ESus Technical Paper

Table of Contents

ESus Technical Paper *1*

1. Business Technology Platform (BTP) Trial Account *3*

2. BTP Services *3*

3. Configurations *5*

3.1 Trust Configuration **5**

3.2 Role Collections **5**

4. Front End: SAP Build Apps *6*

4.1 Logic & Formulas **6**

4.1.1 Logic 7

4.1.2 Formula 8

5. Back End Integration – SAP Business Application Studio *9*

1. Business Technology Platform (BTP) Trial Account

ESus was created using BTP which is SAP's Platform As A Service (PaaS). It provides the technology, tools, and services to assist businesses in developing, integrating, and managing their enterprise applications.

This technical paper outlines how the configurations and logic were done and we also provide guidance the back end integration if the recommended extensions to the app in the report were to be implemented

The first step is signing up for the BTP Trial Account. Initially the account lasts for about 30 days, however, frequent use of the account will automatically extend the account to 90 days. This worked well for us as our capstone project was well within this period.

The BTP cockpit shows you an overview of your Global account, from which you can then create a subaccount, choosing an AWS region to host your account. Usually with the free tier account, you only have an option to choose between EU or US and ideally you need to choose a region that is closest to your users. This reduces any latency issues and improves user experience. Ideally, if we would be deploying it in Australia but this isn't an option for the free tier account

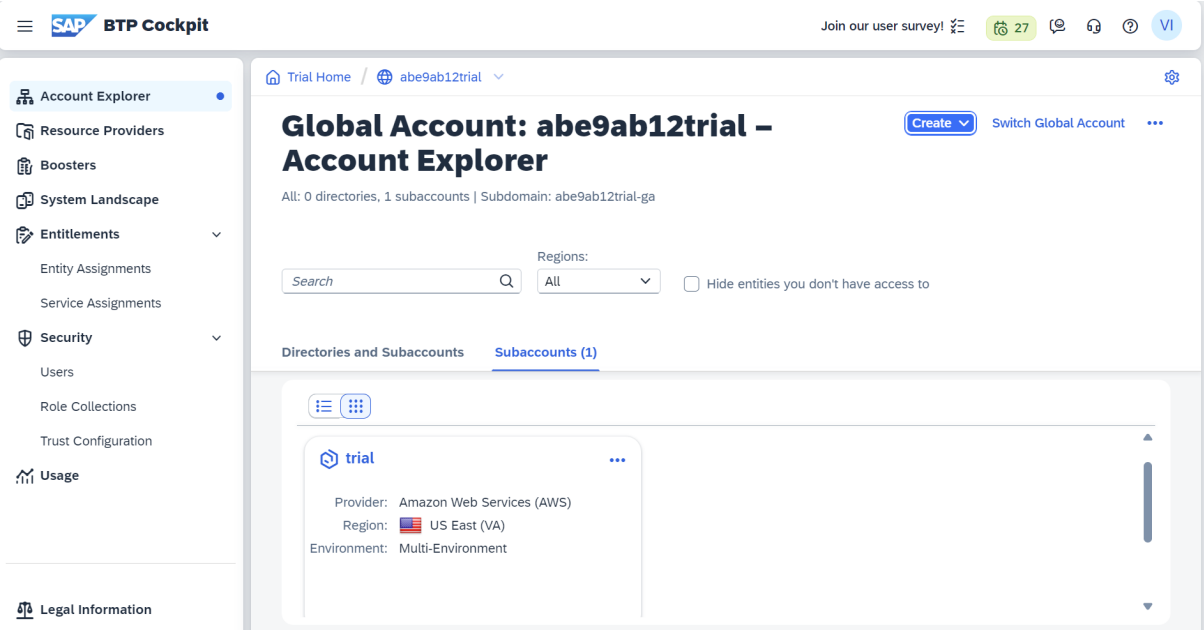

Figure 1: BTP Cockpit

2. BTP Services

Once our subaccount was created, we subscribed to the different services we required for building our application. The services that are available for use within the Trial environment can be found in the Service Marketplace and the below outlines the various services that we subscribed to:

