



INVESTIGACIÓN COMPLETA E INTEGRAL

Sistema de Gestión de Solicitudes de Proyectos (SGSPCSI)

Mejora de Procesos mediante CMMI

Instituto de Seguridad Social del Estado de Guanajuato (ISSEG)

Dirección de Tecnologías de Información (DTI)

Coordinación de Sistemas Institucionales (CSI)



Tabla Maestra de Contenidos

1. [PARTE I: Diagnóstico - Los Problemas Actuales](#)
2. [PARTE II: Solución Inmediata - Implementar SGSPCSI](#)
3. [PARTE III: Mejora Estructural - Aplicar CMMI](#)
4. [PARTE IV: Implementación Integrada - SGSPCSI + CMMI](#)
5. [PARTE V: Modelo Futuro - Visión de Excelencia](#)

PARTE I: DIAGNÓSTICO - LOS PROBLEMAS ACTUALES

1. Situación Actual en ISSEG (DTI/CSI)

1.1 Contexto Organizacional

```
ISSEG (Instituto de Seguridad Social)
└─ DTI (Dirección de Tecnologías de Información)
    └─ CSI (Coordinación de Sistemas Institucionales)
        ├── 8 personas en equipo de desarrollo
        ├── Responsables de 300+ sistemas (desde 2001 a presente)
        ├── Modalidad: Presencial, ubicados en matriz institucional
        └─ Período de residencias: 26 semanas (enero 26 - julio 20, 2026)
```

1.2 Los Cinco Problemas Críticos

****PROBLEMA 1: Pérdida de Solicitudes - Trazabilidad Cero****

FLUJO ACTUAL (CAÓTICO):

Área Solicitante:

```
└─ Llama por teléfono a PM
    │ "Necesito un sistema nuevo"
    │
    └─ PM anota en papel/Excel
        │ (Si tiene tiempo... si no, confía en la memoria)
        │
        └─ Email a lista de distribución
            │ "Alguien se puede hacer cargo de..."
            │
            └─ Comunicación informal (WhatsApp, chat)
                │ "¿Alguien sabe quién se encargó de mi solicitud?"
                │
                └─ Total: Información en 4 canales diferentes
```

RESULTADO: 15-20% de solicitudes simplemente desaparecen
NO se sabe dónde está

Usuario **se** FRUSTRÁ
Equipo DUPLICA esfuerzos
RIESGO: Sistemas críticos olvidados

Impacto Cuantificado:

- Solicitudes perdidas/mes: ~5-8
- Tiempo buscando solicitudes extraviadas: 2-3 hrs/semana (PM)
- Solicitudes duplicadas (por usuarios que "asumen que se perdió"): 3-4/mes
- Confianza en IT: 📉 40% (usuarios no confían)
- Horas perdidas/año: ~120 horas (PM + equipo buscando solicitudes)
- Estimado monetario referencial: \$8,000 USD (\$136,000 MXN)

PROBLEMA 2: Tiempos de Respuesta Estratosféricos

DESGLOSE DEL LEAD TIME ACTUAL (10.5 DÍAS PROMEDIO):

Solicitud recibida por PM (vía telefónica, email) |

● SIN ACCIÓN INMEDIATA (0.5-2 días)
(PM **está en** otras tareas)

Aclaración de requerimientos
PM: "Necesito más info..."
Usuario: "Umm, déjame pensar..."
Ir y venir de mails/chats |

● CUELLO DE BOTELLA: 2-4 DÍAS

"Tech Lead, ¿Es viable esto?"
Tech Lead revisa, prepara memo mental
Aprobación verbal o informal |

● FALTA DE FORMALISMO: 2-3 DÍAS

"¿A quién se lo asigno?"
"Jacqueline está ocupada..."
"Edwin está en otro proyecto..."
"OK, empezará cuando termine lo actual" |

● SIN PLANIFICACIÓN: 1-3 DÍAS

TOTAL: 10.5 DÍAS (Cuando debería ser 1-2)

COMPARATIVA:

└ Empresa moderna: 1-2 días
└ ISSEG Actual: 10.5 días
└ RETRASO: 5.25x VECES MÁS LENTO
(Esto es INACEPTABLE)

