Ejercicio 6. Ajustes en el Registro

Desarrollaremos el siguiente requisito:

3. Al registrar la recepción del pedido de compra:
    - Satisfactorio: se registra la recepción de forma normal
    - No Satisfactorio: se registra la recepción de forma normal y se realizará un ajuste negativo de las cantidades para darlas de baja del stock

En este ejercicio aprenderemos:

- A utilizar Enums
- A desarrollar y ejecutar Tests para nuestros desarrollos
- A utilizar el nuevo comando **AL: Find event**

Indicaciones

Para completar el ejercicio realiza las siguientes acciones:

1. Crea una nueva rama de Git
2. Haz los tests necesarios para asegurar que se cumplen los requisitos
3. Haz el desarrollo necesario para realizar un ajuste negativo cuando el producto recibido ha sido No Satisfactorio
4. Haz un merge de tu rama de desarrollo con la rama principal
5. Elimina la rama del desarrollo

Indicaciones paso a paso

Para completar el ejercicio sigue los siguientes pasos en Visual Studio Code:

1. Crea una nueva rama de Git llamada *Feature_PostingAdjustments*
2. Desarrollo de tests
    1. En la extensión de test, añade el siguiente procedimiento a la codeunit de test

```
[Test]
procedure PurchaseOrderPostingP007();
var
    PurchaseHeader: Record "Purchase Header";
    PurchaseLine1: Record "Purchase Line";
    PurchaseLine2: Record "Purchase Line";
    Item1: Record Item;
    Item2: Record Item;
    ItemLedgerEntry: Record "Item Ledger Entry";
    QualityControlLibrary: Codeunit "Clip Quality Control - Library";
    PurchaseLibrary: Codeunit "Library - Purchase";
    PurchaseDocumentType: Enum "Purchase Document Type";
    PostedDocumentNo: Code[20];
    ReceiptILEEntryNo: Integer;
begin
```

```
    // [Scenario] If an Item line is non-satisfactory, the system will
post the reception and a negative adjustment for the same quantity

    // [Given] Setup: Two items that requires Quality Control
    //                A Purchase Order that uses both items
    //                Quality Control information set on the order
    //                      - Satisfactory for the first item
    //                      - Non-Satisfactory for the second item
    QualityControlLibrary.CreateItemWithQualityMeasures(Item1);
    QualityControlLibrary.CreateItemWithQualityMeasures(Item2);

    PurchaseLibrary.CreatePurchHeader(PurchaseHeader,
PurchaseDocumentType::Order.AsInteger(), '');
    PurchaseLibrary.CreatePurchaseLine(PurchaseLine1, PurchaseHeader,
PurchaseLine1.Type::Item.AsInteger(), Item1."No.", 10);
    PurchaseLibrary.CreatePurchaseLine(PurchaseLine2, PurchaseHeader,
PurchaseLine2.Type::Item.AsInteger(), Item2."No.", 20);


QualityControlLibrary.SetQualityControlInformationOnPurchaseLine(Purcha
seLine1, PurchaseLine1."Clip Quality Control Result"::Satisfactory);

QualityControlLibrary.SetQualityControlInformationOnPurchaseLine(Purcha
seLine2, PurchaseLine2."Clip Quality Control Result"::"Non-
Satisfactory");

    // [When] Exercise: Post the reception
    PostedDocumentNo :=
PurchaseLibrary.PostPurchaseDocument(PurchaseHeader, true, false);

    // [Then] Verify: The first item has no negative adjustment. The
second item has a negative adjustment
    ItemLedgerEntry.SetRange("Document No.", PostedDocumentNo);
    ItemLedgerEntry.SetRange("Item No.", PurchaseLine1."No.");
    AssertThat.AreEqual(1, ItemLedgerEntry.Count(), 'C01.P007.A001
First item ItemLedgerEntry.Count()');

    ItemLedgerEntry.SetRange("Document No.", PostedDocumentNo);
    ItemLedgerEntry.SetRange("Item No.", PurchaseLine2."No.");
    AssertThat.AreEqual(2, ItemLedgerEntry.Count(), 'C01.P007.A011
Second item ItemLedgerEntry.Count()');
    ItemLedgerEntry.FindFirst();
    AssertThat.AreEqual(ItemLedgerEntry."Entry Type"::Purchase,
ItemLedgerEntry."Entry Type", 'C01.P007.A012 Second item
ItemLedgerEntry."Entry Type"');

    ReceiptILEEntryNo := ItemLedgerEntry."Entry No.";
    ItemLedgerEntry.FindLast();
    AssertThat.AreEqual(ItemLedgerEntry."Entry Type"::"Negative
Adjmt.", ItemLedgerEntry."Entry Type", 'C01.P007.A021 Second item
ItemLedgerEntry."Entry Type"');
    AssertThat.AreEqual(-PurchaseLine2.Quantity,
ItemLedgerEntry.Quantity, 'C01.P007.A022 Second item
ItemLedgerEntry.Quantity');
```

```
    AssertThat.AreEqual(ReceiptILEEntryNo, ItemLedgerEntry."Applies-to
Entry", 'C01.P007.A023 Second item ItemLedgerEntry."Applies-to
Entry"');
end;
```

2. Publica y ejecuta los tests para comprobar que este último falla

3. Registro de un Ajuste Negativo cuando el producto es No Satisfactorio
    1. En la codeunit 50102 "Clip Purchase Post", utiliza el comando **AL: Find Event** para suscribirte al evento *OnAfterPostItemJnlLine* y escribe el siguiente código:

```
[EventSubscriber(ObjectType::Codeunit, Codeunit::"Purch.-Post",
'OnAfterPostItemJnlLine', '', false, false)]
local procedure OnAfterPostItemJnlLine(var ItemJournalLine: Record
"Item Journal Line"; var PurchaseLine: Record "Purchase Line"; var
PurchaseHeader: Record "Purchase Header"; var ItemJnlPostLine: Codeunit
"Item Jnl.-Post Line");
var
    ItemJnlLine: Record "Item Journal Line";
begin
    case PurchaseLine."Clip Quality Control Result" of
        PurchaseLine."Clip Quality Control Result"::" ",
PurchaseLine."Clip Quality Control Result"::Satisfactory:
            ;// Do nothing
        PurchaseLine."Clip Quality Control Result"::"Non-Satisfactory":
            begin
                ItemJnlLine := ItemJournalLine;
                ItemJnlLine."Entry Type" := ItemJnlLine."Entry
Type"::"Negative Adjmt.";
                ItemJnlLine."Item Shpt. Entry No." := 0;
                ItemJnlLine."Applies-to Entry" := ItemJournalLine."Item
Shpt. Entry No.";
                ItemJnlPostLine.RunWithCheck(ItemJnlLine);
            end;
    end;
end;
```

    2. Publica el desarrollo
    3. Ejecuta los tests para comprobar que el desarrollo realizado cumple con los tests planteados
4. Haz un merge de tu rama de desarrollo con la rama principal
5. Elimina la rama de desarrollo