Challenge Case Study

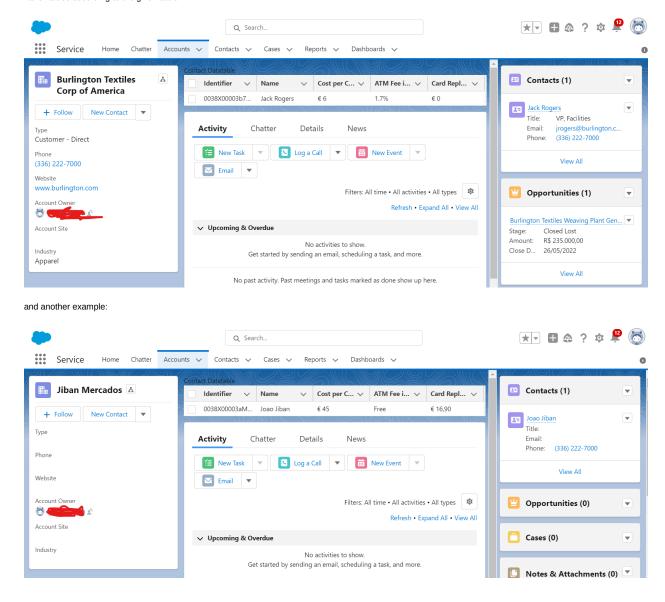
First, if you want access to the dev org to test, please email me and I grant you credentials.

All the detailed steps will be seen below:

1- Component to display tables with values.

The final output will be shown first and then I will explain how I did it.

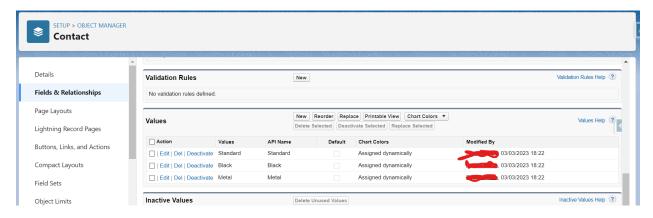
These are 2 accounts that can have 1 or more contacts. The Contact will, according to fields Product_c and Home_Country_c. These are picklist fields that can have values according to the given table.



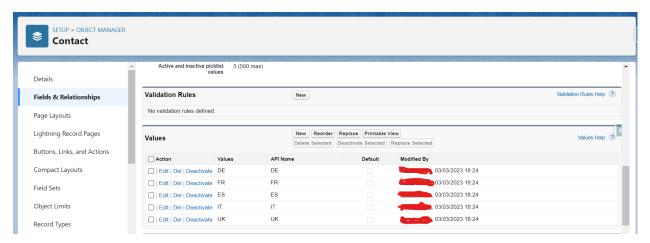
How was done:

1 - created fields Product__c and Home_Country__c. Both are picklists.

Product__c



Home_Country__c



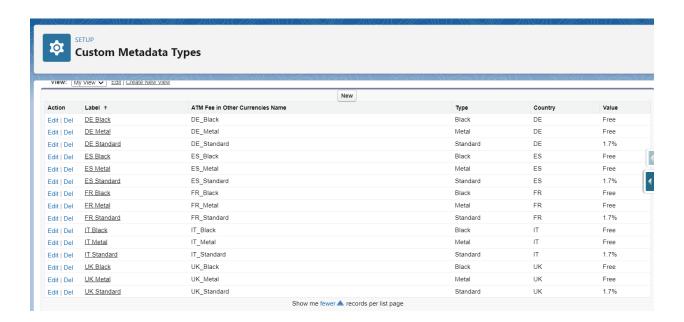
Metadata

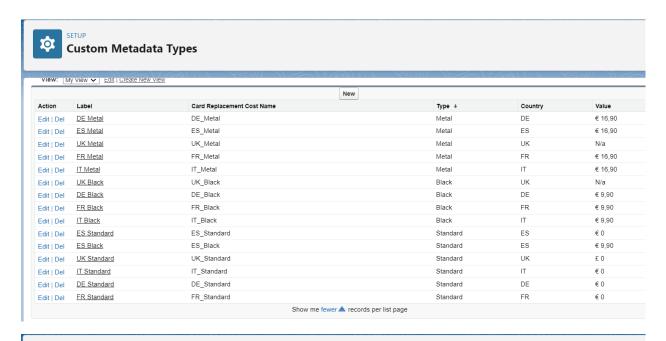
I decided to create 3 metadata objects to store the table:

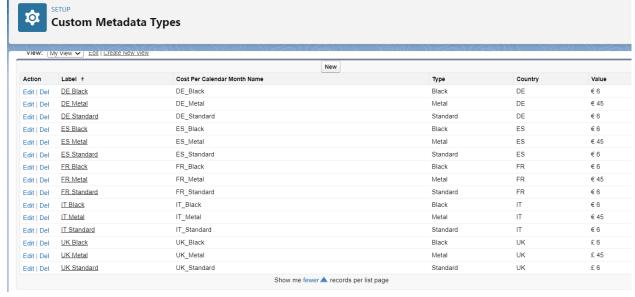
Cost per Calendar Month	DE	FR	ES	IT	UK
Standard	€ 0	€ 0	€ 0	€ 0	£ 0
Black	€ 9,90	€ 9,90	€ 9,90	€ 9,90	N/a
Metal	€ 16,90	€ 16,90	€ 16,90	€ 16,90	N/a
ATM Fee in other currencies	DE	FR	ES	IT	UK
Standard	1.7%	1.7%	1.7%	1.7%	1.7%
Black	Free	Free	Free	Free	Free
Metal	Free	Free	Free	Free	Free
Card Replacement Cost	DE	FR	ES	IT	UK
Standard	€ 6	€ 6	€ 6	€ 6	£ 6
Black	€ 6	€ 6	€ 6	€ 6	£ 6
Metal	€ 45	€ 45	€ 45	€ 45	£ 45

The reason to use metadata is that changing 1 value in one of its instances will automatically have different values when accessing an Account and having displayed the fees and values for each Contact. And 3 metadata objects is better cause you avoid having a single point of failure. To refresh the values, an Admin just need to access one of these 3 objects and replace values.

