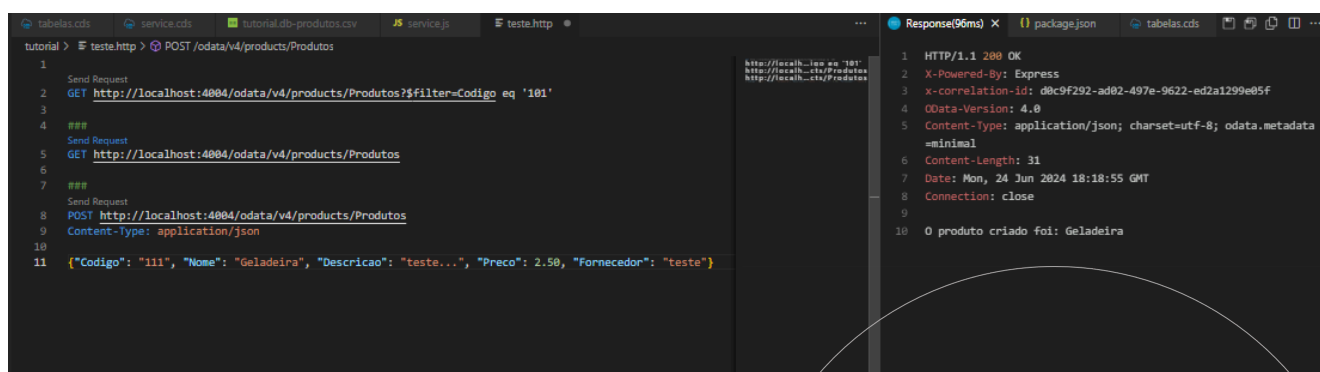


```

tutorial > db > data > tutorial.db-produtos.csv > data
1  Código, Nome, Descrição, Preço, Fornecedor
2  100, Iphone XI, Celular Iphone com 128GB de memória, 1500.50, Iphone
3  101, Apple Watch, Relógio inteligente com controle de mensagens, 2300.80, Iphone
4  102, Samsung Galaxy A13, Celular Samsung com 256GB de memória, 1279.90, Samsung
5  103, TV LG 50, Televisão de 50 polegadas Led e 4K UHD Smart, 2349.90, LG
  
```

Com essa estrutura básica, já é possível rodar sua aplicação com dados dinâmicos, realizar consultas e manipular dados. Para facilitar os testes, existe a possibilidade de criarmos um arquivo “teste.http” na raiz do projeto que irá simular as requests para os serviços expostos com o CAP.

Por exemplo, eu posso realizar um GET para verificar o retorno ou um POST para adicionar dados à base:



```

tutorial > teste.http > POST /odata/v4/products/Produtos
1
2  Send Request
3  GET http://localhost:4004/odata/v4/products/Produtos?$filter=Codigo eq '101'
4
5  ###
6  Send Request
7  GET http://localhost:4004/odata/v4/products/Produtos
8
9  ###
10 Send Request
11 POST http://localhost:4004/odata/v4/products/Produtos
12 Content-Type: application/json
13 { "Codigo": "111", "Nome": "Geladeira", "Descricao": "teste...", "Preco": 2.50, "Fornecedor": "teste" }
  
```

```

1  HTTP/1.1 200 OK
2  X-Powered-By: Express
3  x-correlation-id: 08c9f292-ad02-497e-9622-ed2a1299e05f
4  OData-Version: 4.0
5  Content-type: application/json; charset=utf-8; odata.metadata=minimal
6  Content-Length: 31
7  Date: Mon, 24 Jun 2024 18:18:55 GMT
8  Connection: close
9
10 0 produto criado foi: Geladeira
  
```

## Aprimorando modelos de domínio em CDS

Nossas definições de modelo (estruturas de tabela) podem receber funções a fim de criar regras de controle de acesso, tipagem de dados, associações, validação de entrada, dentre outras funções, além de facilitar sua criação, organização e também a reutilização de declarações comuns, fazendo uso das melhores práticas de desenvolvimento. Algumas das principais funções que podem ser aplicadas são as associações e composições entre entidades e suas anotações.

