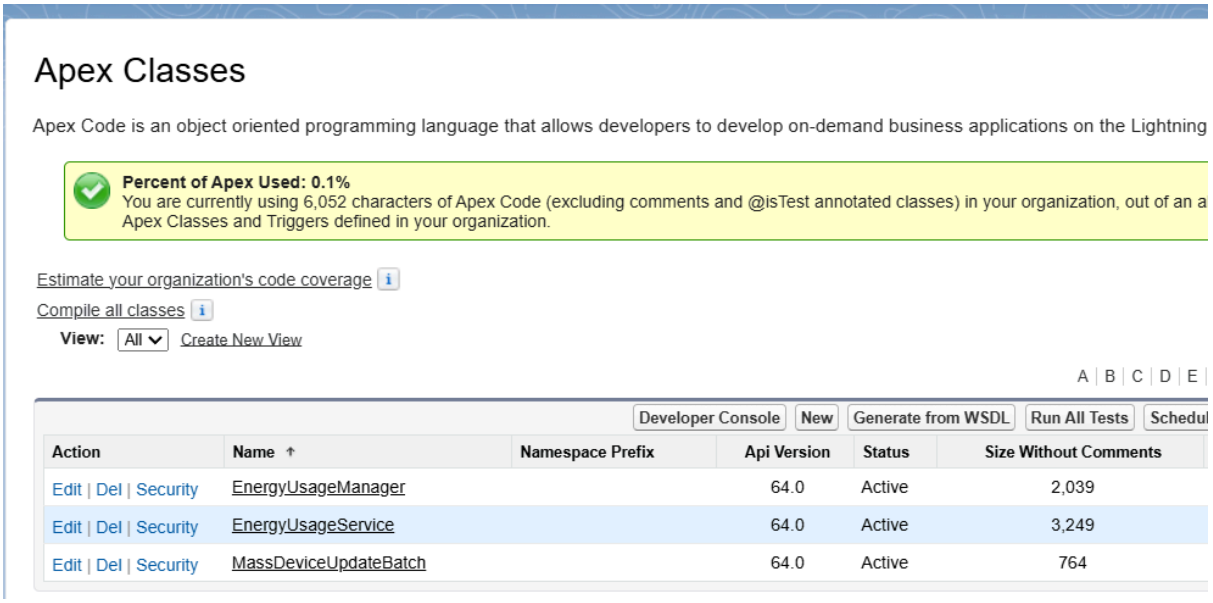
Phase 5 Report – Apex Trigger Implementation

1. Objective of Phase 5

The objective of Phase 5 is to automate and enforce business rules for the Smart Energy Management System using Apex triggers and handler classes. This implementation focuses on three key objects within the system:

- Energy Usage Record (Energy_Usage__c)
- Energy Device (Energy_Device__c)
- Personnel (Personnel__c)

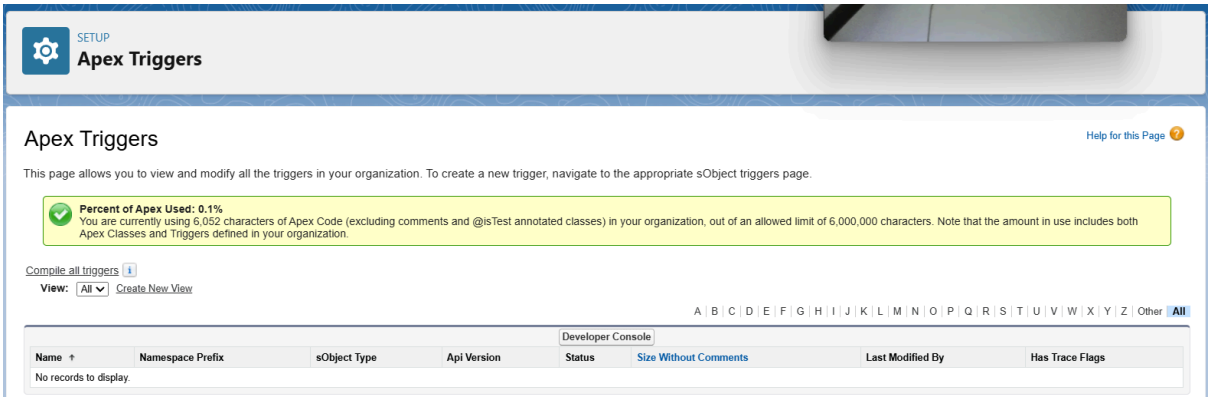
The outcome is increased data integrity, streamlined automation, and enhanced operational efficiency for the main business processes.



2. Trigger Design Pattern

A structured approach was applied for trigger development to ensure maintainability and future-proofing:

- Only one trigger is created for each object, simplifying management and reducing potential errors.
- Triggers themselves do not contain business logic. Instead, all logic is placed within handler classes, which act as dedicated containers for automation rules.
- Logic centralization within handler classes improves code reusability, readability, and makes testing and debugging more efficient.