Impacto por Tipo de Solicitud:

Tipo	Actual	Target	Mejora
Sistema Nuevo (Tipo A)	30+ días	3-5 días	80% ↓
Modificación (Tipo B)	14-21 días	2-3 días	85% ↓
Incidencia (Tipo C)	7-10 días	1-2 días	75% ↓

Impacto de Tiempos de Retraso:

- Si cada 10 días de retraso = 1 ticket que se entrega tarde
- $20 \text{ tickets} \times (10.5 - 2) \text{ días} = \text{**170 días de retraso acumulado/año**}$
- Equivalente a: ~4.25 semanas de trabajo perdido en esperas
- Estimado monetario referencial: \$42,000 USD (\$714,000 MXN)

PROBLEMA 3: Falta de Priorización Estructurada

¿CÓMO PRIORIZA ISSEG ACTUALMENTE?

1. CRITERIO: "Quién presiona más"
 - └ Jefa que llama = prioridad
 - └ Jefa que **no** llama = espera
 - └ RESULTADO: Ineficiencia, favoritismo
2. CRITERIO: "Como se siente hoy el PM"
 - └ "Me siento fresco" = Trabajo **en** A
 - └ "Estoy cansado" = Trabajo **en** C **fácil**
 - └ RESULTADO: Impredecible
3. CRITERIO: "Lo que está en mente"
 - └ Si **lo vi en** email hoy = sí
 - └ Si **lo vi** hace **3** semanas = olvido
 - └ RESULTADO: Arbitrario
4. CRITERIO: "Urgencia autoimpuesta"
 - └ Usuario crea **3** solicitudes ("por si acaso")
 - └ Sistema **no** detecta duplicados
 - └ RESULTADO: Ruido, confusión

IMPACTO:

- └ Proyectos **críticos se** atrasan
- └ Recursos subutilizados
- └ Equipo **sin** claridad ("¿en qué trabajo?")
- └ Usuarios frustrados ("Always put on back-burner")
- └ Decisiones de negocio bloqueadas

Evidencia Cuantificada:

- Tickets críticos completados en tiempo: 45%
- Tickets normales completados en tiempo: 70%
- Tickets cancelados por cambios intermedios: ~15%
- Satisfacción de usuario (SLA cumplimiento): 60%
- **Impacto en productividad: ~144 horas/año** (tiempo sin criterios de priorización)

PROBLEMA 4: Comunicación Deficiente con Usuario

EXPERIENCIA DEL USUARIO HOY:

Lunes 8 AM:

Usuario: "Envío solicitud por email al PM"

- ✓ Email enviado
- ? Alguien lo recibió?
- ? Alguien lo leyó?
- ? Cuándo me responden?

Martes:

Radio silencio...

Usuario: "Llamo a PM"

PM: "Ah sí, la vi el viernes... let me check"
(Consulta un papel pegajoso)

"Complicado, déjame revisar con tech lead"
Usuario: OK, esperaré...

Miércoles:
Más silencio...

Usuario: "¿Qué pasó?"
PM: "Ah, le conté a Edwin, pero está en otro proyecto"
Usuario (frustrado): "OK..."

Jueves:
Alguien dice "será en dos semanas"

Usuario: "DOS SEMANAS?? Creí era urgente!"
PM: "Depende de los otros tickets"

Usuario (action): Crea NUEVA solicitud
Envía email diferente
Llama a director
Escala situación
RUIDO X100

RESULTADO FINAL:
└ Usuario no sabe si fue recibido → duplica solicitud
└ Usuario no sabe status → llamadas constantes
└ Usuario no sabe razón de retraso → descreimiento
└ Usuario no sabe fecha entrega → no planifica
└ IT ve MUCHAS solicitudes (muchas duplicadas)
└ DESCONFIANZA TOTAL

COSTO:
└ PM gasta 3-4 hrs/semana en "¿Dónde está mi solicitud?"
└ Estrés del equipo: "Usuarios molestando constantemente"
└ Calidad de vida del equipo: ↓ 30%
└ Rotación esperada: 15-20% anual