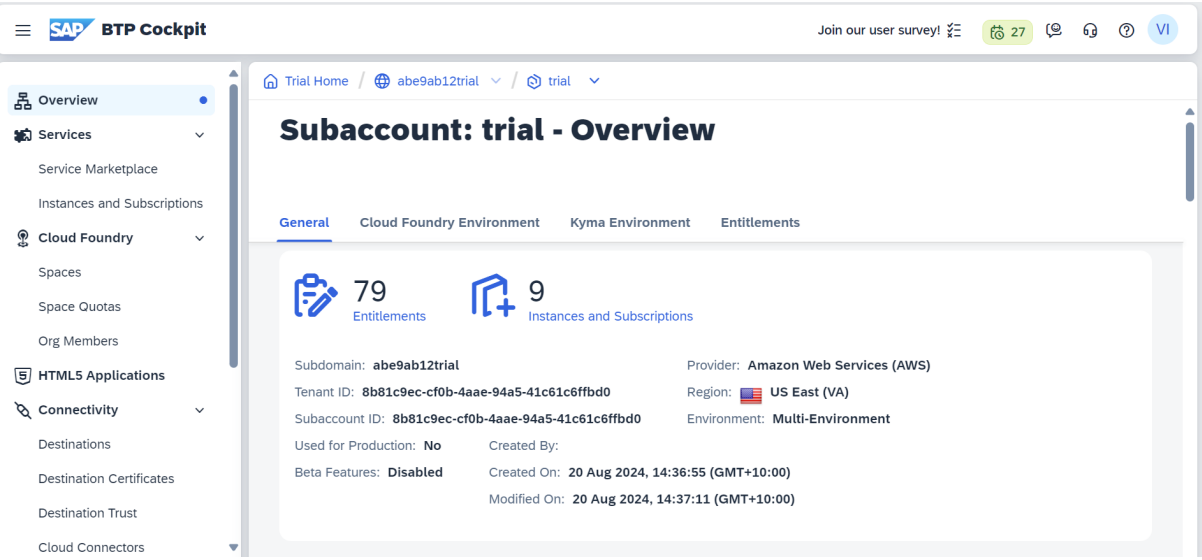

Figure 2: BTP Subaccount overview

- **Cloud Foundry instance:** This was an essential step, as it provided us with a runtime environment. We created a Cloud Foundry instance to provision our resources and the services that we required to develop our application.
- **SAP Build Apps:** A low code, no code platform for our Front-End Development
- **SAP Build Code:** Previously now as Business Application Studio, for our Back-End development.
- **Cloud Identity Services:** provides secure user authentication, identity management, and access control for applications hosted on SAP or other environments.

