Ejercicio 8. Interfícies

Nos llega una ampliación de los requisitos para la funcionalidad que estamos desarrollando que dice:

```
En ocasiones los productos recibidos no se pueden calificar inmediatemente como
Satisfactorios o No Satisfactorios, sino que requieren de pruebas adicionales para
determinar el resultado del Control de Calidad.

En estos casos se debe poder realizar la recepción de los productos, pero estos no
deben estar disponibles a la venta, de modo que serán traspasados al almacén de
CALIDAD, donde los usuarios realizarán pruebas adicionales y una vez terminadas
manualmente realizarán un Ajuste Negativo si el resultado es "No Satisfactorio" o
un traspaso al almacén general si el resultado es "Satisfactorio".

Para ello se requieren los siguientes desarrollos adicionales:
    - El usuario debe poder establecer la valoración general del producto como
"Pendiente"
    - Cuando la valoración es "Pendiente", se puede registrar la recepción de
compra aunque no se haya rellenado la información del control de calidad
    - Cuando la valoración es "Pendiente", se realizará la recepción de forma
normal y posteriormente debe transferirse al almacén de CALIDAD
    - Cuando la valoración es "Pendiente", no puede incluirse el producto en una
Factura de Compra

Aunque está pendiente de definir por parte de los usuarios, en el futuro se
querrán ampliar las posibilidades del Control de Calidad con almenos las
siguientes opciones, y posiblemente con más:
* Aceptable
* No Satisfactorio - Destruir
* No Satisfactorio - Devolver
* No Satisfactorio - Procesar
```

Para cumplir con todos estos requisitos utilizaremos un nuevo concepto de desarrollo introducido en Business Central 2020 Release Wave 1 (BC16): las interfícies.

En este ejercicio aprenderemos:

- A definir Interfícies en objetos de tipo Enum
- A ejecutar Tests para asegurar que un refactoring de código no altera el resultado desde un punto de vista funcional
- A implementar Interfícies en objetos de tipo Enum

Indicaciones

Para completar el ejercicio realiza las siguientes acciones:

1. Crea una nueva rama de Git
2. Modifica los desarrollos utilizados hasta el momento para utilizar interfícies

3. Ejecuta los tests desarrollados hasta el momento para asegurar que los requisitos funcionales se siguen cumpliendo aun habiendo hecho una refactorización del código
4. Crea el valor "Pendiente" en el Enum "Clip Quality Control Result"
5. Haz los tests necesarios para asegurar que se cumplen los requisitos del valor "Pendiente"
6. Haz los desarrollos necesarios para cumplir con los requisitos del valor "Pendiente"
7. Haz un merge de tu rama de desarrollo con la rama principal
8. 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_PendingQC*
2. Utilización de interfícies en un Enum

    1. Identifica los 3 sitios en el desarrollo dónde se utiliza el Enum "Clip Quality Control Result"

        - En el control de recepción: codeunit 50102 "Clip Purchase Post", procedimiento OnBeforePostPurchaseDoc
        - En la recepción: codeunit 50102 "Clip Purchase Post", procedimiento OnAfterPostItemJnlLine
        - En la creación de la factura: codeunit 50103 "Clip Purchase Invoice Creation", procedimiento OnBeforeInsertInvLineFromRcptLine

        Cuando creemos un nuevo valor para el Enum, deberemos decidir e implementar:

        - Qué pasa con este nuevo valor en el control de recepción
        - Qué pasa con este nuevo valor en la recepción
        - Qué pasa con este nuevo valor en la creación de la factura

    2. Haz un refactoring del código existente de modo que la implementación de lo que tiene que suceder con cada uno de los valores del Enum esté en distintas codeunits

        1. Crea las siguientes tres codeunits:

```
codeunit 50104 "Clip Blank QC Handler"
{
    procedure PurchaseReceptionControl()
    begin
    end;

    procedure PurchaseReception()
    begin
    end;

    procedure PurchaseInvoiceLineCreationFromRcptLine()
    begin
    end;
}

codeunit 50105 "Clip Satisfactory QC Handler"
```

```
{
    procedure PurchaseReceptionControl()
    begin
    end;

    procedure PurchaseReception()
    begin
    end;

    procedure PurchaseInvoiceLineCreationFromRcptLine()
    begin
    end;
}

codeunit 50106 "Clip NonSatisfactory QCHandler"
{
    procedure PurchaseReceptionControl()
    begin
    end;

    procedure PurchaseReception()
    begin
    end;

    procedure PurchaseInvoiceLineCreationFromRcptLine()
    begin
    end;
}
```