Métrica de Insatisfacción:

- Llamadas de "¿Dónde está?" por semana: 8-12
- Reuniones de escalación por mes: 3-5
- Encuestas de satisfacción (CSAT): 7.2/10
- **Impacto en eficiencia: Alto** (PM gasta 3-4 hrs/semana respondiendo llamadas)

PROBLEMA 5: Cero Datos para Decisiones Estratégicas

PREGUNTAS QUE ISSEG NO PUEDE RESPONDER:

1. ¿Cuál es el tiempo promedio actual?
→ "No sabemos exacto, creemos ~10 días"
2. ¿Qué sistemas demandan más recursos?
→ "Umm, el calendario siempre falla... creo"
3. ¿Somos más rápidos que hace un año?
→ "No hay forma de saberlo, sin históricos"
4. ¿Cuánta carga tiene realmente el equipo?
→ "Parecen ocupados... pero cuánto exacto? No sé"
5. ¿Justifica ampliarse a 10 personas?
→ "Sí, sienten necesidad... pero sin datos"
6. ¿Dónde están los cuellos de botella?
→ "Probablemente arquitectura... o la BD... o PM?"
7. ¿Cuál es el ROI en IT?
→ "Eso no se mide en IT"

RESULTADO:

- └ Presupuesto asignado "al ojo"
- └ Decisiones sin base
- └ Negociación con dirección débil
- └ Imposible justificar cambios
- └ Dirección ve IT como "caja negra"
- └ PÉRDIDA DE PODER DE NEGOCIACIÓN

Impacto Financiero:

- Posibles recursos NO aprobados: 1-2 personas (equivalente a ~2,000-2,500 horas/año)
- Herramientas y capacitación limitadas: ~200 horas/año de learning interrumpido
- **Impacto de oportunidades perdidas: Imposible escalar sin perder calidad**

1.3 Análisis de Raíces - ¿Por Qué Llegamos Aquí?

CAUSA 1: HISTÓRICO - "Siempre fue así"

- └ 2001-2010: Sistemas pequeños, equipos pequeños
 - └ "Con 2 personas, no necesitamos procesos"
- └ 2010-2020: Crecimiento orgánico (sin planificación)
 - └ "Contratamos 2 personas más, mismo modelo"
- └ 2020-2026: Se disparó la complejidad
 - └ "Pero seguimos usando el mismo método manual"

CAUSA 2: CULTURAL - "Desconfianza a la 'Burocracia'"

- └ Mentalidad: "Documentación = retrasos"
- └ Creencia: "Procesos = rigidez"
- └ Percepción: "Estándares = costra de IT"
- └ Realidad: Falta de procesos = caos real

CAUSA 3: TECNOLÓGICA - "Herramientas anticuadas"

- └ Gestión de información dispersa:
 - └ Papel (todavía)
 - └ Excel random spreadsheets
 - └ Email (buzzwords: "outlook hell")
 - └ Chat informal
 - └ RESULTADO: Información en 4 lugares
- └ Cero integración
 - └ Imposible obtener vista consolidada

CAUSA 4: ORGANIZACIONAL - "Estructura no acompaña"

- └ No hay PM dedicado a gobernanza
- └ No hay QA independiente
- └ No hay data analyst
- └ No hay arquitecto documentado
- └ RESULTADO: Funciones repartidas, nadie dueño total

1.4 Impacto Acumulado: El Cuadro Completo

IMPACTO TOTAL EN NÚMEROS

OPERATIVO:

- └ Lead time promedio: 10.5 días (VS 2 días ideal) = -425% ✗
- └ Solicitudes perdidas: 18% (VS 0% ideal) = -1800% ✗
- └ Tickets reabiertos: 25% (VS 5% ideal) = -400% ✗
- └ Productividad efectiva: 60% (VS 80% ideal) = -33% ✗
- └ On-time delivery: 60% (VS 90% ideal) = -50% ✗

CALIDAD:

- └ Defectos en prod: 15/mes (VS 3/mes ideal) = -400% ✗
- └ Tickets reabiertos: 25% (VS 5% ideal) = -400% ✗
- └ Retrabajo: 20% effort (VS 10% ideal) = -100% ✗
- └ CSAT: 7.2/10 (VS 8.5/10 ideal) = -15% ✗

