emineo so geht digital _

Exercise 3

Implement new Business Service with CAP Model

Table of Contents

1. Objectives . . . 1
2. Implementation . . . 2
    2.1. Service Definition . . . 2
    2.2. Change dev space . . . 2
    2.3. Create a new application `ratings-srv` . . . 2
    2.4. Folder structure . . . 2
    2.5. Create new schema . . . 2
    2.6. Create new service . . . 3
    2.7. Test the service . . . 3
    2.8. Add mock data . . . 4
    2.9. Calculate the `star*Perc` properties . . . 5
    2.10. Test the service . . . 6
    2.11. Deploy to Cloud Foundry . . . 7

1. Objectives

- Create a simple service application
- Add mock data
- Learn to use CDS-Views and the Cloud Foundry CLI
- Test the service
- Deploy and verify the service

2. Implementation

2.1. Service Definition

We want to add ratings view (1 to 5 Star) for specified products. Ratings shall be aggregated as percentage of all ratings and displayed as barchart. The data for this functionality will be persisted in Hana-DB instance of the cloud.

2.2. Change dev space

Since we need access to CDS toolchain we want to use "cds" dev space.

2.3. Create a new application `ratings-srv`

Open new terminal in Business Application Studio and execute following commands:

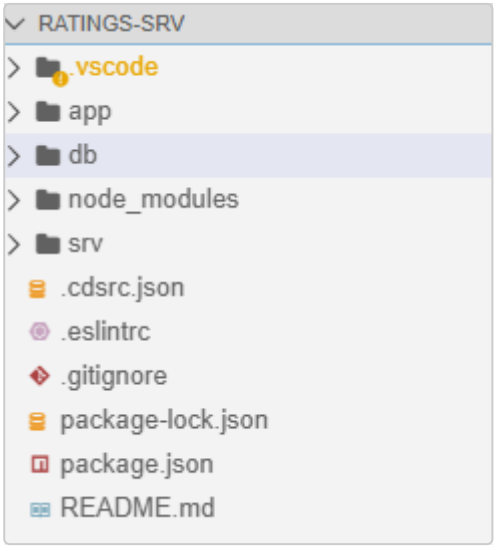
```
$ mkdir ratings-srv
$ cd ratings-srv
$ cds init
```

Install all relevant dependencies

```
$ npm install
```

2.4. Folder structure

Make sure that following folder structure is generated.



2.5. Create new schema

Create entities "Ratings" and "ProductRatings". Entity "Ratings" contains unique rating record. Entity "ProductRatings" contains composition or Ratinds and supplies aggregated view of all corresponding Ratings as percentage.
In folder db create new file schema.cds and paste following code.

schema.cds

```
namespace db;

entity Ratings {
   key RatingID  : Integer;
      ProductID : Integer;
      Name      : String;
      Date      : DateTime;
      Rating    : Integer;
}

entity ProductRatings {
   key ProductID : Integer;
      star1     : Integer;
      star2     : Integer;
      star3     : Integer;
      star4     : Integer;
      star5     : Integer;
      Ratings   : Composition of many Ratings
                  on ProductID = Ratings.ProductID;
}
```

2.6. Create new service

In srv folder create a new file ratings-service.cds and paste following code.

ratings-service.cds

```
using {db} from '../db/schema';

@path : '/ratings'    ②

service RatingsService { ①

   entity Ratings       as projection on db.Ratings; ③

   entity ProductRatings as  ④
      select from db.ProductRatings {
         *,
         null as count     : Integer,
         null as star1Perc : DecimalFloat, ⑤
         null as star2Perc : DecimalFloat,
         null as star3Perc : DecimalFloat,
         null as star4Perc : DecimalFloat,
         null as star5Perc : DecimalFloat
      };
}
```

① Define service RatingsService.

② Serve this service under the path /ratings.

③ Use previously defined entity Ratings.