2. Cambia el código de la codeunit 50102 "Clip Purchase Post", procedimiento OnBeforePostPurchaseDoc

```
[EventSubscriber(ObjectType::Codeunit, Codeunit::"Purch.-Post",
'OnBeforePostPurchaseDoc', '', false, false)]
local procedure OnBeforePostPurchaseDoc(var Sender: Codeunit
"Purch.-Post"; var PurchaseHeader: Record "Purchase Header";
PreviewMode: Boolean; CommitIsSupressed: Boolean; var
HideProgressWindow: Boolean);
var
    PurchaseLine: Record "Purchase Line";
    BlankQCHandler: Codeunit "Clip Blank QC Handler";
    SatisfactoryQCHandler: Codeunit "Clip Satisfactory QC
Handler";
    NonSatisfactoryQCHandler: Codeunit "Clip NonSatisfactory
QCHandler";
begin
    if PurchaseHeader."Document Type" <> PurchaseHeader."Document
Type"::Order then
        exit;

    if not PurchaseHeader.Receive then
```

```
        exit;

    PurchaseLine.SetRange("Document Type",
PurchaseHeader."Document Type");
    PurchaseLine.SetRange("Document No.", PurchaseHeader."No.");
    PurchaseLine.SetRange(Type, PurchaseLine.Type::Item);
    PurchaseLine.SetFilter("Qty. to Receive", '<>0');
    if PurchaseLine.FindSet() then
        repeat
            if ItemRequieresQualityControl(PurchaseLine."No.")
then
                case PurchaseLine."Clip Quality Control Result" of
                    PurchaseLine."Clip Quality Control Result"::"
":

BlankQCHandler.PurchaseReceptionControl(PurchaseLine);
                    PurchaseLine."Clip Quality Control
Result"::Satisfactory:

SatisfactoryQCHandler.PurchaseReceptionControl(PurchaseLine);
                    PurchaseLine."Clip Quality Control
Result"::"Non-Satisfactory":

NonSatisfactoryQCHandler.PurchaseReceptionControl(PurchaseLine);
                end;
        until PurchaseLine.Next() = 0;
end;
```

3. Traspasa el código que había en la codeunit 50102 "Clip Purchase Post", procedimiento OnBeforePostPurchaseDoc, a las funciones PurchaseReceptionControl de las 3 codeunits creadas en pasos anteriores
   - codeunit 50104 "Clip Blank QC Handler"

     ```
     procedure PurchaseReceptionControl(PurchaseLine: Record
     "Purchase Line")
     var
         QualityControlIsNotFullyFilledInErr: Label 'All the
     information related to Quality Control must be filled in in
     line %1', Comment = 'ESP="Debe rellenar toda la información
     del control de calidad en la línea %1"';
     begin
         Error(QualityControlIsNotFullyFilledInErr,
     PurchaseLine."Line No.");
     end;
     ```

   - codeunit 50105 "Clip Satisfactory QC Handler"

     ```
     procedure PurchaseReceptionControl(PurchaseLine: Record
     "Purchase Line")
     ```

```
var
    PurchaseQualityMeasures: Record "Clip Purchase Quality
Measures";
    QualityControlIsNotFullyFilledInErr: Label 'All the
information related to Quality Control must be filled in in
line %1', Comment = 'ESP="Debe rellenar toda la información
del control de calidad en la línea %1"';
begin
    PurchaseQualityMeasures.SetRange("Document Type",
PurchaseLine."Document Type");
    PurchaseQualityMeasures.SetRange("Document No.",
PurchaseLine."Document No.");
    PurchaseQualityMeasures.SetRange("Line No.",
PurchaseLine."Line No.");
    PurchaseQualityMeasures.SetFilter(Value, '%1', '');
    if not PurchaseQualityMeasures.IsEmpty() then
        Error(QualityControlIsNotFullyFilledInErr,
PurchaseLine."Line No.");
end;
```