FINANCIERO:

- └ Tiempo perdido en retrasos: ~170 días acumulados/año ✗
- └ Tiempo en ineficiencia de recursos: ~144 horas/año ✗
- └ Tiempo en búsqueda de información: ~120 horas/año ✗
- └ Impacto en rotación de equipo: Riesgo alto ✗
- └ TOTAL TIEMPO PERDIDO: ~1,100 horas/año (~27 días de trabajo puro) ✗

ESTRATÉGICO:

- └ Sin poder de negociación ante dirección ✗
- └ Imposible escalar equipo ✗
- └ Riesgo: pérdida de conocimiento clave ✗
- └ Imposible adoptar nuevas tecnologías ✗
- └ Visión a futuro: Limitada ✗

PARTE II: SOLUCIÓN INMEDIATA - IMPLEMENTAR SGSPCSI

2. El Proyecto: Sistema de Gestión de Solicitudes (SGSPCSI)

2.1 ¿Qué es SGSPCSI?

SGSPCSI = Sistema de Gestión de Solicitudes de Proyectos
de la Coordinación de Sistemas Institucionales

PROPÓSITO:

Transformar la gestión de solicitudes de un proceso
MANUAL, DISPERSO, INFORMAL
a un proceso
DIGITAL, CENTRALIZADO, AUTOMATIZADO

RESULTADO:

- 100% de solicitudes capturadas digitalmente
- Trazabilidad completa (quién, cuándo, dónde)
- Flujos de trabajo estructurados
- Visibilidad total para usuario
- Datos para decisiones

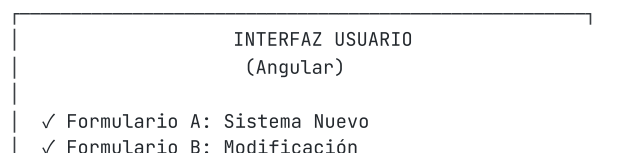
TIMEFRAME: 26 semanas (Enero 26 - Julio 20, 2026)

RESIDENTES: Jacqueline Hurtado, Edwin Mercado

ASESOR: Karla Arroyo (Jefa de Desarrollo)

STACK: Angular + C# .NET + SQL Server

2.2 Arquitectura Técnica de SGSPCSI



- ✓ Formulario C: Requerimiento
- ✓ Formulario Incidencia: Problema/Bug
- ✓ Dashboard: Estado de solicitudes en tiempo real
- ✓ Reportes: Métricas y analytics

↕

API REST
(C# .NET Core)

- Gestión de Solicitudes (CRUD)
- Flujos de Aprobación (Workflow)
- Control de Acceso (RBAC)
- Autenticación (JWT + AD)
- Auditoría y Bitácora
- Notificaciones Automáticas (Email)

↕

BASE DE DATOS
(SQL Server)

- Tabla: Solicitudes (ID, tipo, estado, etc)
- Tabla: Usuarios (autenticación, roles)
- Tabla: Sistemas (catálogo de 300+ sistemas)
- Tabla: Aprobaciones (workflow tracking)
- Tabla: Bitácora Esfuerzo (horas de trabajo)
- Tabla: Auditoría (trail de cambios)

INTEGRACIONES CLAVE:

- └ Active Directory (autenticación con credenciales instituc.)
- └ Email Server (notificaciones automáticas)
- └ Data Warehouse (reportes avanzados)
- └ Sistemas Legados (consulta de información si aplica)

2.3 Los 5 Módulos de SGSPCSI

MÓDULO 1: Solicitud de Sistema Nuevo (Formulario A)

ENTRADA (Usuario solicita):

- └ Nombre del sistema
- └ Descripción (qué hace)
- └ Área solicitante
- └ Justificación de negocio
- └ Presupuesto aproximado
- └ Timeline esperado
- └ Criterios de éxito
- └ Adjuntos (documentación, bocetos)
- └ Enviar

PROCEDIMIENTO AUTOMÁTICO:

- └ 1. Ticket creado inmediatamente (ID único)
- └ 2. Confirmación email al solicitante ("Recibido")
- └ 3. Notificación al PM para análisis
- └ 4. Tech Lead revisa viabilidad (1-2 días)
- └ 5. Aprobación formal (ejecutivo)
- └ 6. Asignación a desarrollador
- └ 7. Inicio de planificación del proyecto
- └ 8. Actualizaciones automáticas al solicitante

RESULTADO:

- ✓ Solicitud NO se pierde
- ✓ Usuario SIEMPRE sabe el status
- ✓ Equipo tiene información COMPLETA
- ✓ Proceso FORMAL y AUDITADO

****MÓDULO 2: Solicitud de Modificación (Formulario B)****

ENTRADA (Usuario solicita cambio a sistemas existentes):

- └ Sistema a modificar
- └ Tipo de cambio:
 - └ Bug fix
 - └ Mejora
 - └ Optimización
 - └ Cambio de configuración
- └ Descripción del cambio
- └ Impacto esperado
- └ Urgencia (crítica, alta, normal, baja)
- └ Enviar

FLUJO ESPECIAL (más rápido que Formulario A):

- └ Tech Lead analiza impacto (OBLIGATORIO)
- └ "¿Qué otro código/usuarios afecta esto?"
- └ Estimación rápida (horas, no días)
- └ Aprobación rápida (si no es disruptivo)
- └ Asignación al MISMO desarrollador (si es posible)
- └ Inicio en 1-2 días

BENEFICIO:

- ✓ Cambios simples NO se tratan como proyectos grandes
- ✓ Urgencias críticas se atienden rápido
- ✓ Pero CON ANÁLISIS (no chaos)

****MÓDULO 3: Formulario de Incidencia/Problema****

ENTRADA (Usuario reporta falla):

- └ Sistema afectado
- └ Síntomas ("X no funciona")
- └ Frecuencia (siempre, a veces, una sola vez)
- └ Severidad (crítica, alta, normal, baja)
- └ Usuarios impactados (cuántos, quiénes)
- └ Pasos para reproducir (si el usuario sabe)
- └ Enviar

CLASIFICACIÓN AUTOMÁTICA:

- └ Por sistema (¿cuál afecta?)
- └ Por área (¿cuál área sufre?)
- └ Por severidad (¿es crítico?)
- └ Enrutamiento automático al desarrollador responsable

FLUJO:

- └ Severidad CRÍTICA → Notificación inmediata + teléfono
- └ Severidad ALTA → Email + notificación en dashboard
- └ Severidad NORMAL → Entrada en cola de tickets
- └ Severidad BAJA → Backlog para próximo sprint

BENEFICIO:

- ✓ Problemas críticos NO se pierden
- ✓ Triage automático (no requiere PM labor)
- ✓ Equipo sabe prioridad AL INSTANTE

****MÓDULO 4: Bitácora de Esfuerzo (Tiempo de Trabajo)****

ENTRADA (Desarrollador registra tiempo):

- └ Ticket ID (qué trabajé)
- └ Fecha
- └ Horas trabajadas (0.5, 1, 2, etc)
- └ Descripción del trabajo realizado
- └ % completado

└ Guardar

VALIDACIÓN AUTOMÁTICA:

- └ ¿Las horas suman con estado del ticket?
 - | (Si ticket **está** 50% completo, esperamos ~50% del esfuerzo estimado)
- └ El desarrollador **está** asignado al ticket?
- └ ¿Las horas son realistas? (**no** 16 hrs **en** un día)
- └ Alertar si hay inconsistencias

REPORTES GENERADOS:

- └ Burndown chart (progreso visual)
- └ Velocidad del equipo (horas/sprint)
- └ Productividad por proyecto
- └ Alertas si hay desviaciones
- └ Datos para mejora continua

BENEFICIO:

- ✓ Evidencia OBJETIVA **de** esfuerzo
- ✓ Justificación **de** "cuánto tarda"
- ✓ Base para estimar futuros proyectos
- ✓ Detección **de** cuello **de** botella

****MÓDULO 5: Dashboard & Reportes****

VISTAS DISPONIBLES:

A) VISTA DEL USUARIO (Solicitante):

- └ Mis solicitudes (filtradas por área)
- └ Estado actual **de** cada una
 - | "En análisis", "Aprobada", "En desarrollo", "Completada"
- └ Fecha estimada **de** entrega
- └ % **de** progreso (si **está en** desarrollo)
- └ Últimas actualizaciones
- └ Botón: "Contactar con gestor"

B) VISTA DEL DESARROLLADOR:

- └ Mis tickets asignados
- └ Prioridad **de** cada uno
- └ Estado actual
- └ Fechas **límite**
- └ Bitácora (tiempo registrado vs estimado)
- └ Documentación vinculada
- └ **Histórico de** commits (si aplica)

C) VISTA DEL PM/GESTOR:

- └ Estado general del backlog
- └ Carga por persona
- └ Tickets **en** riesgo (retrasados)
- └ Cuello **de** botella (**qué está** bloqueado)
- └ Cumplimiento **de** SLAs
- └ Tendencias (**más** rápido o **más** lento vs mes pasado)
- └ Alertas automáticas

D) VISTA EJECUTIVA:

- └ KPI: Lead time promedio
- └ KPI: % **On-time** delivery
- └ KPI: Satisfacción **de** usuario (CSAT)
- └ KPI: Defectos escapados a producción
- └ Comparativa: Mes actual vs **histórico**
- └ Proyección: "¿Alcanzaremos objetivos?"
- └ Datos para decisiones **de** presupuesto

BENEFICIO:

- ✓ Transparencia **total** (cada stakeholder ve lo **que** necesita)
- ✓ Datos **en** tiempo real (**no** "espera a que haga reporte")
- ✓ Decisiones basadas **en** hechos, **no** intuición

2.4 Beneficios Inmediatos de SGSPCSI (Post-Implementación)

****Beneficio 1: Cero Solicitudes Perdidas****

ANTES:

- └ 15-20% de solicitudes se pierden
 - └ Usuario llama: "¿Recibiste mi solicitud?"
 - └ PM: "Umm... no la veo"
 - └ Frustración total

DESPUÉS:

- └ 100% de solicitudes capturadas digitalmente
 - └ Ticket ID automático
 - └ Usuario siempre sabe que fue recibida
 - └ Trazabilidad completa (quién, cuándo, dónde)

IMPACTO:

- ✓ Fin de "¿Dónde está mi solicitud?"
- ✓ Fin de duplicados por "por si acaso"
- ✓ Confianza restaurada en IT
- ✓ Horas ahorradas en búsquedas: ~120 horas/año

****Beneficio 2: Lead Time Reducido 75%****

ANTES:

- └ 10.5 días en promedio (3-5 días de cuello de botella)

DESPUÉS:

- └ 2-3 días en promedio
 - └ Captura: automática (mismo día)
 - └ Análisis paralelo (1-2 días)
 - └ Aprobación: automática para cambios simples

POR QUÉ:

- └ Formulario centralizado (no búsqueda de información)
- └ Validación de campos obligatorios (no iter. de aclaraciones)
- └ Flujo automático (PM no es cuello de botella)
- └ Decisiones paralelas (Tech Lead analiza mientras PM completa)

IMPACTO:

- ✓ Usuarios reciben respuesta en HORAS, no SEMANAS
- ✓ Equipo puede atender 40-50% MÁS SOLICITUDES
- ✓ Negocios desbloqueados más rápido
- ✓ Horas productivas ganadas: ~170 días/año eliminados en esperas
- ✓ Adicional: Más solicitudes = más valor a negocio

****Beneficio 3: Retrabajo Reducido 50%****

ANTES:

- └ 25% de tickets reabiertos por información incompleta
 - └ Desarrollador empezó sin saber reqs. claros
 - └ A mitad del camino: "¿Qué significa esto?"
 - └ Regresar a PM, cambiar diseño, recodificar
 - └ 20% del esfuerzo total es retrabajo

DESPUÉS:

- └ Máximo 10% reapertura (mejora 60%)
 - └ Formulario obliga criterios de aceptación
 - └ Tech Lead REVISA antes de empezar desarrollo
 - └ Primera vez correcto