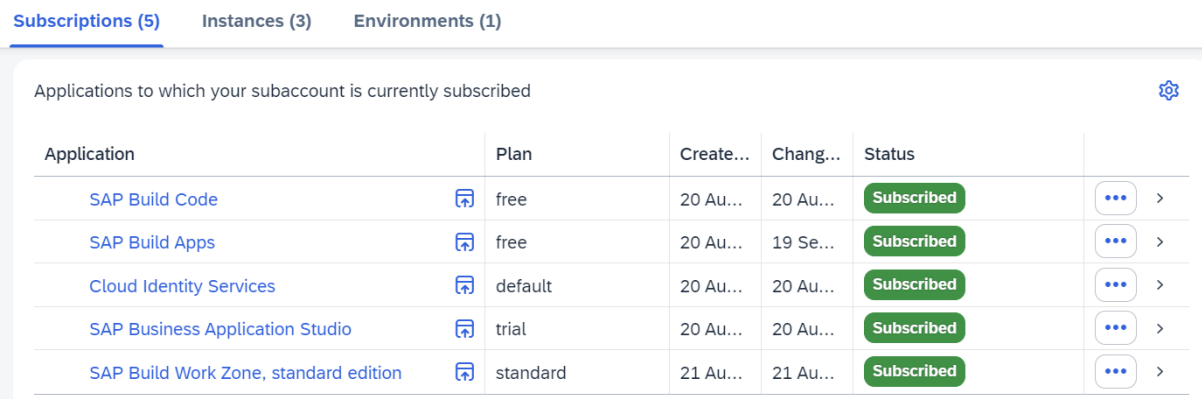

Figure 3: Service Subscriptions

- **Dev Space Manager:** Dev Space manager is a cloud-based development environment that provides a similar isolated workspace, much like what you would experience in a Virtual environment. This isolation allows the users to create multiple Dev spaces for specific development scenarios. For example, Fiori Apps, Cloud Application Programming (CAP) and HANA, etc.

3. Configurations

Below are some of the key configurations that were carried out to be able to utilise these services

3.1 Trust Configuration

Before we could run any service that we had subscribed to, we needed to establish trust. This is essential as it establishes a secure connection ***between*** the ***identity provider*** (Cloud Identity Services) and the ***services provider (BTP and other services).*** This ensures that the services provider can authenticate the users on behalf of the application, allowing Single Sign Ons (SSO) as the user navigates between these services without having to re-authenticate themselves each time.

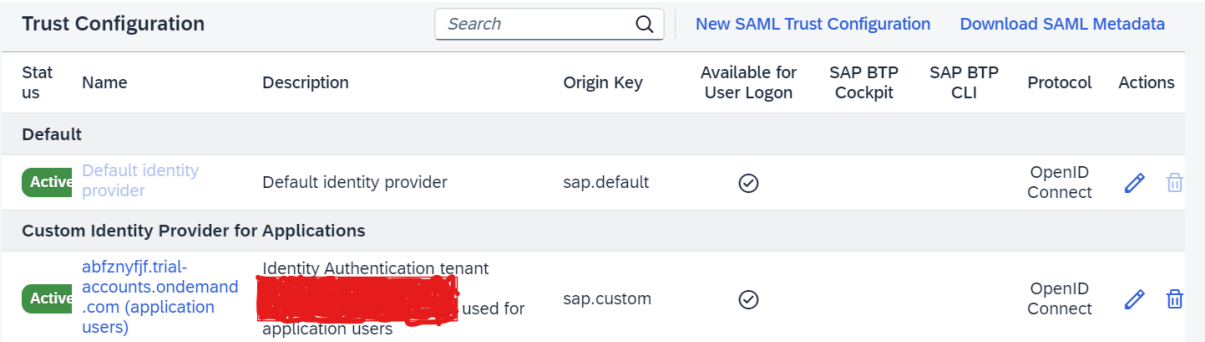

Figure 4: Trust Configuration

3.2 Role Collections

Once trust has been established, we moved on to the Role Collections. This is where we can group together multiple roles for efficient management of user access and granting them certain levels of permissions. While some roles a assigned to the creator of the subaccount by default, the roles and permissions for the different services needed to be configured accordingly.

For Business Application studio, we required various privileges, so we created a user and assigned several roles: developer, extension developer and various webIDE roles.