- codeunit 50106 "Clip NonSatisfactory QCHandler"

```
procedure PurchaseReceptionControl(PurchaseLine: Record
"Purchase Line")
var
    PurchaseQualityMeasures: Record "Clip Purchase Quality
Measures";
    QualityControlIsNotFullyFilledInErr: Label 'All the
information related to Quality Control must be filled in in
line %1', Comment = 'ESP="Debe rellenar toda la información
del control de calidad en la línea %1"';
begin
    PurchaseQualityMeasures.SetRange("Document Type",
PurchaseLine."Document Type");
    PurchaseQualityMeasures.SetRange("Document No.",
PurchaseLine."Document No.");
    PurchaseQualityMeasures.SetRange("Line No.",
PurchaseLine."Line No.");
    PurchaseQualityMeasures.SetFilter(Value, '%1', '');
    if not PurchaseQualityMeasures.IsEmpty() then
        Error(QualityControlIsNotFullyFilledInErr,
PurchaseLine."Line No.");
end;
```

4. Publica las modificaciones y ejecuta los tests para comprobar que el desarrollo sigue cumpliendo con los requisitos
5. Haz un commit con las modificaciones realizadas hasta el momento
6. Cambia el código de la codeunit 50102 "Clip Purchase Post", procedimiento OnAfterPostItemJnlLine

```
[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
    BlankQCHandler: Codeunit "Clip Blank QC Handler";
    SatisfactoryQCHandler: Codeunit "Clip Satisfactory QC
Handler";
    NonSatisfactoryQCHandler: Codeunit "Clip NonSatisfactory
QCHandler";
begin
    case PurchaseLine."Clip Quality Control Result" of
        PurchaseLine."Clip Quality Control Result"::" ":
            BlankQCHandler.PurchaseReception(ItemJournalLine,
ItemJnlPostLine);
        PurchaseLine."Clip Quality Control Result"::Satisfactory:

SatisfactoryQCHandler.PurchaseReception(ItemJournalLine,
ItemJnlPostLine);
        PurchaseLine."Clip Quality Control Result"::"Non-
Satisfactory":

NonSatisfactoryQCHandler.PurchaseReception(ItemJournalLine,
ItemJnlPostLine);
    end;
end;
```

7. Traspasa el código que había en la codeunit 50102 "Clip Purchase Post", procedimiento OnAfterPostItemJnlLine, a las funciones PurchaseReception de las 3 codeunits creadas en pasos anteriores
    - codeunit 50104 "Clip Blank QC Handler"

      ```
      procedure PurchaseReception(var ItemJournalLine: Record "Item
      Journal Line"; var ItemJnlPostLine: Codeunit "Item Jnl.-Post
      Line")
      begin
      end;
      ```

    - codeunit 50105 "Clip Satisfactory QC Handler"

      ```
      procedure PurchaseReception(var ItemJournalLine: Record "Item
      Journal Line"; var ItemJnlPostLine: Codeunit "Item Jnl.-Post
      Line")
      begin
      end;
      ```

- codeunit 50106 "Clip NonSatisfactory QCHandler"

```
procedure PurchaseReception(var ItemJournalLine: Record "Item
Journal Line"; var ItemJnlPostLine: Codeunit "Item Jnl.-Post
Line")
var
    ItemJnlLine: Record "Item Journal Line";
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;
```

8. Publica las modificaciones y ejecuta los tests para comprobar que el desarrollo sigue cumpliendo con los requisitos
9. Haz un commit con las modificaciones realizadas hasta el momento
10. Cambia el código de la codeunit 50103 "Clip Purchase Invoice Creation", procedimiento OnBeforeInsertInvLineFromRcptLine

```
[EventSubscriber(ObjectType::Table, Database::"Purch. Rcpt. Line",
'OnBeforeInsertInvLineFromRcptLine', '', false, false)]
local procedure OnBeforeInsertInvLineFromRcptLine(var
PurchRcptLine: Record "Purch. Rcpt. Line"; var PurchLine: Record
"Purchase Line"; PurchOrderLine: Record "Purchase Line");
var
    BlankQCHandler: Codeunit "Clip Blank QC Handler";
    SatisfactoryQCHandler: Codeunit "Clip Satisfactory QC
Handler";
    NonSatisfactoryQCHandler: Codeunit "Clip NonSatisfactory
QCHandler";
begin
    case PurchOrderLine."Clip Quality Control Result" of
        PurchOrderLine."Clip Quality Control Result"::" ":

BlankQCHandler.PurchaseInvoiceLineCreationFromRcptLine(PurchLine);
        PurchOrderLine."Clip Quality Control
Result"::Satisfactory:

SatisfactoryQCHandler.PurchaseInvoiceLineCreationFromRcptLine(Purc
hLine);
        PurchOrderLine."Clip Quality Control Result"::"Non-
Satisfactory":

NonSatisfactoryQCHandler.PurchaseInvoiceLineCreationFromRcptLine(P
urchLine);
```

```
    end;
end;
```