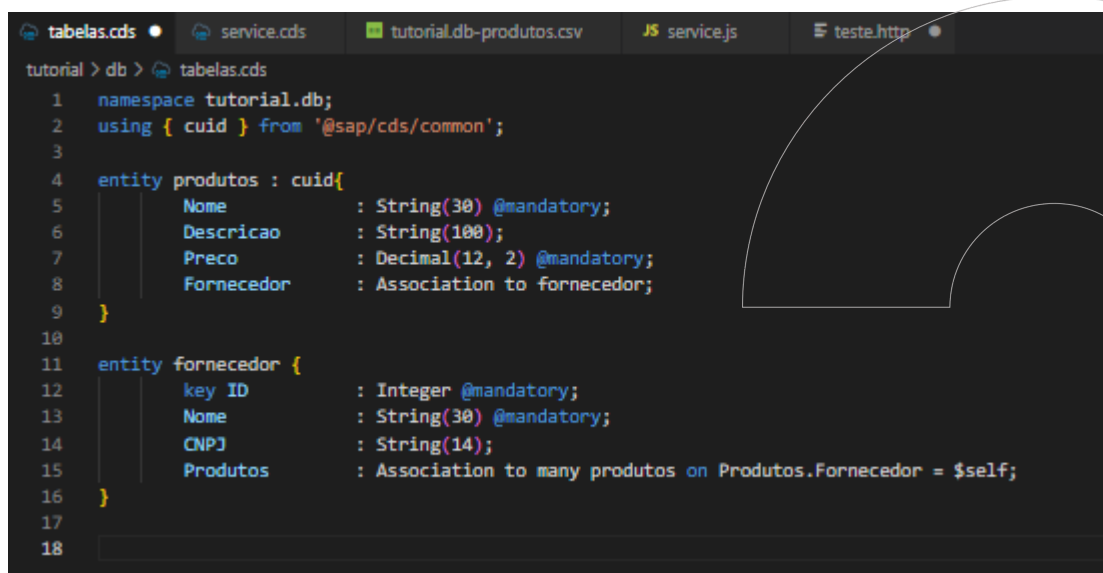


As Associações capturam relacionamentos entre entidades. Elas são como junções declaradas para frente adicionadas a uma definição de tabela em SQL, o que isso quer dizer? Com associações é possível criar definições de tabelas que mantêm dados cruzados entre uma ou mais entidades, de forma que através de foreign keys seja possível atribuir dados de uma à outra. Elas podem ser, , se tratando de dados entre ambas, “de 1 para 1” como também de “1 para vários”.

Composições constituem estruturas de documentos por meio de relacionamentos contidos . Elas frequentemente aparecem em cenários de cabeçalho-filho”para-vários”, ou seja, trabalham com os dados filhos da mesma entidade onde há vários grupos de dados vinculados entre si. Composições estendem isso para modelar facilmente estruturas de documentos.

Por fim, as anotações permitem enriquecer modelos com metadados adicionais, como UIs, validações, validação de entrada, autorização e nomenclaturas. Essas anotações são capazes de determinar o funcionamento das entidades, bem como atribuir valores de forma padronizada, evitando a necessidade de manipulações em tempo de processo do serviço.

Abaixo veremos um exemplo de como enriquecer nossas definições de modelo com alguns desses atributos:



```
tutorial > db > tabelas.cds
1 namespace tutorial.db;
2 using { cuid } from '@sap/cds/common';
3
4 entity produtos : cuid{
5     Nome          : String(30) @mandatory;
6     Descricao     : String(100);
7     Preco         : Decimal(12, 2) @mandatory;
8     Fornecedor    : Association to fornecedor;
9 }
10
11 entity fornecedor {
12     key ID        : Integer @mandatory;
13     Nome          : String(30) @mandatory;
14     CNPJ          : String(14);
15     Produtos      : Association to many produtos on Produtos.Fornecedor = $self;
16 }
17
18
```

```

tabelas.cds  service.cds X tutorial.db-produtos.csv JS service.js
tutorial > srv > service.cds
1  using tutorial.db as base from '../db/tabelas';
2  using { cuid } from '@sap/cds/common';
3
4  service products{
5      entity Produtos as projection on base.produtos;
6      @readonly entity Fornecedor as projection on base.fornecedor;
7  }
8
9

```

Observe como foram empregadas as anotações para automatizar a criação de PKs (cuid gera UUID) e determinar os campos obrigatórios (mandatory) dentro da definição de modelo, além de criar validação de entrada para permitir somente leitura de uma das entidades (readonly).

Além disso, foram atribuídos valores cruzados entre ambas as entidades com o auxílio de associações, dando a possibilidade de visualizar dados vinculados dentro das requisições:



```

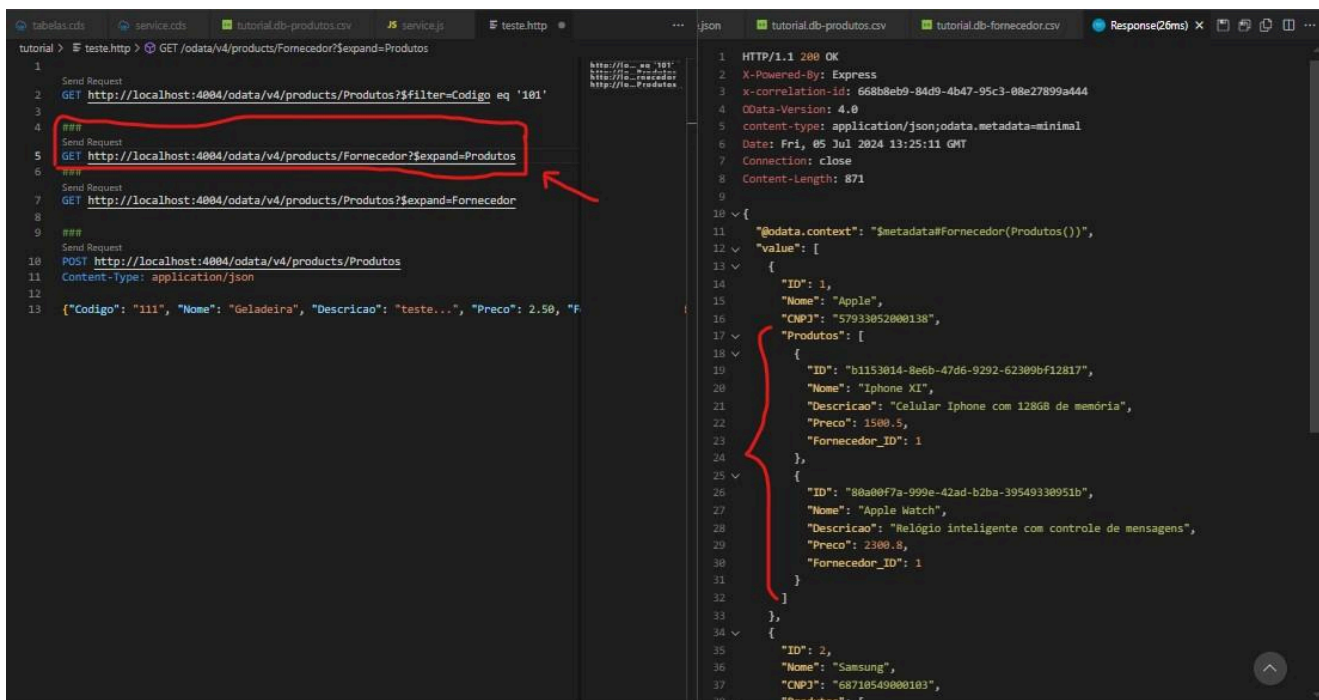
1  tutorial > teste.http > GET /odata/v4/products/Fornecedor?$expand=Produtos
2  Send Request
3  GET http://localhost:4804/odata/v4/products/Produtos?$filter=Codigo eq '101'
4  ###
5  Send Request
6  GET http://localhost:4804/odata/v4/products/Fornecedor?$expand=Produtos
7  ###
8  Send Request
9  GET http://localhost:4804/odata/v4/products/Produtos?$expand=Fornecedor
10 ###
11 Send Request
12 POST http://localhost:4804/odata/v4/products/Produtos
13 Content-Type: application/json
14 {"Codigo": "111", "Nome": "Geladeira", "Descricao": "teste...", "Preco": 2.50, "F