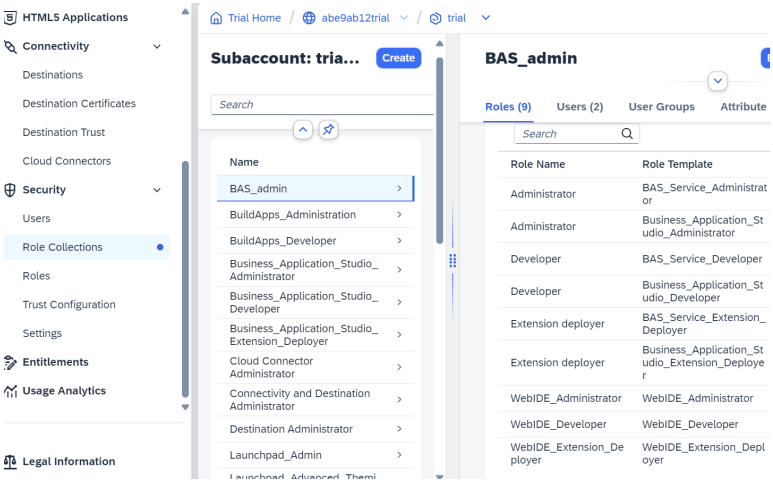

Figure 5: Role Collection for BAS

For Build Apps, there were two different collections created – the Developer and Administrator. The below

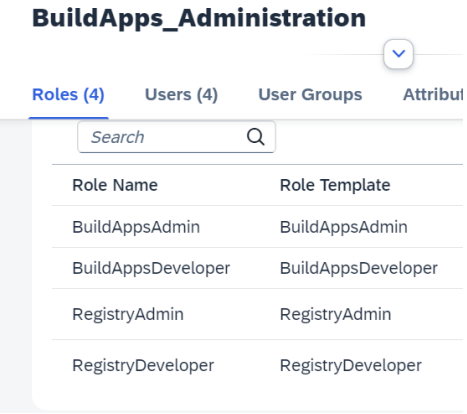

Figure 7: Build Apps Administration

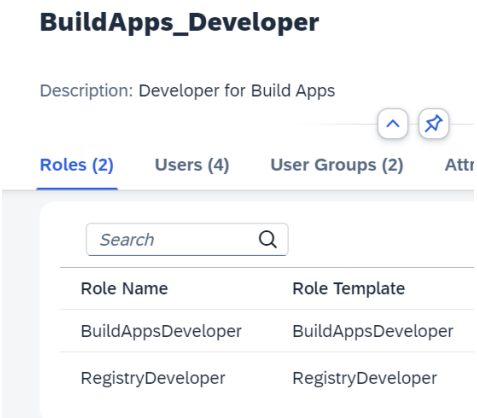

Figure 6: Build Apps Developer

4. Front End: SAP Build Apps

Once all the necessary configurations were done, we started building out the Front End using SAP Build Apps. The low-code, no-code platform provided drag and drop functionalities, allowing an easy flow throughout the design time referencing our prototype design done on Figma.