11. Traspasa el código que había en la codeunit 50103 "Clip Purchase Invoice Creation", procedimiento OnBeforeInsertInvLineFromRcptLine, a las funciones PurchaseReception de las 3 codeunits creadas en pasos anteriores
    - codeunit 50104 "Clip Blank QC Handler"

      ```
      procedure PurchaseInvoiceLineCreationFromRcptLine(var
      PurchLine: Record "Purchase Line")
      begin

      end;
      ```

    - codeunit 50105 "Clip Satisfactory QC Handler"

      ```
      procedure PurchaseInvoiceLineCreationFromRcptLine(var
      PurchLine: Record "Purchase Line")
      begin

      end;
      ```

    - codeunit 50106 "Clip NonSatisfactory QCHandler"

      ```
      procedure PurchaseInvoiceLineCreationFromRcptLine(var
      PurchLine: Record "Purchase Line")
      begin
          PurchLine.Validate("Line Discount %", 100);
      end;
      ```

12. Publica las modificaciones y ejecuta los tests para comprobar que el desarrollo sigue cumpliendo con los requisitos
13. Haz un commit con las modificaciones realizadas hasta el momento

3. Crea un nuevo objeto de tipo interface que especifique los 3 métodos que hemos identificado que todo valor del Enum debe implementar. Para ello, utiliza el snippet *tinterface*

```
interface "Clip Quality Control Result Interface"
{
    procedure PurchaseReceptionControl(PurchaseLine: Record "Purchase
Line")
    procedure PurchaseReception(var ItemJournalLine: Record "Item
Journal Line"; var ItemJnlPostLine: Codeunit "Item Jnl.-Post Line")
    procedure PurchaseInvoiceLineCreationFromRcptLine(var PurchLine:
```

```
Record "Purchase Line")
}
```

4. Modifica la definición de las 3 codeunits creadas en pasos anteriores, así como del Enum, para indicar que implementan la interfície creada en el paso anterior

```
codeunit 50104 "Clip Blank QC Handler" implements "Clip Quality Control
Result Interface"
codeunit 50105 "Clip Satisfactory QC Handler" implements "Clip Quality
Control Result Interface"
codeunit 50106 "Clip NonSatisfactory QCHandler" implements "Clip
Quality Control Result Interface"
enum 50100 "Clip Quality Control Result" implements "Clip Quality
Control Result Interface"
```

5. Modifica el código que hace las llamadas a las funciones concretas de cada una de las codeunits para que haga llamadas a las funciones de la interfície

    - codeunit 50102 "Clip Purchase Post", procedimiento OnBeforePostPurchaseDoc

```
[EventSubscriber(ObjectType::Codeunit, Codeunit::"Purch.-Post",
'OnBeforePostPurchaseDoc', '', false, false)]
local procedure OnBeforePostPurchaseDoc(var Sender: Codeunit
"Purch.-Post"; var PurchaseHeader: Record "Purchase Header";
PreviewMode: Boolean; CommitIsSupressed: Boolean; var
HideProgressWindow: Boolean);
var
    PurchaseLine: Record "Purchase Line";
    QCHandler: Interface "Clip Quality Control Result Interface";
begin
    if PurchaseHeader."Document Type" <> PurchaseHeader."Document
Type"::Order then
        exit;

    if not PurchaseHeader.Receive then
        exit;

    PurchaseLine.SetRange("Document Type",
PurchaseHeader."Document Type");
    PurchaseLine.SetRange("Document No.", PurchaseHeader."No.");
    PurchaseLine.SetRange(Type, PurchaseLine.Type::Item);
    PurchaseLine.SetFilter("Qty. to Receive", '<>0');
    if PurchaseLine.FindSet() then
        repeat
            if ItemRequieresQualityControl(PurchaseLine."No.")
then begin
                QCHandler := PurchaseLine."Clip Quality Control
Result";
                QCHandler.PurchaseReceptionControl(PurchaseLine);
```

```
            end;
        until PurchaseLine.Next() = 0;
 end;
```