```

```

1  HTTP/1.1 200 OK
2  X-Powered-By: Express
3  x-correlation-id: ca4a5d91-2f90-4ef1-a6a3-735e542504ce
4  OData-Version: 4.0
5  content-type: application/json;odata.metadata=minimal
6  Date: Fri, 05 Jul 2024 13:31:43 GMT
7  Connection: close
8  Content-Length: 932
9
10 {
11   "@odata.context": "$metadata#Produtos(Fornecedor())",
12   "value": [
13     {
14       "ID": "7b737e56-1a72-4ef6-8581-e5d7e83ebe0e",
15       "Nome": "Samsung Galaxy A13",
16       "Descricao": "Celular Samsung com 256GB de memória",
17       "Preco": 1279.0,
18       "Fornecedor_ID": 2,
19       "Fornecedor": {
20         "ID": 2,
21         "Nome": "Samsung",
22         "CNPJ": "768716549000103"
23       },
24     },
25     {
26       "ID": "80a0ef7a-999e-42ad-b2ba-39549330951b",
27       "Nome": "Apple Watch",
28       "Descricao": "Relógio inteligente com controle de mensagens",
29       "Preco": 2300.8,
30       "Fornecedor_ID": 1,
31       "Fornecedor": {
32         "ID": 1,
33         "Nome": "Apple",
34         "CNPJ": "57933052000138"
35       },
36     },
37     {
38       "ID": "b1153014-8e6b-47d6-9292-62309bf12817",

```



```

1  teste.http
2  GET /odata/v4/products/Fornecedor?$expand=Produtos
3
4  ###
5  GET http://localhost:4004/odata/v4/products/Fornecedor?$expand=Produtos
6  ###
7  Send Request
8  GET http://localhost:4004/odata/v4/products/Produtos?$expand=Fornecedor
9  ###
10 Send Request
11 POST http://localhost:4004/odata/v4/products/Produtos
12 Content-Type: application/json
13 {"Codigo": "111", "Nome": "Geladeira", "Descricao": "teste...", "Preco": 2.50, "F...

1  HTTP/1.1 200 OK
2  X-Powered-By: Express
3  x-correlation-id: 668b8eb9-84d9-4b47-95c3-08e27899a444
4  OData-Version: 4.0
5  content-type: application/json;odata.metadata=minimal
6  Date: Fri, 05 Jul 2024 13:25:11 GMT
7  Connection: close
8  Content-Length: 871
9
10 {
11   "@odata.context": "$metadata#Fornecedor(Produtos())",
12   "value": [
13     {
14       "ID": 1,
15       "Nome": "Apple",
16       "CNPJ": "57933052000138",
17       "Produtos": [
18         {
19           "ID": "b1153014-8eb6-47d6-9292-62309bf12817",
20           "Nome": "Iphone XI",
21           "Descricao": "Celular Iphone com 128GB de memória",
22           "Preco": 1500.5,
23           "Fornecedor_ID": 1
24         },
25         {
26           "ID": "88a00f7a-999e-42ad-b2ba-39549330951b",
27           "Nome": "Apple Watch",
28           "Descricao": "Relógio inteligente com controle de mensagens",
29           "Preco": 2300.8,
30           "Fornecedor_ID": 1
31         }
32       ]
33     },
34     {
35       "ID": 2,
36       "Nome": "Samsung",
37       "CNPJ": "68710549000103",
38       "Produtos": [
  
```

## Deploy da aplicação (MTA)

Existem variadas formas de realizar o deploy de uma aplicação CAP. Nesse documento vamos apresentar a forma simples de realizar esta ação com MTA (Multi-Target Application). As instruções básicas para realizar o deploy com mta de uma aplicação são simples, porém é necessário prepará-la para tal. Primeiro certifique-se que possui o MBT (Cloud MTA Build Tool) instalado e gere o arquivo MTA:

- para checar se já está instalado utilize o comando "mbt -version"
- caso seja necessário instalar utilize o comando "npm install -global mbt"
- para gerar o arquivo mta, utilize o comando "cds add mta"

O segundo passo seria se conectar com o ambiente que deseja realizar o deploy. Para isso, conecte ao Cloud Foundry com os comandos:

- cf api <API Endpoint of your landscape>
- cf login (será solicitado seu e-mail e senha)
- escolha qual ambiente deseja atribuir seu deploy

\* o endpoint e nome da organização do seu ambiente destino poderá ser encontrado na aba "overview", dentro da sub-account de destino

Após estes passos sua aplicação já estará conectada com o ambiente destino. Agora precisamos gerar o build da aplicação e para isso basta utilizar o comando:

- mbt build -t ./ (o “./” garante que está buildando o projeto a partir de sua raiz)

Agora é só realizar o deploy de fato, com o comando:

- CF deploy <ID da aplicação>

\* o ID da aplicação, assim como sua versão, estão configurados dentro do arquivo MTA.yaml

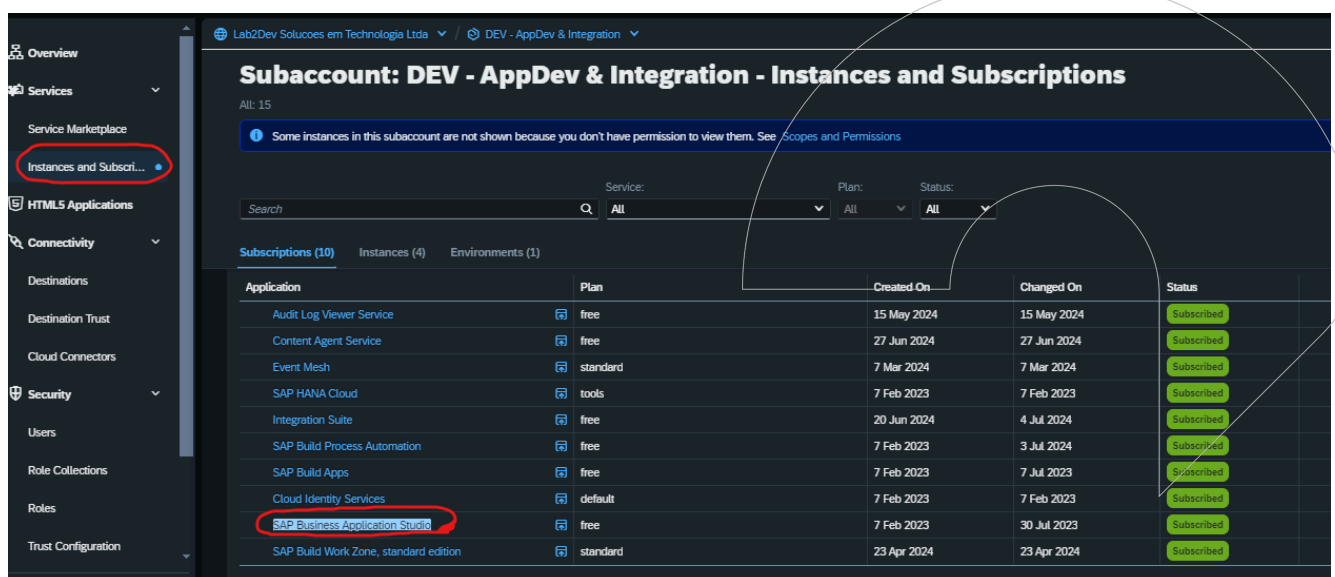
Realizado este processo, sua aplicação será enviada para o ambiente destino, ficando disponível para utilização interna.

## Criação de Dev Space (BAS)

A ferramenta de desenvolvimento a ser empregada para criação de um projeto CAP pode ser escolhida de acordo com o gosto do programador, porém a SAP possui em seu próprio dashboard diversas ferramentas que auxiliam no desenvolvimento. O BAS (Business Application Studio) é similar ao já conceituado Visual Studio Code e tem a vantagem de já estar dentro da subaccount destino, o que concede maior segurança ao projeto como um todo.

Para criar um projeto lá dentro é simples:

- Com a instância do BAS configurada na sub-account, entre nela:



Subaccount: DEV - AppDev & Integration - Instances and Subscriptions

All: 15

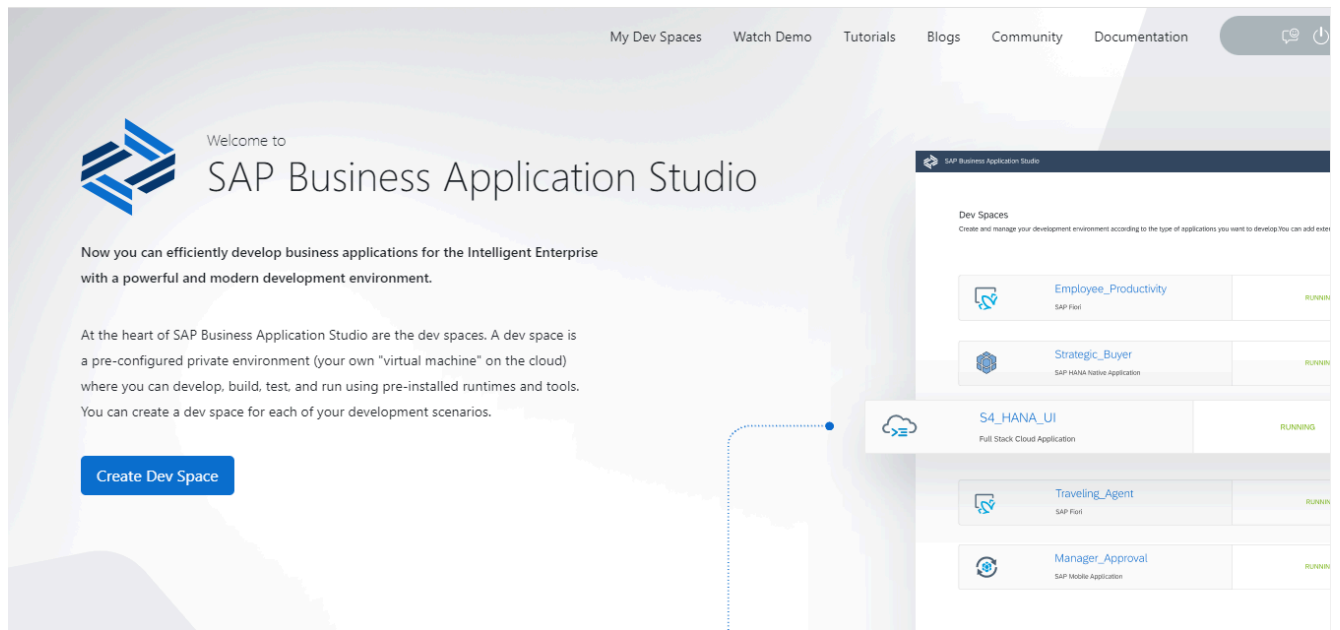
Some instances in this subaccount are not shown because you don't have permission to view them. See [Scopes and Permissions](#)

Search: [ ] Service: [All] Plan: [All] Status: [All]

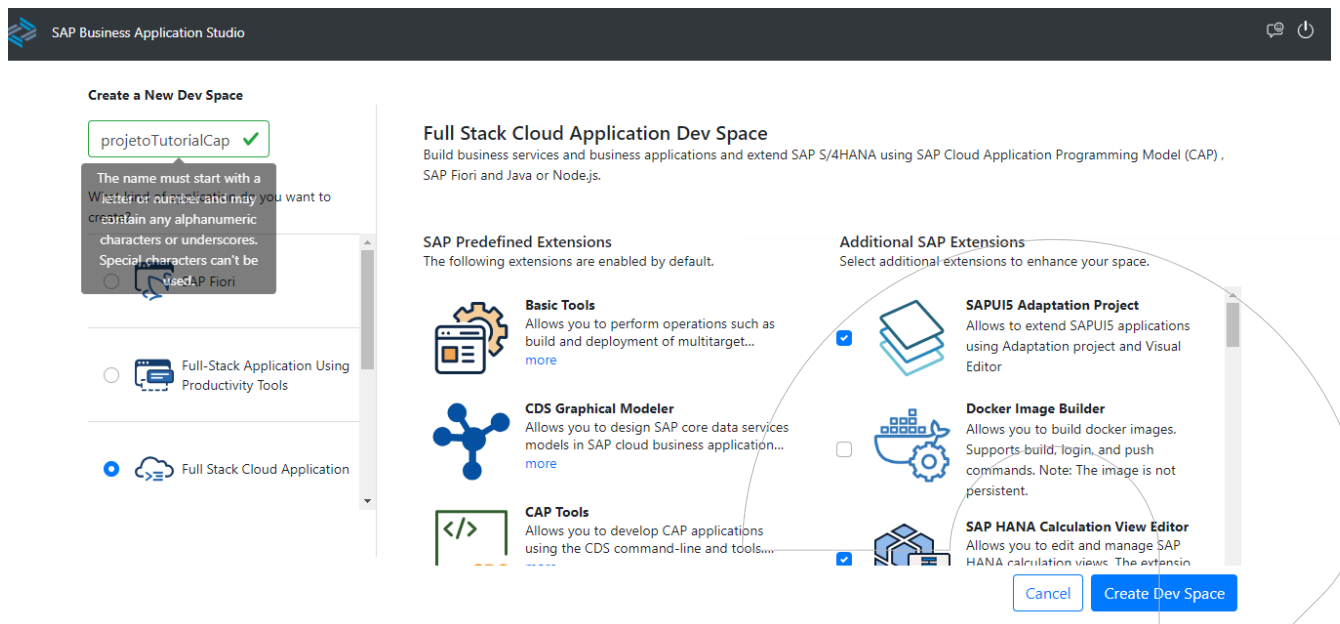
Subscriptions (10) Instances (4) Environments (1)

Application	Plan	Created On	Changed On	Status
Audit Log Viewer Service	free	15 May 2024	15 May 2024	Subscribed
Content Agent Service	free	27 Jun 2024	27 Jun 2024	Subscribed
Event Mesh	standard	7 Mar 2024	7 Mar 2024	Subscribed
SAP HANA Cloud	tools	7 Feb 2023	7 Feb 2023	Subscribed
Integration Suite	free	20 Jun 2024	4 Jul 2024	Subscribed
SAP Build Process Automation	free	7 Feb 2023	3 Jul 2024	Subscribed
SAP Build Apps	free	7 Feb 2023	7 Jul 2023	Subscribed
Cloud Identity Services	default	7 Feb 2023	7 Feb 2023	Subscribed
<b>SAP Business Application Studio</b>	free	7 Feb 2023	30 Jul 2023	Subscribed
SAP Build Work Zone, standard edition	standard	23 Apr 2024	23 Apr 2024	Subscribed