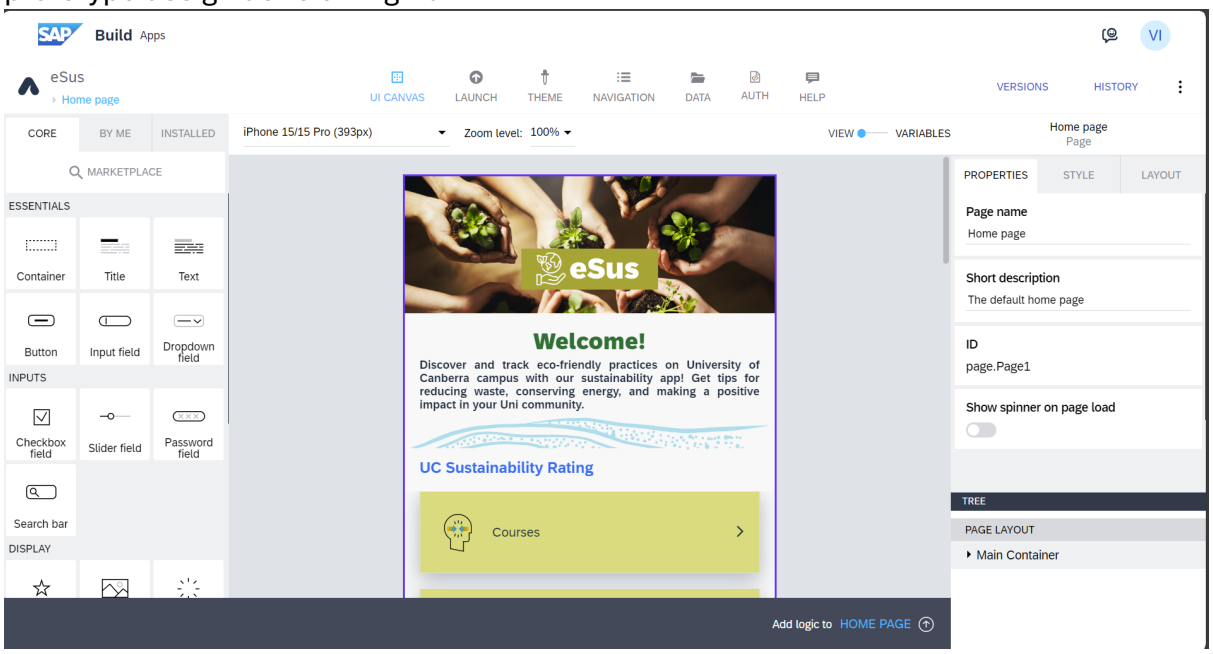

Figure 8: Home Page design

4.1 Logic & Formulas

Logic and Formulas describe the way in which the app should react or behave. For this paper, we will look at the Tips List as an example for the usage of both.

- We stored the list as an object under a Page Variable called DailyTips. There are various types of variables that are available depending on what they are used for. App variables are variables that can be accessed from anywhere within the app and Page variables can only be accessed within the page it has been set to. Therefore, in this case, the DailyTips list can only be accessed or called within the Home Page.

4.1.1 Logic

- The logic we used for this example was quite simple- we were only required to set the page variable to DailyTips

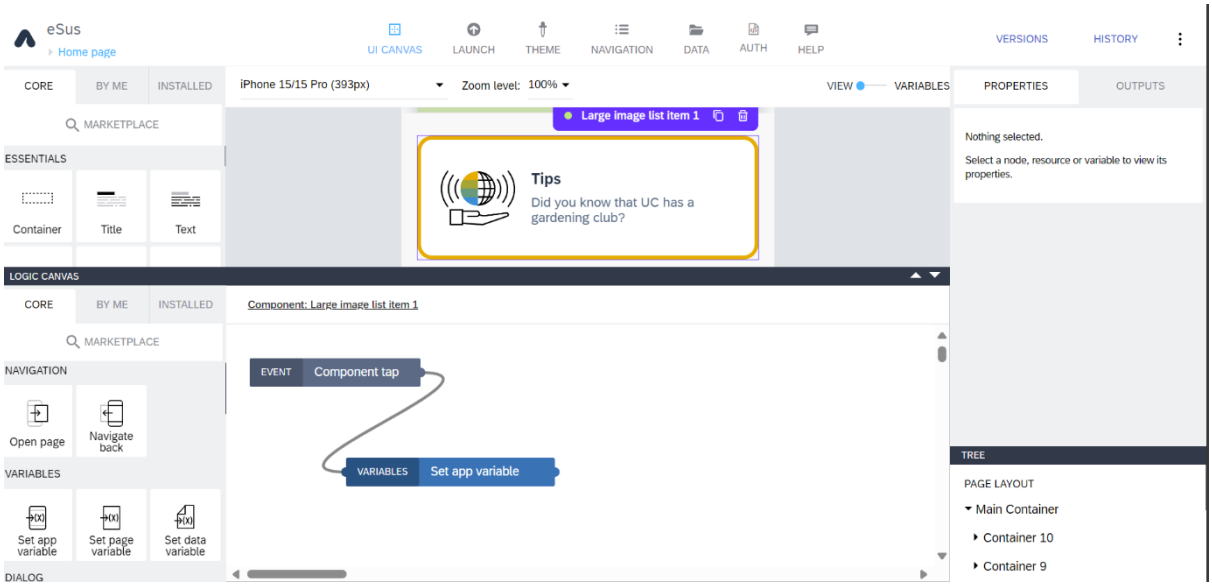

Figure 9: Set Variable Logic

4.1.2 Formula

Another aspect of the app which we used quite a lot was the formula. This required scripting to allow the app to react in a certain way.