- codeunit 50102 "Clip Purchase Post", procedimiento OnBeforePostPurchaseDoc

```
[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
    QCHandler: Interface "Clip Quality Control Result Interface";
begin
    QCHandler := PurchaseLine."Clip Quality Control Result";
    QCHandler.PurchaseReception(ItemJournalLine, ItemJnlPostLine);
end;
```

- codeunit 50103 "Clip Purchase Invoice Creation"

```
[EventSubscriber(ObjectType::Table, Database::"Purch. Rcpt. Line",
'OnBeforeInsertInvLineFromRcptLine', '', false, false)]
local procedure OnBeforeInsertInvLineFromRcptLine(var
PurchRcptLine: Record "Purch. Rcpt. Line"; var PurchLine: Record
"Purchase Line"; PurchOrderLine: Record "Purchase Line");
var
    QCHandler: Interface "Clip Quality Control Result Interface";
begin
    QCHandler := PurchOrderLine."Clip Quality Control Result";
    QCHandler.PurchaseInvoiceLineCreationFromRcptLine(PurchLine);
end;
```

6. Especifica en cada uno de los valores del Enum, qué codeunit es la que realiza su implementación

```
enum 50100 "Clip Quality Control Result" implements "Clip Quality
Control Result Interface"
{
    Extensible = true;

    value(0; " ")
    {
        Caption = ' ', comment = 'ESP=" "';
        Implementation = "Clip Quality Control Result Interface" =
"Clip Blank QC Handler";
    }
```

```
    value(1; Satisfactory)
    {
        Caption = 'Satisfactory', comment = 'ESP="Satisfactorio"';
        Implementation = "Clip Quality Control Result Interface" =
"Clip Satisfactory QC Handler";
    }
    value(2; "Non-Satisfactory")
    {
        Caption = 'Non-Satisfactory', comment = 'ESP="No
Satisfactorio"';
        Implementation = "Clip Quality Control Result Interface" =
"Clip NonSatisfactory QCHandler";
    }
}
```

7. Publica las modificaciones y ejecuta los tests para comprobar que el desarrollo sigue cumpliendo con los requisitos

8. Haz un commit con las modificaciones realizadas hasta el momento

3. Creación de un nuevo valor en un Enum
    1. Crea un nuevo valor para el Enum

```
value(3; Pending)
{
    Caption = 'Pending', comment = 'ESP="Pendiente"';
    Implementation = "Clip Quality Control Result Interface" = "Clip
Pending QC Handler";
}
```

    2. Crea una nueva codeunit para desarrollar la implementación del nuevo valor del Enum

```
codeunit 50107 "Clip Pending QC Handler" implements "Clip Quality
Control Result Interface"
{
    procedure PurchaseReceptionControl(PurchaseLine: Record "Purchase
Line")
    begin
    end;

    procedure PurchaseReception(var ItemJournalLine: Record "Item
Journal Line"; var ItemJnlPostLine: Codeunit "Item Jnl.-Post Line")
    begin
    end;

    procedure PurchaseInvoiceLineCreationFromRcptLine(var PurchLine:
Record "Purchase Line")
    begin
```

```
    end;
}
```