- Será redirecionado para a página do BAS onde poderá criar seu dev space

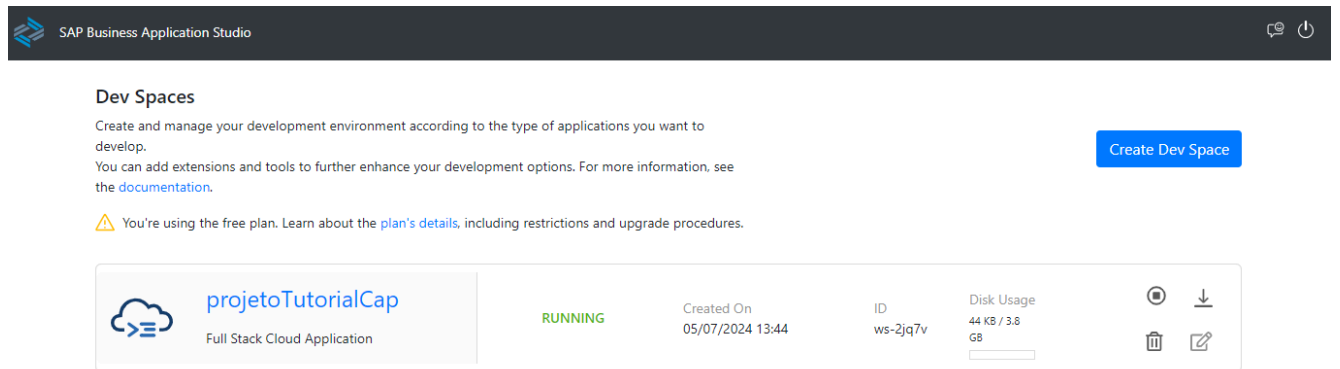


- Após clicar em “create Dev Space” será redirecionado para a página de configuração inicial do projeto:



- Insira um nome para o projeto, escolha o tipo de aplicação, nesse caso para o CAP pode ser o Full Stack Cloud Application. Então as extensões essenciais serão vinculadas ao seu projeto, podendo ainda selecionar outras adicionais que sejam interessantes de acordo com cada objetivo.

- O dev space foi criado. Aguarde até que seja iniciado (starting > running)

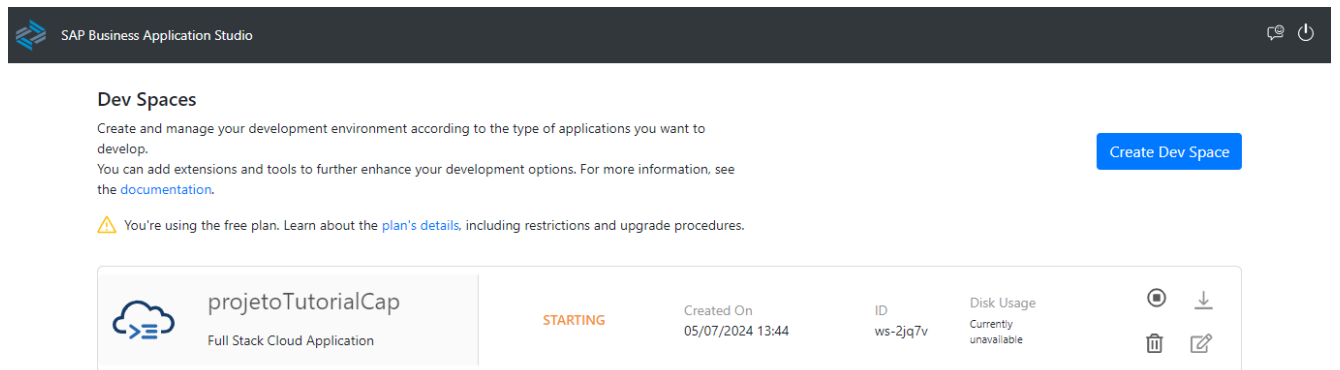


**Dev Spaces**

Create and manage your development environment according to the type of applications you want to develop. You can add extensions and tools to further enhance your development options. For more information, see the [documentation](#).

⚠ You're using the free plan. Learn about the [plan's details](#), including restrictions and upgrade procedures.

Icon	projetoTutorialCap	Status	Created On	ID	Disk Usage	Actions
	Full Stack Cloud Application	RUNNING	05/07/2024 13:44	ws-2jq7v	44 KB / 3.8 GB	