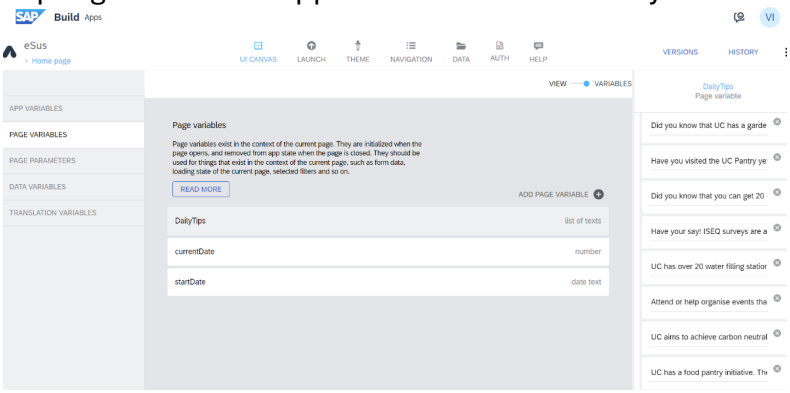

Figure 10: Daily Tips Variable

- Once this has been set, we need a UI component where we can call the variable. We used a Large List item which contains three different components which can be seen below. The *image* component, the *title* component, and the text component.

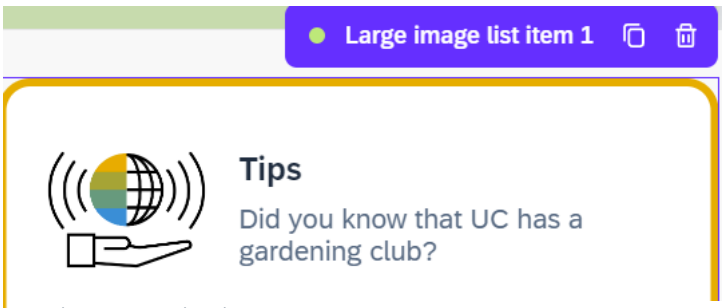

Figure 11: Daily Tips UI

- The image and the title are fixed, but we added a formula to bind to the text component so that every day, it iterates through the list.

```
pageVars.DailyTips[MOD(FLOOR(TIMESTAMP()/36000000/24), COUNT(pageVars.DailyTips))]
```

Figure 12: List iteration Formula

- The timestamp function retrieves the current timestamp in milliseconds using Unix epoch (since Jan 1, 1970). This divides the current timestamp by 10,000 to convert from milliseconds to seconds. It then divides the result by 24 to convert it from seconds to days. So basically, it gives us the number of days since Unix Epoch.
- The Floor function rounds the number to the nearest whole number, which gives us a complete number of days.
- The Count function will then count the number of tips available in the DailyTips array

- MOD function calculates the remainder of when the total number of complete days is divided by the number of daily tips.
- This step is important as it always ensures that the index is always within the number of items in the DailyTips array.
- So when we add to the list or remove, it will not break the code or the intended behaviour of Daily Tips functionality.

5. Back End Integration – SAP Business Application Studio

Based on the sponsor's requirements, there are no user inputs necessitating writing operations to the database. As a result, the backend primarily handles logic for reading from the database and has been set up for authentication.

In our report, we recommend that the sponsor consider adopting the Cloud Application Programming (CAP) framework. CAP is particularly suited for this project as it supports multiple programming languages and offers a model-driven approach, allowing developers to define data models and business logic using Core Data Services (CDS).

Additionally, CAP provides built-in features for security management, database access, and OData service creation, significantly streamlining the development process. A development space can be created with CAP extension tools integrated, allowing for an efficient, managed environment. The dev space acts as a virtual machine that maintains and organises the development setup.