4. Desarrollo de tests
    1. Crea el siguiente procedimiento en la codeunit de test para comprobar que se puede registrar la recepción de un pedido de compra cuando el resultado es Pendiente y no hay resultados específicos del control de calidad

```
[Test]
procedure PurchaseOrderPostingP009();
var
    PurchaseHeader: Record "Purchase Header";
    PurchaseLine: Record "Purchase Line";
    Item: Record Item;
    PurchRcptHeader: Record "Purch. Rcpt. Header";
    QualityControlLibrary: Codeunit "Clip Quality Control - Library";
    PurchaseLibrary: Codeunit "Library - Purchase";
    PurchaseDocumentType: Enum "Purchase Document Type";
    PostedDocumentNo: Code[20];
begin
    // [Scenario] The system will allow to post the reception of a
purchase order when the quality control information is set to Pending

    // [Given] Setup: An Item that requires Quality Control
    //                A Purchase Order that uses the item
    //                Quality Control set to Pending and no information
for the specific values
    QualityControlLibrary.CreateItemWithQualityMeasures(Item);

    PurchaseLibrary.CreatePurchHeader(PurchaseHeader,
PurchaseDocumentType::Order.AsInteger(), '');
    PurchaseLibrary.CreatePurchaseLine(PurchaseLine, PurchaseHeader,
PurchaseLine.Type::Item.AsInteger(), Item."No.", 1);

    PurchaseLine."Clip Quality Control Result" := PurchaseLine."Clip
Quality Control Result"::Pending;
    PurchaseLine.Modify();

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

    // [Then] Verify: The order has been posted
    AssertThat.AreNotEqual('', PostedDocumentNo, 'C01.P009.A001
PostedDocumentNo');
    AssertThat.AreEqual(true, PurchRcptHeader.Get(PostedDocumentNo),
'C01.P009.A002 PurchRcptHeader.Get');
end;
```

2. Crea el siguiente procedimiento de test para comprobar que al registrar una línea de pedido de compra con resultado Pendiente, la cantidad es traspasada al almacén de CALIDAD

```
[Test]
procedure PurchaseOrderPostingP010();
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 Pending, the system will post the
reception and a transfer for the same quantity to a QUALITY location

    // [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
    //                      - Pending 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"::Pending);

    // [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.P010.A001
```

```
First item ItemLedgerEntry.Count()');

    ItemLedgerEntry.SetRange("Document No.", PostedDocumentNo);
    ItemLedgerEntry.SetRange("Item No.", PurchaseLine2."No.");
    AssertThat.AreEqual(3, ItemLedgerEntry.Count(), 'C01.P010.A011
Second item ItemLedgerEntry.Count()');

    ItemLedgerEntry.FindFirst();
    AssertThat.AreEqual(ItemLedgerEntry."Entry Type"::Purchase,
ItemLedgerEntry."Entry Type", 'C01.P010.A012 Second item
ItemLedgerEntry."Entry Type"');

    ReceiptILEEntryNo := ItemLedgerEntry."Entry No.";
    ItemLedgerEntry.Next();
    AssertThat.AreEqual(ItemLedgerEntry."Entry Type"::Transfer,
ItemLedgerEntry."Entry Type", 'C01.P010.A021 Second item
ItemLedgerEntry."Entry Type"');
    AssertThat.AreEqual(-PurchaseLine2.Quantity,
ItemLedgerEntry.Quantity, 'C01.P010.A022 Second item
ItemLedgerEntry.Quantity');
    AssertThat.AreEqual(ReceiptILEEntryNo, ItemLedgerEntry."Applies-to
Entry", 'C01.P010.A023 Second item ItemLedgerEntry."Applies-to
Entry"');

    ItemLedgerEntry.Next();
    AssertThat.AreEqual(ItemLedgerEntry."Entry Type"::Transfer,
ItemLedgerEntry."Entry Type", 'C01.P010.A031 Third item
ItemLedgerEntry."Entry Type"');
    AssertThat.AreEqual(PurchaseLine2.Quantity,
ItemLedgerEntry.Quantity, 'C01.P010.A032 Third item
ItemLedgerEntry.Quantity');
    AssertThat.AreEqual('ROJO', ItemLedgerEntry."Location Code",
'C01.P010.A033 Third item ItemLedgerEntry."Location Code"');
end;
```

