

## HOJA DE TRABAJO PROYECTO FINAL

### INFORMACIÓN DEL PROYECTO

---

#### REQUERIMIENTO FUNCIONAL

**Título:** Sistema de Auditoría y Corrección de Business Partners

**Descripción:**

Desarrollar un programa que identifique Business Partners con datos inconsistentes o incompletos, valide la información según normativas y permita la corrección masiva de registros mediante APIs estándar de SAP.

**Validaciones requeridas:**

- **NIT Empresarial:** Formato XXXXXXXX-X (8-9 dígitos + verificador)
- **DPI Personal:** 13 dígitos consecutivos
- **Email:** Formato dirección RFC 5322, formato estándar con @ y dominio
- **Teléfono:** +502 seguido de 8 dígitos

**Funcionalidades:**

- Lectura selectiva de BPs desde BUT000
  - Validación automática según tipo (Persona/Organización)
  - Reporte visual con semáforos (Correcto/Advertencia/Error)
  - Actualización controlada vía BAPI
  - Log de auditoría de cambios realizados
-

## **BLOQUE 1: Diseño y Arquitectura**

### **Objetivos**

Diseñar la arquitectura aplicando principios SOLID y orientación a objetos.

### **Tareas**

#### **1.1 Aplicar Principios SOLID**

- Una clase por responsabilidad
  - Lectura de datos
  - Validación organizaciones
  - Validación personas
  - Validación contactos
  - Actualización
  - Coordinación

#### **1.2 Diseño de Interfaces**

- Interfaz con métodos de validación comunes
- Interfaz para lectura de datos
- Interfaz para actualización de registros

#### **1.3 Arquitectura de Componentes**

- Definir responsabilidades de cada clase
- Establecer flujo de comunicación entre componentes

### **Entregable para la sesión del 06 de enero:**

Creación de documento con los prompts diseñados para cada una de las tareas y el diseño de la arquitectura de la solución creada con apoyo de COPILOT.

Esta información la presentarán en la sesión.

---

## **BLOQUE 2: Desarrollo Core**

### **Objetivos**

Implementar lógica de negocio con sintaxis ABAP 7.4+.

### **Tareas**

#### **2.1 Sintaxis Moderna ABAP**

- Declaraciones inline: DATA(lv\_...), FIELD-SYMBOL(<fs\_...>)
- Operadores de tabla: FILTER, VALUE, CORRESPONDING
- Expresiones lógicas: COND, SWITCH

#### **2.2 Implementar Clases de Validación**

- Clase para validar NIT
- Clase para validar DPI/CUI
- Clase para validar email y teléfono

#### **2.3 Clase Procesadora**

- Coordina lectura, validación y preparación de datos
- Determinar tipo de BP y aplicar validaciones correspondientes
- Generar propuestas de corrección

**Entregable sesión 13 de febrero:** Clases implementadas funcionalmente

---




## **BLOQUE 3: UI y Persistencia**

### **Objetivos**

Crear interfaz de usuario y gestionar actualizaciones transaccionales.

### **Tareas**

#### **3.1 Reporte ALV**

- Usar CL\_SALV\_TABLE
- Columnas: BP | Tipo | Campo | Valor Actual | Valor Corregido | Estado
- Semáforos de estado:
  -  Verde: Dato válido
  -  Amarillo: Dato corregible
  -  Rojo: Error crítico

#### **3.2 Pantalla de Selección**

- Rango de Business Partners
- Tipo de BP (Persona/Organización/Todos)
- Checkbox "Modo simulación"
- Checkbox "Solo con errores"

#### **3.3 Gestión Transaccional**

- Llamar BAPI para actualización de datos
- Ejecutar BAPI para COMMIT, solo si actualización exitosa
- Ejecutar BAPI para ROLLBACK en caso de error
- Capturar y mostrar mensajes de retorno

#### **3.4 Log de Operaciones**

- Registrar BPs procesados
- Registrar cambios realizados
- Registrar errores encontrados

**Entregable sesión del 20 de enero:** Reporte ejecutable con ALV y transacciones funcionales

---

## BLOQUE 4: Calidad y Documentación

### Objetivos

Asegurar calidad del código mediante pruebas y documentación técnica.

### Tareas

#### 4.1 ABAP Unit Tests Crear clases de prueba para:

- Validación NIT (casos válidos, inválidos)
- Validación DPI (válidos, inválidos)
- Validación Email (válidos y con errores de formato)
- Validación Teléfono (con y sin código país correcto)

#### 4.2 ABAPDoc Completo Documentar usando formato estándar:

abap

```
"! <p class="shorttext synchronized">Descripción breve</p>
```

```
"! Descripción detallada explicando funcionalidad
```

```
"! @parameter iv_nit | NIT a validar formato XXXXXXXX-X
```

```
"! @returning rv_valid | Verdadero si cumple formato
```

```
"! @raising zcx_bp_validation | Error de validación
```

Documentar:

- Todas las clases públicas
- Todos los métodos públicos
- Incluir reglas de negocio y normativas aplicables

#### 4.3 Revisión con Copilot

- Verificar parámetros de BAPIs contra documentación oficial
- Validar que no existan campos o parámetros incorrectos
- Revisar manejo completo de excepciones
- Confirmar adherencia a convenciones de nomenclatura SAP

**Entregable sesión 27 de febrero:** Suite de tests ejecutable + documentación ABAPDoc generada

---

## CRITERIOS DE ACEPTACIÓN

- ✓ Arquitectura respeta principios SOLID
- ✓ Sintaxis ABAP 7.4+ aplicada consistentemente
- ✓ Validaciones funcionales para NIT, DPI, email y teléfono
- ✓ ALV presenta información clara con semáforos
- ✓ Actualización transaccional robusta con manejo de errores
- ✓ Cobertura de pruebas unitarias  $\geq 75\%$
- ✓ ABAPDoc completo en todos los componentes públicos
- ✓ Código revisado y validado con Copilot
- ✓ Entrega de informe con estimaciones de tiempo

---

## INSTRUCCIONES GENERALES

1. Completar bloques en orden secuencial
2. Usar GitHub Copilot activamente en cada fase
3. No utilizar datos reales de clientes en pruebas
4. Documentar decisiones técnicas importantes
5. Solicitar code review al finalizar cada bloque