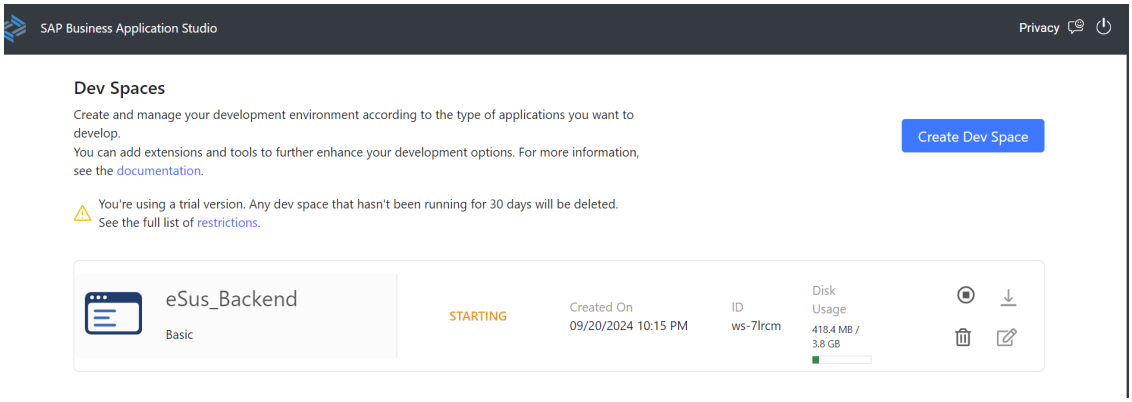

Figure 13: eSus Backend Dev Space

If the University opts to upload data via CSV files, a schema can be configured to read the data from the CSV files stored in our designated Data Folder. This data will then be reflected in the application, specifically across the various activities being rated. Additionally, a schema for carbon footprint measurements can also be incorporated, allowing for seamless integration of that data into the system.

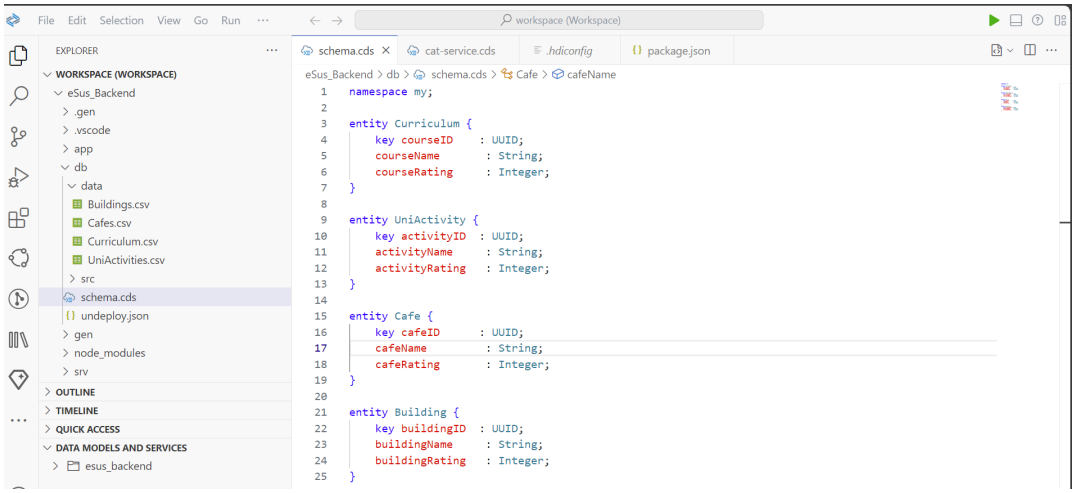

Figure 14: eSus schema

The first step is to deploy the backend on SAP Business Technology Platform (BTP). After deployment, the OData service should be exposed, enabling access. Once the service is exposed, the service URL can be retrieved, and the endpoint tested using Postman to ensure it is functioning as expected.

