Phase 7: Integration & External Access

This phase focused on **Integration and External Access**, a critical capability of the Salesforce platform. Integration allows the **RetailHub CRM** to act as a central hub, communicating in real-time with external systems. The phase covered the architecture required for both **inbound** (receiving data) and **outbound** (sending data) communication patterns.

7.1 Implemented Feature: Inbound Purchase API

To connect the RetailHub CRM with an **online e-commerce platform**, an inbound web service was implemented using **Apex REST**.

Purpose:

- Allows an external website to automatically submit new purchase information into Salesforce after a customer completes online checkout.

Implementation:

- An Apex class named `PurchaseAPI` was created with the `@RestResource` annotation, exposing a **custom REST API endpoint**.
- The endpoint accepts a **JSON payload** containing customer and order details.
- Logic includes:
    - Parsing the incoming JSON payload
    - Performing **upsert logic** to find an existing `Customer__c` or create a new one based on the email address
    - Creating corresponding `Purchase__c` and `Purchase_Line_Item__c` records

Testing:

- The API was tested using the **Workbench REST client**, which simulated an external system sending JSON requests.

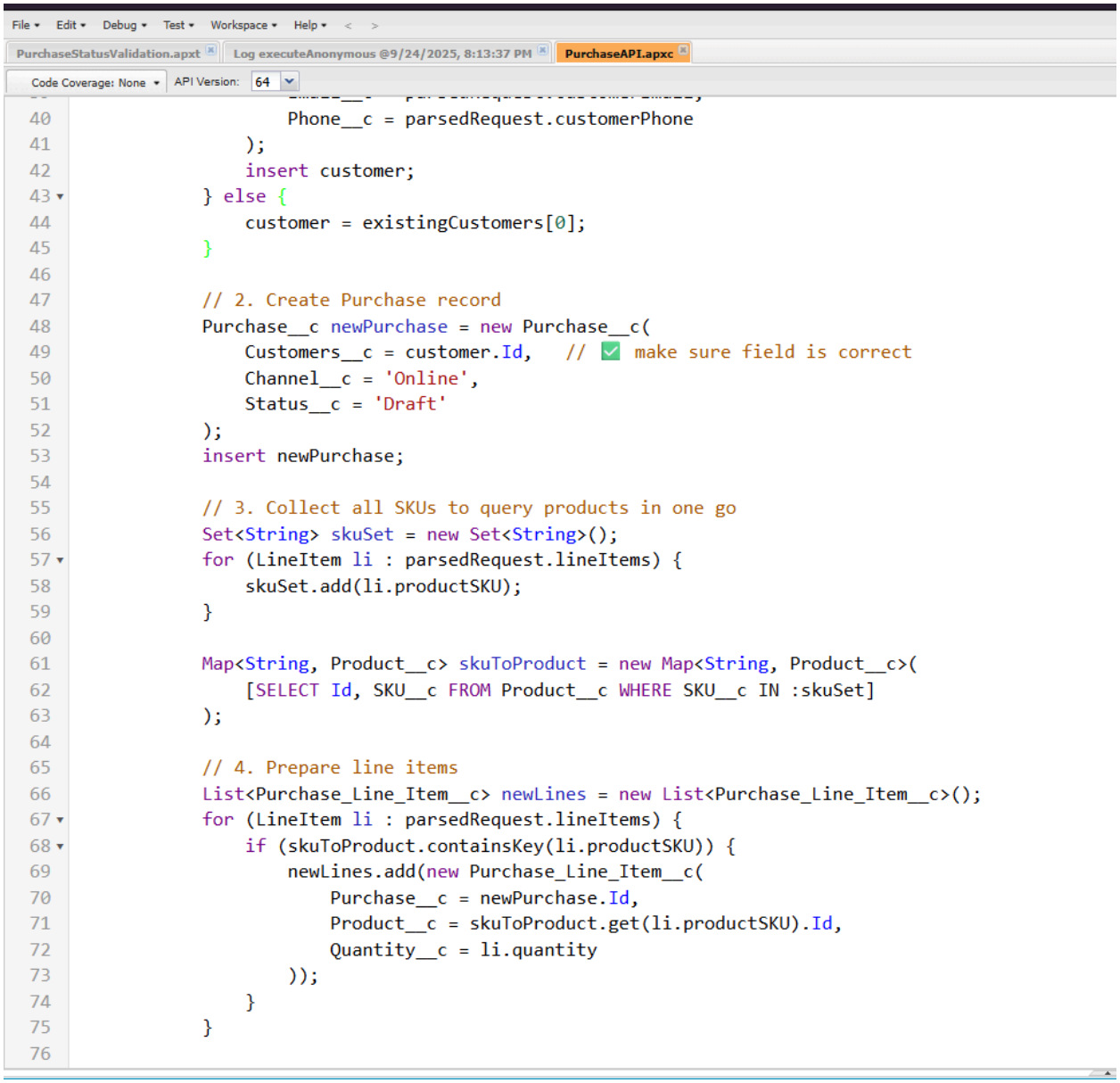
- The test confirmed that the Apex service successfully received the data and created all necessary records in the CRM.