So, here is the result for the 3 metadatas:







APEX and LWC

To fetch an Account, its Contacts and these Metadatas, I created an Apex Class:

ContactFeeService:

Will receive the current Account Id, get its Contacts and for each of them, if fields Product_c and Home_Country_c are populated, then the Account page will display the data for its Id, Name and respective values in the table.

And we are using an internal MetaContactDetailsObject wrapper class to group the values and send to the LWC component.

Apex

ContactFeeService.cls

```
public with sharing class ContactFeeService {
    @auraEnabled(cacheable=true)
    public static List<MetaContactDetailsObject> getContactCostsByAccount(String accountId){
         List<MetaContactDetailsObject> records = new List<MetaContactDetailsObject>();
Contact contact = [SELECT Id, Name, Product_c, Home_Country_c FROM Contact WHERE AccountId = :accountId LIMIT 1];
MetaContactDetailsObject record = new MetaContactDetailsObject();
         record.Id = contact.Id;
record.name = contact.Name;
         record.calendar = getCalendarMonthValue(contact.Product_c, contact.Home_Country_c);
record.fee = getFeeinOtherCurrenciesValue(contact.Product_c, contact.Home_Country_c);
         record.card = getCardReplacementValue(contact.Product__c, contact.Home_Country__c);
         return records;
    //btw: querying against metadata without textArea fields won't bother to SOQL limits but only rows counting
    public static String getCalendarMonthValue(String product, String homeCountry){
    List<Cost_Per_Calendar_Month_mdt> records = [SELECT Id, Value_c FROM Cost_Per_Calendar_Month_mdt WHERE Type_c = :product AND Country_c = :homeCountry];
         return records[0].Value_c;
    public \ static \ String \ getFeeinOtherCurrenciesValue(String \ product, \ String \ homeCountry) \{
         List<ATM_Fee_in_Other_Currencies__mdt> records = [SELECT Id, Value_c FROM ATM_Fee_in_Other_Currencies__mdt WHERE Type_c = :product AND Country_c = :homeCountry];
         return records[0].Value_c;
    public static String getCardReplacementValue(String product, String homeCountry){
         List<Card_Replacement_Cost_mdt records = [SELECT Id, Value_c FROM Card_Replacement_Cost_mdt WHERE Type_c = :product AND Country_c = :homeCountry]; return records[0].Value_c;
    public class MetaContactDetailsObject{
         public String Id;
          @AuraEnabled
         public String name:
          @auraEnabled
         public String calendar;
         @auraEnabled
public String fee;
         @auraEnabled
         public String card;
   }
```

LWC

We will have to fetch the data and to display the following files:

contactFees.js

```
import \ \{ \ Lightning Element, \ api, \ wire, \ track \ \} \ from \ 'lwc'; \\ import \ getContactCostsByAccount \ from \ '@salesforce/apex/ContactFeeService.getContactCostsByAccount'; \\ import \ getContactCostsByAccount \ from \ '@salesforce/apex/ContactFeeService.getContactCostsByAccount'; \\ import \ getContactCostsByAccount \ from \ 'gsalesforce/apex/ContactFeeService.getContactCostsByAccount'; \\ import \ getContactCostsByAccount \ getContactCostsByAccount \ getContactCostsByAccount'; \\ import \ getContactCostsByAccount \ getContactCostsByAccount \ getContactCostsByAccount \ getContactCostsByAccount \ getContactCostsByAccount \ getContactCostsByAccount \ getCostsByAccount \ getC
 export default class ContactFees extends LightningElement {
                   @api recordId:
                   contact:
                   @track columns = [{
                                          label: 'Identifier',
                                         fieldName: 'Id',
                                         type: 'text',
                                       sortable: true
                                       label: 'Name',
                                         fieldName: 'name',
                                         type: 'text',
                                          sortable: true
                                       label: 'Cost per Calendar Month',
                                       fieldName: 'calendar',
type: 'text',
                                         sortable: true
                                       label: 'ATM Fee in other currencies', fieldName: 'fee',
                                       sortable: true
                                         label: 'Card Replacement Cost',
```

```
fieldName: 'card',
    type: 'text',
    sortable: true
}

@wire(getContactCostsByAccount, {accountId: '$recordId'})
wireContact((error, data)){
    if(data){
        this.contact = data;
        this.error = undefined;
    }else{
        this.contact = undefined;
        this.error = error;
    }
}
```

contactFees.html

PART 2

Rest Controller

I have developed class ApexQueryAccountDetailsRestController as following.

It used the previous constructed class in part 1 and you need to pass the Account's Id on the URL.

 $ApexQueryAccountDetailsRestController\ .cls$

```
@RestResource(urlMapping='/Account/*')
global with sharing class ApexQueryAccountDetailsRestController {
    @HttpGet
    global static String doGet(){
        RestRequest req = RestContext.request;
        RestResponse res = RestContext.response;
        String accountId = req.requestURI.blstring(req.requestURI.lastIndexOf('/') + 1);
        List<ContactFeeService.MetaContactDetailsObject> result = ContactFeeService.getContactCostsByAccount(accountId);
        return JSON.serialize(result);
}
```

The output will be a JSON:

 $|[\{"name":"Jack\ Rogers","Id":"0038X00003b77aLQAQ","fee":"1.7%","card":"\\ \in 0","calendar":"\\ \in 6"\}]$