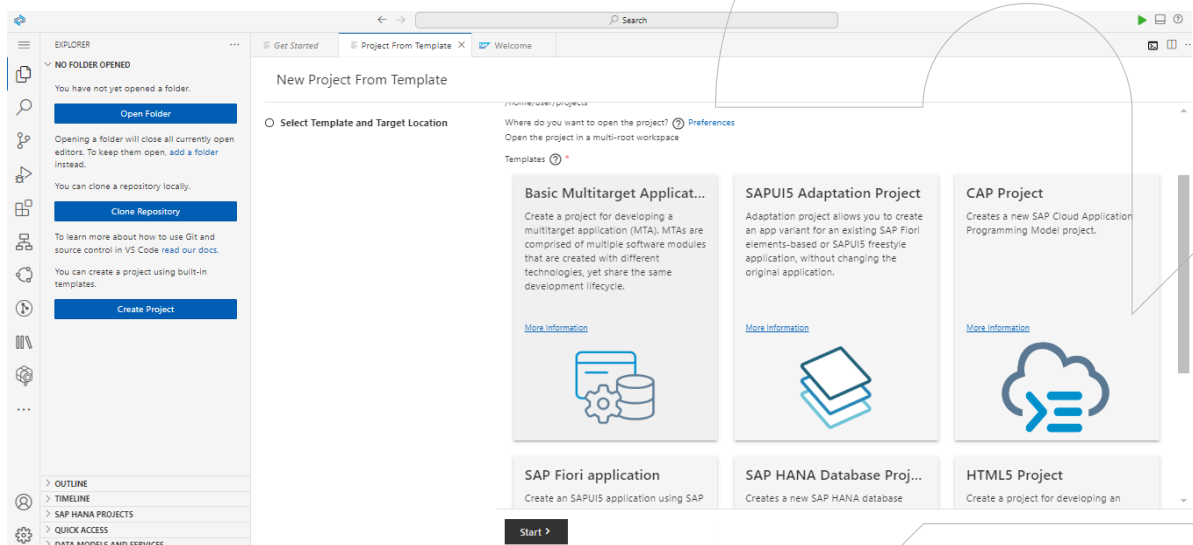
**Dev Spaces**

Create and manage your development environment according to the type of applications you want to develop. You can add extensions and tools to further enhance your development options. For more information, see the [documentation](#).

⚠ You're using the free plan. Learn about the [plan's details](#), including restrictions and upgrade procedures.

Icon	projetoTutorialCap	Status	Created On	ID	Disk Usage	Actions
	Full Stack Cloud Application	STARTING	05/07/2024 13:44	ws-2jq7v	Currently unavailable	

- Agora poderá acessar seu dev space e criar um o projeto. Existe a possibilidade de utilizar as ferramentas de geração de template automática do BAS que auxilia a pular algumas etapas de set up da aplicação: clique em “New Project From Template” > CAP Project > sete as informações necessárias e que façam sentido ao projeto > Finish



**New Project From Template**

Select Template and Target Location

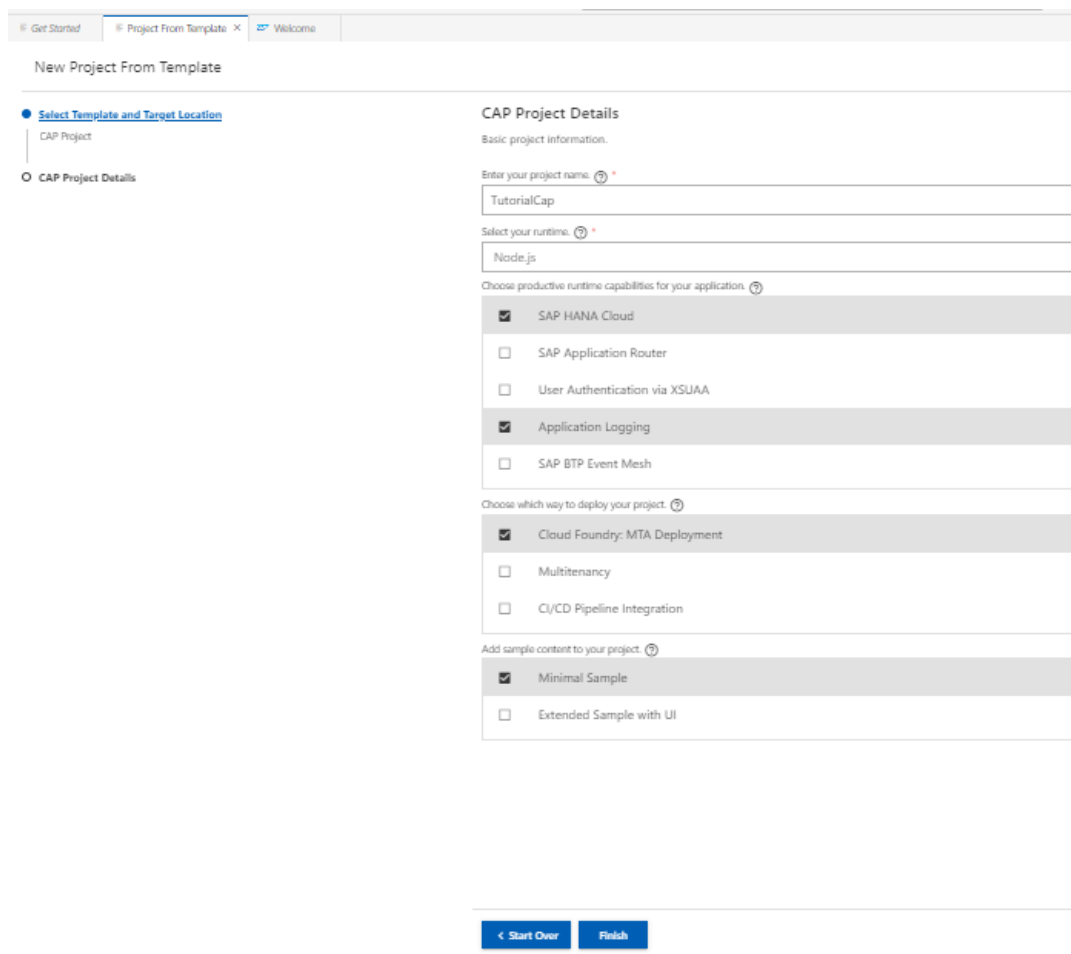
Where do you want to open the project? [Preferences](#)