3. Implementation Steps

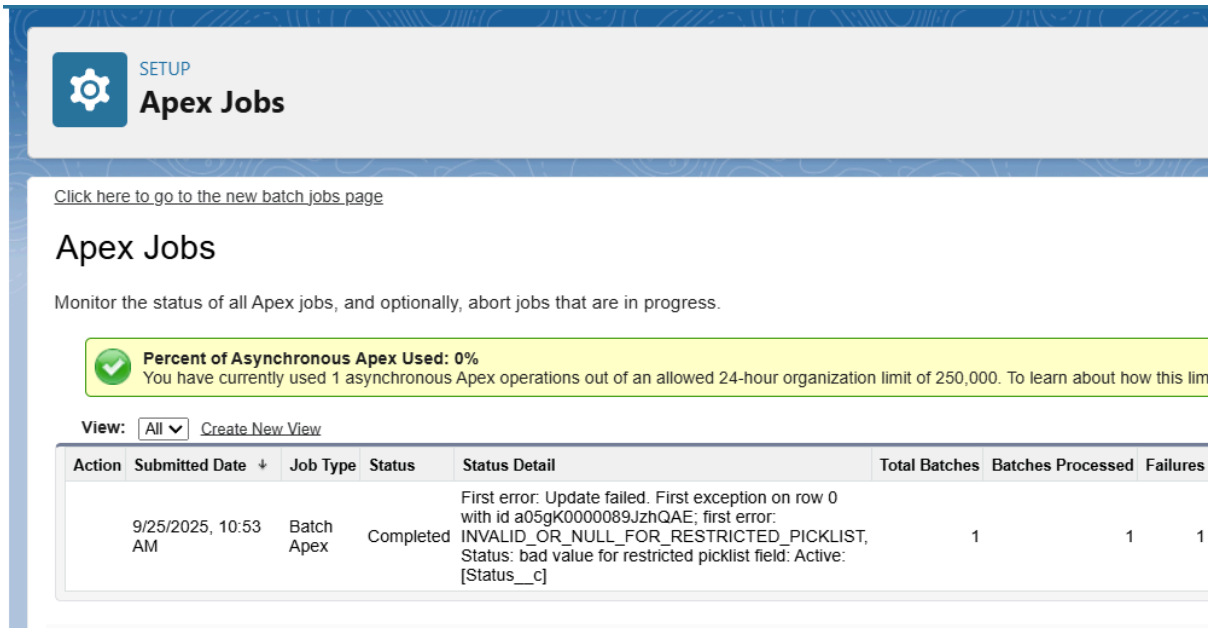
- The implementation began by defining and creating trigger files for each major object in the system, specifically assigning contexts for object events such as insertions, updates, and deletions.
- Corresponding handler classes were developed to handle all business rules and automation. These handler classes covered validation, pre-deletion logic, post-insertion automation, post-update automation, and post-deletion actions for each object.
- Metadata files were created for each Apex class to ensure smooth deployment and integration with the Salesforce platform. Proper file organization and compliance with Salesforce standards were maintained throughout.

```
> main > default > classes > MassDeviceUpdateBatch.cls > MassDeviceUpdateBatch > execute(Database.BatchableContext, List<
lobal class MassDeviceUpdateBatch implements Database.Batchable<SObject> {

    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator(
            'SELECT Id, Status__c FROM Energy_Device__c WHERE Status__c = NULL'
        );
    }

    global void execute(Database.BatchableContext bc, List<SObject> scope) {
        List<Energy_Device__c> devicesToUpdate = new List<Energy_Device__c>();
        for (SObject s : scope) {
            Energy_Device__c d = (Energy_Device__c)s;
            d.Status__c = 'Active';
            devicesToUpdate.add(d);
        }
        update devicesToUpdate;
    }
3   TERMINAL   OUTPUT   PORTS
```

4. Deployment Steps

- All necessary files and classes were organized within the Salesforce project structure using VS Code and Salesforce CLI tools.

- Deployment was executed systematically, and the existence of correct triggers was verified within Salesforce Setup by checking their appearance in the Object Manager section for each relevant object.



5. Expansion for Other Objects

This design pattern is extensible. The same structure and approach were applied for the two other main objects:

- Energy Device (Energy_Device__c)
- Personnel (Personnel__c)

The pattern ensures consistent automation, easy future rule additions, and system scalability.

6. Benefits of This Approach

- Code remains clean and concise with separation of trigger and business logic.
- Handler methods are easily reusable, which supports future Salesforce expansions and reduces development time for further automation.
- Testing is simplified, making it more efficient to verify that rules operate as expected.
- The overall system architecture is easier to maintain and is flexible for accommodating future business requirements and process changes.