File Edit Debug Test Workspace Help

PurchaseStatusValidation.apxt | Log executeAnonymous @9/24/2025, 8:13:37 PM | PurchaseAPI.apxc

Code Coverage: None | API Version: 64

```apex
@RestResource(urlMapping='/Purchase/*')
global with sharing class PurchaseAPI {

    // Inner DTO classes for JSON parsing
    global class PurchaseRequest {
        public String customerName;
        public String customerEmail;
        public String customerPhone;
        public List<LineItem> lineItems;
    }

    global class LineItem {
        public String productSKU;
        public Integer quantity;
    }

    @HttpPost
    global static String createPurchase() {
        try {
            // Get request body
            RestRequest req = RestContext.request;
            String requestBody = req.requestBody.toString();

            // Parse JSON into Apex object
            PurchaseRequest parsedRequest =
                (PurchaseRequest) JSON.deserialize(requestBody, PurchaseRequest.class);

            // 1. Find or create the Customer
            List<Customer__c> existingCustomers = [
                SELECT Id
                FROM Customer__c
                WHERE Email__c = :parsedRequest.customerEmail
                LIMIT 1
            ];
            Customer__c customer;
            if (existingCustomers.isEmpty()) {
                customer = new Customer__c(
```


File Edit Debug Test Workspace Help
PurchaseStatusValidation.apxt
Log executeAnonymous @9/24/2025, 8:13:37 PM
PurchaseAPI.apxc
Code Coverage: None
API Version: 64
Phone__c = parsedRequest.customerPhone
);
insert customer;
} else {
customer = existingCustomers[0];
}
// 2. Create Purchase record
Purchase__c newPurchase = new Purchase__c(
Customers__c = customer.Id, // ✅ make sure field is correct
Channel__c = 'Online',
Status__c = 'Draft'
);
insert newPurchase;
// 3. Collect all SKUs to query products in one go
Set<String> skuSet = new Set<String>();
for (LineItem li : parsedRequest.lineItems) {
skuSet.add(li.productSKU);
}
Map<String, Product__c> skuToProduct = new Map<String, Product__c>(
[SELECT Id, SKU__c FROM Product__c WHERE SKU__c IN :skuSet]
);
// 4. Prepare line items
List<Purchase_Line_Item__c> newLines = new List<Purchase_Line_Item__c>();
for (LineItem li : parsedRequest.lineItems) {
if (skuToProduct.containsKey(li.productSKU)) {
newLines.add(new Purchase_Line_Item__c(
Purchase__c = newPurchase.Id,
Product__c = skuToProduct.get(li.productSKU).Id,
Quantity__c = li.quantity
));
}
}

```
                }
            }

            if (!newLines.isEmpty()) {
                insert newLines;
            }

            return 'Success! Created Purchase with ID: ' + newPurchase.Id;

        } catch (Exception e) {
            // Return error in JSON format
            RestResponse res = RestContext.response;
            res.statusCode = 400;
            return 'Error: ' + e.getMessage();
        }
    }
}
```

7.2 Future Enhancements & Other Integration Concepts

While not fully implemented in the current version, the project plan accounted for additional integration patterns to enhance scalability and functionality:

1. Outbound Notifications (Callouts):

- To send automated **"Thank You" SMS messages**, an outbound Apex callout can be implemented.
- This involves connecting to a third-party SMS provider (e.g., Twilio) via API.
- **Security:** A **Named Credential** would store endpoint URLs and authentication details, keeping sensitive information out of code.

2. Event-Driven Architecture (Platform Events):

- For a **scalable and decoupled architecture**, Platform Events can be used.
- Example: The main "RetailHub Flow" could fire a **"Purchase Completed"** platform event.
- Other internal or external systems could subscribe to this event to trigger independent processes, such as notifying a **shipping and fulfillment system**, without tightly coupling

to the core CRM logic.

Benefits of Integration & External Access

- Enables **real-time data synchronization** between Salesforce and external platforms
- Supports automated **post-purchase processes** such as notifications and order fulfillment
- Provides a foundation for **future enhancements** using modern integration patterns
- Ensures **secure, scalable, and maintainable** communication between systems