PoC4 - Validación de Historias de Usuario con IA

# 📊 Diseño Funcional

## 🧭 Propósito funcional

Validar, puntuar y enriquecer historias de usuario en Jira a través de un microservicio que actúe como evaluador automático. Este sistema debe integrarse con Jira en tiempo real y devolver la evaluación directamente en el mismo issue.

## 🧩 Componentes funcionales

|  |  |
| --- | --- |
| Componente | Función principal |
| Jira Cloud | Plataforma donde se escriben las HU (historias de usuario) |
| Webhook de Jira | Dispara una notificación cuando se crea o edita un issue |
| Azure Function | API REST que recibe la HU y ejecuta la validación |
| Motor de IA - Copilot Studio | Analiza la estructura del texto con base en comprensión semántica |
| Campo personalizado en Jira | Recibe la calificación o feedback generado automáticamente |

## 📌 Inputs esperados

- Campo `description` del issue en Jira  
- Datos del issue (ID, tipo, proyecto, etc.)

## 📤 Outputs generados

- `score`: Puntaje de cumplimiento de formato `Como, quiero, para`  
- `observaciones`: Texto libre con recomendaciones  
- Actualización en el custom field del mismo issue en Jira

## ⚙️ Acciones funcionales esperadas

|  |  |
| --- | --- |
| Acción del usuario | Comportamiento del sistema |
| Crear un issue en Jira | Se dispara webhook → Azure Function evalúa |
| Editar descripción del issue | Se vuelve a evaluar la historia |
| Historia cumple formato | Se asigna puntaje alto y observación positiva |
| Historia incompleta | Se asigna puntaje bajo y se da retroalimentación |

## ✅ Reglas funcionales clave

1. Solo se validan issues de tipo “Historia de Usuario” (`issuetype.name == Story`)  
2. El análisis se dispara solo si el campo `description` está presente  
3. El puntaje se calcula sobre:  
 - ¿Contiene 'Como'?  
 - ¿Contiene 'quiero'?  
 - ¿Contiene 'para'?  
 - ¿Cumple estructura básica?  
4. La función debe responder en menos de 10 segundos  
5. Debe dejar registro en logs para auditoría