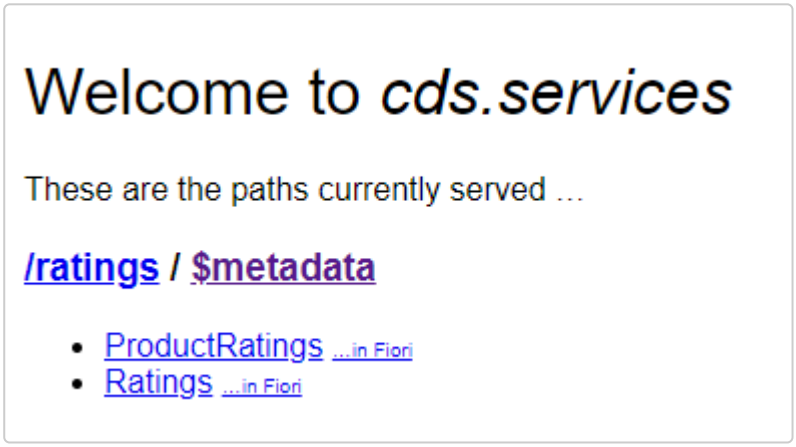
④ Use previously defined entity ProductRatings.

⑤ Add "virtual" attributes star1Perc, etc.

2.7. Test the service

- In terminal run cds watch

- At popup windows choose "Open in New Tab" - out service is up and running



- Add /ratings/$metadata at the end of URL to show the metadata

2.8. Add mock data

- Mock data can be added with .csv files
- Files have to be named according the Entities with the corresponding namespace
- In db directory create new data directory
- Create files db.Ratings.csv and db.ProductRatings.csv

Content of db.Ratings.csv

```
RatingID;ProductID;Name;Date;Rating
1;1;John;2020-01-01T17:00:00Z;3
2;1;Mary;2020-01-03T17:00:00Z;5
3;1;Bart;2020-01-03T17:00:00Z;3
4;1;Eva;2020-01-05T17:00:00Z;4
5;1;Garry;2020-01-16T17:00:00Z;2
6;1;Michele;2020-01-18T17:00:00Z;5
7;2;John;2020-01-01T17:00:00Z;3
8;2;Mary;2020-01-03T17:00:00Z;4
9;2;Bart;2020-01-03T17:00:00Z;2
10;2;Eva;2020-01-05T17:00:00Z;1
11;2;Jane;2020-01-05T17:00:00Z;3
12;2;Michele;2020-01-18T17:00:00Z;3
```

Content of db.ProductRatings.csv

```
ProductID;star1;star2;star3;star4;star5
1;0;1;2;1;2
2;1;1;3;1;0
```

- With cds watch still running you will see the service is filled with data from the 2 files

```
[cds] - using bindings from: { registry: '~/.cds-services.json' }
[cds] - connect to db > sqlite { database: ':memory:' }
 > filling db.ProductRatings from db/data/db.ProductRatings.csv
 > filling db.Ratings from db/data/db.Ratings.csv
/> successfully deployed to sqlite in-memory db
```

- In browser try the path /ratings/ProductRatings(1)?$expand=Ratings


```
{
    "@odata.context": "$metadata#ProductRatings(Ratings())/$entity",
    "ProductID": 1,
    "star1": 0,
    "star2": 1,
    "star3": 2,
    "star4": 1,
    "star5": 2,
    "count": null,
    "star1Perc": null,
    "star2Perc": null,
    "star3Perc": null,
    "star4Perc": null,
    "star5Perc": null,
    "Ratings": [
        {
            "RatingID": 1,
            "ProductID": 1,
            "Name": "John",
            "Date": "2020-01-01T17:00:00Z",
            "Rating": 3
        },
        {
            "RatingID": 2,
            "ProductID": 1,
            "Name": "Mary",
            "Date": "2020-01-03T17:00:00Z",
            "Rating": 5
        },
```