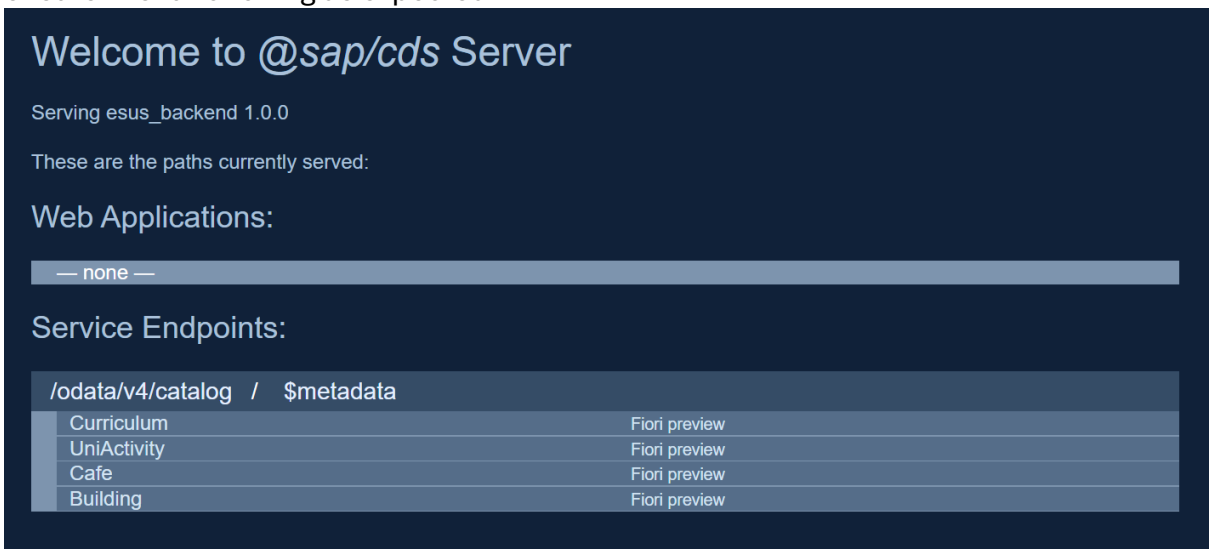

Figure 15: CDS server for views

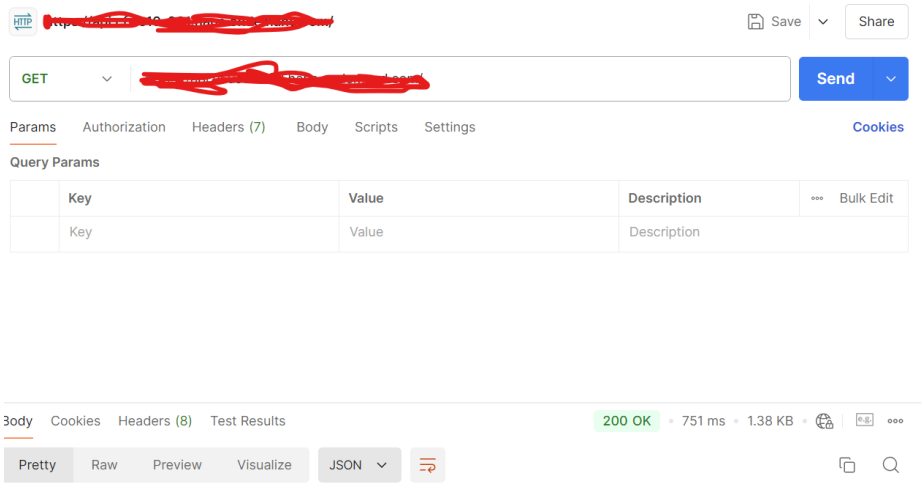

Figure 16: End Point Test on PostMan

After receiving a 200 OK response, then proceed to integrate the backend with the frontend using SAP Build Apps. This step ensures seamless communication between the two, enabling the application to function as intended.