3. Crea el siguiente procedimiento de test para comprobar que una línea registrada con Control de Calidad Pendiente no puede ser incluida en una factura

```
[Test]
[HandlerFunctions('GetReceiptLinesPage')]
procedure InvoiceCreationP011();
var
    PurchaseOrderHeader: Record "Purchase Header";
    PurchaseInvoiceHeader: Record "Purchase Header";
    PurchaseLine: Record "Purchase Line";
    InvoiceLine: Record "Purchase Line";
    Item: Record Item;
    QualityControlLibrary: Codeunit "Clip Quality Control - Library";
    PurchaseLibrary: Codeunit "Library - Purchase";
    PurchaseDocumentType: Enum "Purchase Document Type";
    PostedDocumentNo: Code[20];
```

```
begin
    // [Scenario] When an invoice is created with the "Get Receptions"
functionality, items received as Pending cannot be included in the
invoice

    // [Given] Setup: An item that requires Quality Control
    //                A Purchase Order that uses the items
    //                Pending Quality Control information set on the
order
    //                A Posted Receipt for the order
    //                A Purchase Invoice header
    QualityControlLibrary.CreateItemWithQualityMeasures(Item);

    PurchaseLibrary.CreatePurchHeader(PurchaseOrderHeader,
PurchaseDocumentType::Order.AsInteger(), '');
    PurchaseLibrary.CreatePurchaseLine(PurchaseLine,
PurchaseOrderHeader, PurchaseLine.Type::Item.AsInteger(), Item."No.",
10);


QualityControlLibrary.SetQualityControlInformationOnPurchaseLine(Purcha
seLine, PurchaseLine."Clip Quality Control Result"::Pending);

    PostedDocumentNo :=
PurchaseLibrary.PostPurchaseDocument(PurchaseOrderHeader, true, false);

    PurchaseLibrary.CreatePurchHeader(PurchaseInvoiceHeader,
PurchaseDocumentType::Invoice.AsInteger(), PurchaseOrderHeader."Buy-
from Vendor No.");

    // [When] Exercise: Use the "Get Receptions" functionality to
create the Invoice lines
    InvoiceLine."Document Type" := PurchaseInvoiceHeader."Document
Type";
    InvoiceLine."Document No." := PurchaseInvoiceHeader."No.";
    asserterror PurchaseLibrary.GetPurchaseReceiptLine(InvoiceLine);

    // [Then] Verify: The Pending item line has thrown an error and has
not been created in the invoice
    AssertThat.AreEqual('The item has been received as Pending Quality
Control, it cannot be included in the Invoice', GetLastErrorText(),
'C01.P011.A002 InvoiceLine.Get');
    AssertThat.AreEqual(false,
InvoiceLine.Get(PurchaseInvoiceHeader."Document Type",
PurchaseInvoiceHeader."No.", 20000), 'C01.P011.A002 InvoiceLine.Get');
end;
```

5. Implementación de la interfície para el nuevo valor
    1. Escribe el siguiente código en la codeunit que implementa el valor Pendiente

```
codeunit 50107 "Clip Pending QC Handler" implements "Clip Quality
Control Result Interface"
{
    procedure PurchaseReceptionControl(PurchaseLine: Record "Purchase
Line")
    begin
    end;

    procedure PurchaseReception(var ItemJournalLine: Record "Item
Journal Line"; var ItemJnlPostLine: Codeunit "Item Jnl.-Post Line")
    var
        ItemJnlLine: Record "Item Journal Line";
    begin
        ItemJnlLine := ItemJournalLine;
        ItemJnlLine."Entry Type" := ItemJnlLine."Entry Type"::Transfer;
        ItemJnlLine."New Location Code" := 'ROJO';
        ItemJnlLine."Invoiced Quantity" := ItemJnlLine.Quantity;
        ItemJnlLine."Invoiced Qty. (Base)" := ItemJnlLine."Quantity
(Base)";
        ItemJnlLine."Item Shpt. Entry No." := 0;
        ItemJnlLine."Applies-to Entry" := ItemJournalLine."Item Shpt.
Entry No.";
        ItemJnlPostLine.RunWithCheck(ItemJnlLine);
    end;

    procedure PurchaseInvoiceLineCreationFromRcptLine(var PurchLine:
Record "Purchase Line")
    var
        CannotIncludePendingLinesOnInvoiceErr: Label 'The item has been
received as Pending Quality Control, it cannot be included in the
Invoice', Comment = 'ESP="El producto ha sido recibido con Control de
Calidad Pendiente, no puede ser incluido en la factura"';
    begin
        Error(CannotIncludePendingLinesOnInvoiceErr);
    end;
}
```

6. Haz un merge de tu rama de desarrollo con la rama principal
7. Elimina la rama de desarrollo