2.9. Calculate the star*Perc properties

- Create new file ratings-service.js
- Copy the code

ratings-service.js

```
const { context } = require("@sap/cds");
const cds = require("@sap/cds");

module.exports = cds.service.impl(async (service) => {
  const { Products } = service.entities;

  service.after("READ", "ProductRatings", (context, req) => {
    if (context.length === 0) {
      context.push({
        ProductID: req.data.ProductID,
        star1: 0,
        star2: 0,
        star3: 0,
        star4: 0,
        star5: 0,
        count: 0,
        star1Perc: 0,
        star2Perc: 0,
        star3Perc: 0,
        star4Perc: 0,
        star5Perc: 0,
      });
    } else {
      context.map((e) => {
        e.count = e.star1 + e.star2 + e.star3 + e.star4 + e.star5;
        e.star1Perc = (e.star1 / e.count) * 100;
        e.star2Perc = (e.star2 / e.count) * 100;
        e.star3Perc = (e.star3 / e.count) * 100;
        e.star4Perc = (e.star4 / e.count) * 100;
        e.star5Perc = (e.star5 / e.count) * 100;
      });
    }
  });
});
```

2.10. Test the service

In browser call path /ratings/ProductRatings(1)?$expand=Ratings

```
{
    "@odata.context": "$metadata#ProductRatings(Ratings())/$entity",
    "ProductID": 1,
    "star1": 0,
    "star2": 1,
    "star3": 2,
    "star4": 1,
    "star5": 2,
    "count": 6,
    "star1Perc": 0,
    "star2Perc": 16.666666666666664,
    "star3Perc": 33.33333333333333,
    "star4Perc": 16.666666666666664,
    "star5Perc": 33.33333333333333,
    "Ratings": [
        {
            "RatingID": 1,
            "ProductID": 1,
            "Name": "John",
            "Date": "2020-01-01T17:00:00Z",
            "Rating": 3
        },
        {
            "RatingID": 2,
            "ProductID": 1,
            "Name": "Mary",
            "Date": "2020-01-03T17:00:00Z",
            "Rating": 5
        },
```


Out star*Perc entities are being calculated

2.11. Deploy to Cloud Foundry

In order to deploy runable service add following to package.json file:

package.json

```
"cds": {
  "requires": {
    "db": {
      "kind": "sql"
    }
  }
}
```

In terminal run:

```
$ npm add @sap/hana-client --save
$ cf login ①
```

① Login with your credentials.

```
Password: user: ratings-srv $ cf login
API endpoint: https://api.cf.eu10.hana.ondemand.com

Email: vladimir.forner@gmail.com

Password:
Authenticating...
OK

Targeted org 72e0e727trial

Select a space:
1. dev
2. fiori

Space (enter to skip): 1
Targeted space dev



API endpoint:   https://api.cf.eu10.hana.ondemand.com (API version: 3.86.0)
User:           vladimir.forner@gmail.com
Org:            72e0e727trial
Space:          dev
user: ratings-srv $
```

- Create the HANA service by running following commands:

```
$ cf create-service hanatrial hdi-shared ratings-srv-db
$ cds build --production
$ cf push -f gen/db
$ cf push -f gen/srv --random-route
```

