Taller: Desarrollo ABAP-FI con GitHub Copilot

BLOQUE 1: Introducción y Diseño

Objetivos de Aprendizaje

- Entender cómo transformar requerimientos funcionales en diseño técnico
- Dominar la técnica correcta para aplicar un prompting efectivo
- Aplicar contexto completo en las consultas a Copilot

EJERCICIO 1 - Prompt de Contexto Completo

Instrucciones para el Estudiante:

1. **Abrir GitHub Copilot Chat** en tu IDE (Eclipse ADT)
2. **Analiza el requerimiento para diseñar la arquitectura de tu solución.**
3. **Crea un promt y ejecutar en COPILOT:**

   Sugerencia:

   Asigna un rol para dar un contexto de como debe comportarse COPILOT.

   Proporcionar en detalle el **C**ontexto, **O**bjetivo, **S**intaxis, **T**ipos de datos, **A**rquitectura, **R**estricciones.

   Detallar las actividades según definas la estructura de la solución.

4. **Correo del funcional**:
   CASO DE NEGOCIO - Correo del Funcional

   De: Ana Martínez - Consultor Senior FI
   Para: Equipo Desarrollo ABAP
   Asunto: URGENTE - Reporte Mensual de Antigüedad de Saldos

   Equipo,

   Para el cierre de mes necesitamos un reporte que consolide las partidas abiertas de clientes y proveedores de todas nuestras sociedades.

   REQUERIMIENTOS FUNCIONALES:

   1. DATOS A MOSTRAR:
   - Sociedad
   - Número de cliente/proveedor
   - Nombre del interlocutor comercial
   - Número de documento contable
   - Fecha de documento
   - Fecha de contabilización

- Importe en moneda original
- Importe convertido a USD (moneda de reporte)
- Días transcurridos desde la contabilización
- Clasificación de antigüedad (0-30, 31-60, 61-90, >90 días)
- Indicador visual de estado

2. FILTROS REQUERIDOS:
- Sociedad (obligatorio)
- Rango de clientes
- Rango de proveedores
- Rango de fechas de contabilización
- Moneda de reporte (default USD)

3. REGLAS DE NEGOCIO:
- Solo partidas abiertas (no compensadas)
- Conversión de moneda al tipo de cambio M (medio) de la fecha de contabilización
- Si la moneda original es igual a la de reporte, no convertir
- Clasificación por días:
  * 0-30 días: Verde (bajo riesgo)
  * 31-60 días: Amarillo (riesgo medio)
  * 61-90 días: Naranja (riesgo alto)
  * >90 días: Rojo (riesgo crítico)

4. SALIDA:
- ALV con posibilidad de exportar a Excel
- Totales por moneda al final del reporte
- Agrupamiento por sociedad

El equipo de finanzas usará este reporte semanalmente para gestión de cobranza y pagos. Es crítico que sea confiable y rápido.

Saludos,
Ana Martínez

5. **Analizar la respuesta de Copilot:**
    1. ¿Identificó todas las tablas necesarias?
    2. ¿La arquitectura de clases sigue SOLID?
    3. ¿El diseño es comprensible?
6. **Documentar los resultados**

BLOQUE 2: Análisis y Estructura de Datos

EJERCICIO 2 - Definición de Tipos de Datos

Instrucciones:

PASO 1: Generar tipos básicos

1. **Crear un nuevo programa ABAP**: ZFI _REPORT_XX (XX = tus iniciales)
2. **Crear un prompt para que COPILOT defina los tipos de datos.**
3. **Puntos para considerar en tu instrucción:**

- Define los requisitos de cómo debe estar formadas tus estructuras (nombre, campos...)
- Define los límites de cómo debe trabajar COPILOT, ejemplo:
  Usar nomenclatura clara y estándar
  Incluir comentarios ABAPDoc en cada campo
  Genera el código completo de TYPES con sintaxis ABAP 7.4+

3. **Copiar el código generado** en tu programa
4. **Activar el programa** (Ctrl+F3) y verificar que no hay errores

---

PASO 2: Refinar y mejorar

Si hay campos faltantes o quieres mejoras, crea un prompt indicando que puntos crees que debería mejorar para que responda con resultados que puedas utilizar en tu código:

Ejem: Mejora las estructuras TYPES anteriores añadiendo validaciones internas....

Actualiza el código manteniendo lo anterior.

6. **Reemplazar el código anterior** con la versión mejorada
7. **Activar y verificar**
8. **Documentar los resultados**

BLOQUE 3: Capa de Datos

EJERCICIO 3 - Clase de Acceso a Datos

Instrucciones:

PASO 1: Crear clases con COPILOT

1. **Crear un prompt, dónde definas la creación de la clase, métodos aplicando los principios SOLID.** En esta instrucción debes detallar como será el diseño de tu solución, que campo utilizará, los parámetros de entrada y salida de cada uno de los métodos creados, y todos los requisitos que consideres deben de cumplir (sintaxis, joins, campos de tablas, que lógica aplicar).

2. **Copiar el código generado** en tu programa
3. **Activar** (Ctrl+F3) y corregir errores si los hay

PASO 2: Validar y refinar

4. **Revisar el código generado:**
    - ¿Los JOINs son correctos?
    - ¿Usa sintaxis 7.4+?
    - ¿Tiene ABAPDoc?
5. **Si necesitas mejoras, crea un prompt para que COPILOT refine el código generado:**

Ejemplo:

Mejora la clase lcl_data_access con lo siguiente:

- Añadir TRY-CATCH para capturar cx_sy_sql_error

- Si hay error SQL, retornar tabla vacía y hacer log del error

- Usar CORRESPONDING para mapear campos automáticamente

Actualiza solo la IMPLEMENTATION, mantén la DEFINITION igual.

6. **Actualizar el código** y activar
7. **Documentar los resultados**