Taller: Desarrollo ABAP-FI con GitHub Copilot

BLOQUE 4: Lógica de Negocio y Conversión

Objetivos de Aprendizaje

- Implementar conversión de moneda usando BAPIs estándar
- Aplicar cálculos de antigüedad de saldos
- Usar sintaxis ABAP 7.4+ avanzada (REDUCE, COND, SWITCH)
- Aplicar principios SOLID en la lógica de negocio

EJERCICIO 4 - Conversión de Moneda

Instrucciones:

PASO 1: Crear proceso de conversion conversión

1. **En Copilot Chat, ejecutar:**

Genera una clase ABAP llamada <'lcl_xxxx'> para convertir

importes entre monedas en el módulo FI.

REQUISITOS:

CLASE: lcl_currency_converter

- Aplicar principio Single Responsibility

- Solo responsable de conversión de moneda

- Sin acceso directo a base de datos

MÉTODO PÚBLICO: convert_amount

- Descripción: Convierte un importe de una moneda a otra

- Parámetros de importación:

* is_params: Estructura <'indicamos la estructura'> con:

- amount_from: Importe a convertir

- currency_from: Moneda origen

- currency_to: Moneda destino

- conv_date: Fecha para obtener tipo de cambio

- exch_type: Tipo de cambio (default 'M')

- company_code: Sociedad

- Retorno: Estructura ty_conversion_result

- Lógica:

* Validar que monedas no estén vacías

* Si moneda origen = moneda destino, retornar mismo importe sin convertir

* Usar función para conversión

* Capturar excepciones y retornar mensaje de error

* Status: S=Success, E=Error, W=Warning

IMPORTANTE:

- Incluir ABAPDoc completo

- Usar sintaxis ABAP 7.4+

- TRY-CATCH para excepciones

Genera la clase completa con DEFINITION e IMPLEMENTATION.

2. **Copiar el código** en tu programa (después de lcl_data_access)
3. **Activar y verificar** que no hay errores
4. **Documentar**

PASO 2: Probar conversión

4. **Agregar código de prueba temporal al final del programa:**

```
START-OF-SELECTION.
  " Prueba rápida de conversión
  DATA(lo_converter) = NEW lcl_currency_converter( ).

  DATA(ls_params) = VALUE ty_conversion_params(
    amount_from   = '1000.00'
    currency_from = 'EUR'
    currency_to   = 'USD'
```

```
  conv_date    = sy-datum
  exch_type    = 'M'
  company_code  = '1000'
).

DATA(ls_result) = lo_converter->convert_amount( ls_params ).

WRITE: / 'Importe convertido:', ls_result-amount_converted.
WRITE: / 'Tipo de cambio:', ls_result-exchange_rate.
WRITE: / 'Status:', ls_result-status.
```

5. **Ejecutar el programa** (F8) y verificar que funciona
6. **Comentar o borrar** el código de prueba

EJERCICIO 5 - Clase de Lógica de Negocio

Instrucciones:

PASO 1: Crear clase principal de negocio e implementación

1. **En Copilot:**

Pide que genere la lógica de métodos necesaria para completar la implementación de la lógica de negocio, puedes generar una clase adicional para implementar los métodos o agregar los métodos de lógica de negocio a tu clase y en tu programa implementarlos.

2. **Copiar el código** en tu programa
3. **Activar y corregir** errores si los hay
4. **Documentar**

BLOQUE 6: Debug y Unit Testing

FASE 5: El Rol del Revisor

EJERCICIO 6 - Revisión de Código con Copilot

Instrucciones para el Estudiante:

1. **Selecciona la clase de tu implementación**
2. **Copilot Chat, ejecutar:**

Revisa el siguiente código ABAP y proporciona:

1. ANÁLISIS DE CALIDAD:

- ¿Sigue principios SOLID?
- ¿Hay código duplicado?
- ¿La nomenclatura es clara?

2. MEJORAS DE PERFORMANCE:

- ¿Se pueden optimizar los loops?
- ¿Hay cálculos repetidos?

4. **Leer las sugerencias** y discutir con compañero
5. **Aplicar al menos 2 mejoras sugeridas**
6. **Documentación**

EJERCICIO 8 - Unit Testing con Copilot

Instrucciones:

PASO 1: Crear test para uno de tus metodos

1. **En Copilot Chat:**

Genera una clase de Unit Test ABAP para lcl_business_processor.

REQUISITOS:

CLASE DE TEST: ltc_business_processor

- FOR TESTING

- RISK LEVEL HARMLESS

- DURATION SHORT

TESTS A CREAR: para el método <nombre de metodo>

IMPORTANTE:

- Usar cl_abap_unit_assert para assertions

- Usar sintaxis ABAP 7.4+ en los tests

Genera: clase de test completa con mocks y todos los métodos.

2. **Copiar el código** al final de tu programa
3. **Activar el programa**
4. **Ejecutar los tests**
5. **Verificar que todos pasen**
6. **Documentar**