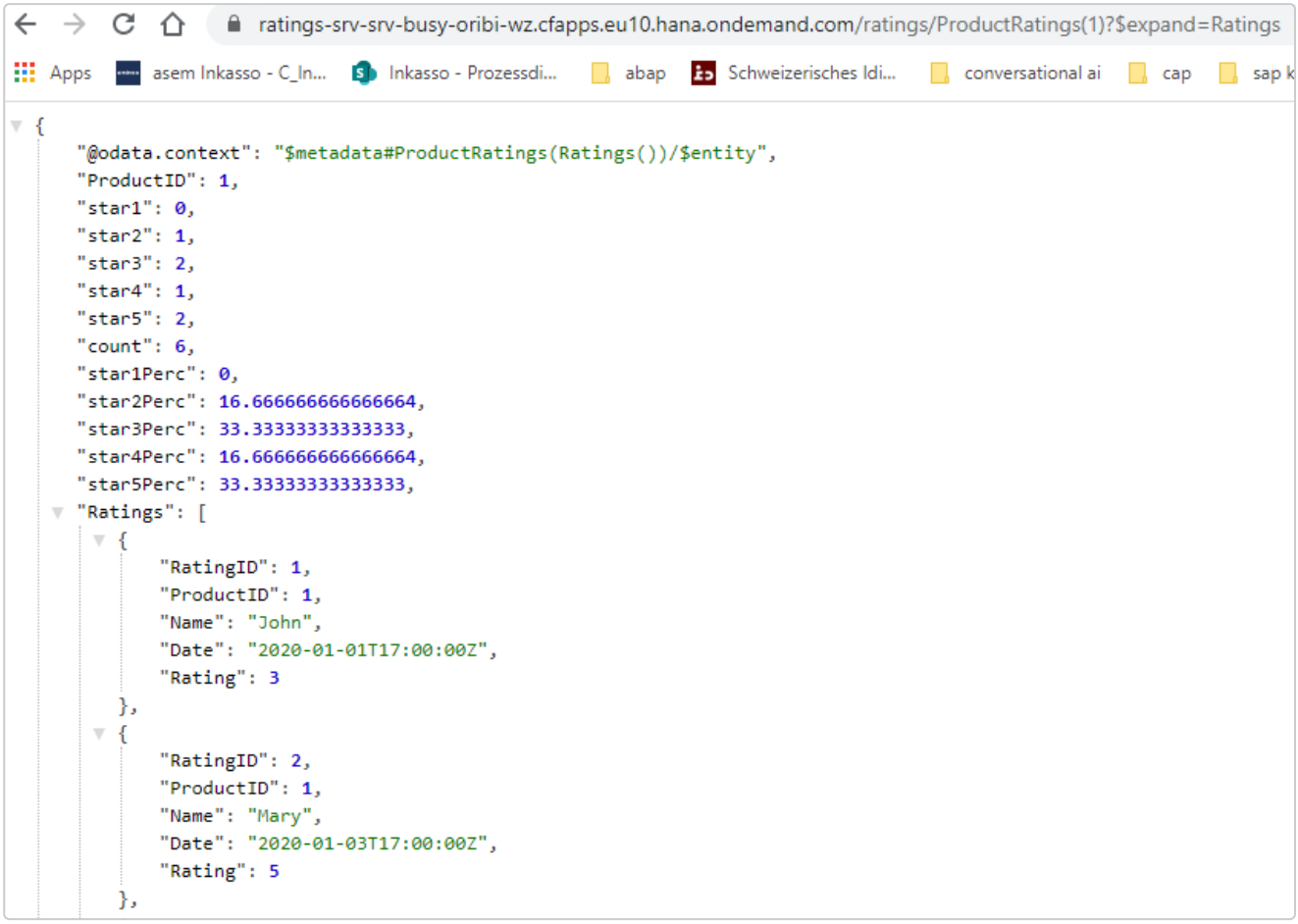
This will take a few minutes. After succesfull deployment you will get the address of your service.

```
Waiting for app to start...

name:               ratings-srv-srv
requested state:    started
isolation segment:  trial
routes:             ratings-srv-srv-busy-oribi-wz.cfapps.eu10.hana.ondemand.com
last uploaded:      Mon 19 Oct 13:06:33 UTC 2020
stack:              cflinuxfs3
buildpacks:         nodejs

type:           web
instances:      1/1
memory usage:   128M
start command:  npm start
     state     since                  cpu    memory          disk           details
#0   running   2020-10-19T13:07:03Z   0.0%   156K of 128M    255.8M of 1G
```

Now you can test the service in the browser.


ratings-srv-srv-busy-oribi-wz.cfapps.eu10.hana.ondemand.com/ratings/ProductRatings(1)?$expand=Ratings
Apps
asem Inkasso - C_In...
Inkasso - Prozessdi...
abap
Schweizerisches Idi...
conversational ai
cap
sap k
{
"@odata.context": "$metadata#ProductRatings(Ratings())/$entity",
"ProductID": 1,
"star1": 0,
"star2": 1,
"star3": 2,
"star4": 1,
"star5": 2,
"count": 6,
"star1Perc": 0,
"star2Perc": 16.666666666666664,
"star3Perc": 33.33333333333333,
"star4Perc": 16.666666666666664,
"star5Perc": 33.33333333333333,
"Ratings": [
{
"RatingID": 1,
"ProductID": 1,
"Name": "John",
"Date": "2020-01-01T17:00:00Z",
"Rating": 3
},
{
"RatingID": 2,
"ProductID": 1,
"Name": "Mary",
"Date": "2020-01-03T17:00:00Z",
"Rating": 5
},