Open the project in a multi-root workspace

Templates

- Basic Multitarget Application...**  
Create a project for developing a multitarget application (MTA). MTAs are comprised of multiple software modules that are created with different technologies, yet share the same development lifecycle.  
[More Information](#)
- SAPUI5 Adaptation Project**  
Adaptation project allows you to create an app variant for an existing SAP Fiori elements-based or SAPUI5 freestyle application, without changing the original application.  
[More Information](#)
- CAP Project**  
Creates a new SAP Cloud Application Programming Model project.  
[More Information](#)
- SAP Fiori application**  
Create an SAPUI5 application using SAP
- SAP HANA Database Proj...**  
Creates a new SAP HANA database
- HTML5 Project**  
Create a project for developing an

**Start >**



Get Started | **Project From Template** | Welcome

New Project From Template

- Select Template and Target Location**
  - CAP Project
- CAP Project Details

### CAP Project Details

Basic project information.

Enter your project name:

Select your runtime:

Choose productive runtime capabilities for your application:

- ☒ SAP HANA Cloud
- ☐ SAP Application Router
- ☐ User Authentication via XSUAA
- ☒ Application Logging
- ☐ SAP BTP Event Mesh

Choose which way to deploy your project:

- ☒ Cloud Foundry: MTA Deployment
- ☐ Multitenancy
- ☐ CI/CD Pipeline Integration

Add sample content to your project:

- ☒ Minimal Sample
- ☐ Extended Sample with UI

[< Start Over](#) [Finish](#)

- Pronto! É só acessar e começar a desenvolver.

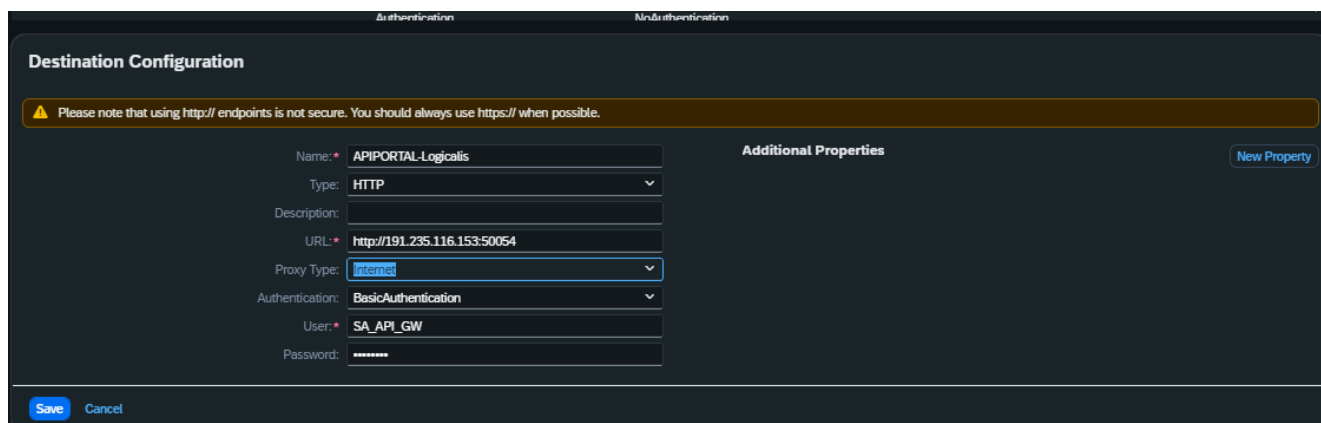
## Configuração de destination

As destinations são túneis seguros que a SAP oferece para que as aplicações possam acessar serviços externos de forma segura. Sua configuração dentro da sub-account do ambiente de destino deve ser criada de acordo com os objetivos do projeto.

Para configurar uma nova destination é necessário que o Cloud Connector esteja previamente criado também, com isso basta acessar: Sub-account > Connectivity > Destinations > Create Destination

Ao criar a destination serão solicitados o nome (qualquer nome), tipo (normalmente HTTP para consumo de uma URL), descrição (qualquer descrição), URL (conforme o Cloud Connector), Tipo de proxy e Autenticação (varia de acordo com o objetivo do projeto):

Por exemplo:



Aqui foi empregado a autenticação básica (com usuário e senha), mas em um ambiente de produção é comum o emprego da autenticações que validem o user SAP logado diretamente, como o Principal Propagation, garantindo a validação de acordo com o solicitante.

## Conclusão

A estrutura básica do CAP trabalha em cima do modelo das definições de base de dados (DB), exposição de entidades associadas à essa base (servico.CDS) e a manipulação dos dados requisitados (servico.JS), podendo ainda fazer uso de atributos nativos que viabilizam fluxos mais complexos de forma simples e verticalizada. A partir daí já temos o ambiente mínimo necessário para expor serviços via OData para consumo de outras aplicações, como por exemplo o Fiori.

Existe ainda uma infinidade de possibilidades dentro do CAP, podendo criar anotações para manipulação dos dados de forma nativa, estruturas customizadas para integrar entidades interdependentes, jobs de execução programada, integração para consumo de remote services, entre outras soluções importantes para diversos tipos de projetos.

A documentação do CAP possui todas essas referências, mas com experiência e conhecimento da estrutura principal desse tipo de aplicação, fica mais claro entender como aplicar cada um desses métodos e construir um projeto sólido.

Para mais detalhes de ferramentas disponibilizadas dentro do CAP, consulte:

<https://cap.cloud.sap/docs/about/>