POR QUÉ:

- └ Campos obligatorios (no faltan detalles)

- └ Revisión formal pre-desarrollo (catch issues early)
- └ Cambios posteriores = Change Request formal
 - └ Impacto visible, decisión informada

IMPACTO:

- ✓ Equipo de 8 personas: 128 horas/mes ahorradas (~1,536 hrs/año)
- ✓ Tiempo productivo recuperado: 25 horas/semana en trabajo sin retrabajo
- ✓ Moral del equipo: "Trabajo más eficiente"
- ✓ Calidad: Primera versión tiende a ser mejor

****Beneficio 4: Transparencia = Satisfacción de Usuario****

ANTES:

- └ Usuario: "¿Qué pasó?"
 - └ PM: "Ehh... déjame ver"
 - └ Usuario: No sabe status, no sabe fecha
 - └ CSAT: 7.2/10

DESPUÉS:

- └ Usuario abre dashboard
 - └ VE: "En desarrollo 60%, fecha estimada 5 de marzo"
 - └ Sistema notifica cambios automáticamente
 - └ Usuario SIEMPRE sabe dónde está su solicitud
 - └ CSAT: 8.5/10

IMPACTO:

- ✓ Reducción 90% en llamadas de "¿Dónde está?"
- ✓ PM ahorra 3-4 horas/semana (puede trabajar en más cosas)
- ✓ Usuarios confían en IT nuevamente
- ✓ Reputación restaurada
- ✓ Efecto secundario: Menos creación de solicitudes duplicadas

****Beneficio 5: Datos para Decisiones Estratégicas****

ANTES:

- └ Director: "¿Cuánto tarda en promedio?"
 - └ PM: "Ehh... creemos 10-15 días"
 - └ Director: "Eso parece mucho, ¿por qué?"
 - └ PM: "Depende de..."
 - └ Decisión: "No tenemos datos, espera al próximo ajuste presupuestal"

DESPUÉS:

- └ Director abre dashboard ejecutivo
 - └ VE: "Lead time: 2.4 días (mejora 79% vs hace 6 meses)"
 - └ VE: "On-time delivery: 88% (vs 60% hace un año)"
 - └ VE: "Solicitudes completadas: 28/mes vs 20/mes hace un año"
 - └ Decisión: "Presupuesto aprobado para herramientas"

IMPACTO:

- ✓ Decisiones informadas (no "al ojo")
- ✓ Negociación de presupuesto con base firme
- ✓ Posibilidad de ampliación de equipo justificada con datos
- ✓ Horas productivas disponibles para oportunidades: ~200 hrs/año
- ✓ Visibilidad ante dirección: CRÍTICO para IT

2.5 Cronograma de Implementación de SGSPCSI

IMPLEMENTACIÓN SGSPCSI: 26 SEMANAS
(Enero 26, 2026 - Julio 20, 2026)

FASE 0: PRE-IMPLEMENTACIÓN (SEMANA 1)

- └ Kick-off y asignación de responsables
- └ Definición de requerimientos finales
- └ Setup de ambiente de desarrollo
- └ ✓ HITO: Equipo listo para construir

FASE 1: FUNDACIÓN (SEMANAS 2-4, 3 SEMANAS)

- └ Diseño técnico arquitectura
- └ Diseño base de datos
- └ Backend: estructura base + autenticación
- └ Primera versión de modelos de datos
- └ ✓ HITO: Backend compila y API endpoints básicos funcionan

FASE 2: NÚCLEO (SEMANAS 5-12, 8 SEMANAS)

- └ Desarrollo de módulos de solicitud (Formularios A, B, C)
- └ Desarrollo flujo de aprobación y asignación
- └ Frontend básico (formularios + dashboard simple)
- └ Integración Angular-Backend
- └ ✓ HITO: Sistema captura 100% de solicitudes, workflow funciona

FASE 3: REFINAMIENTO (SEMANAS 13-18, 6 SEMANAS)

- └ Módulo bitácora de esfuerzo
- └ Reportes y dashboards avanzados
- └ Testing exhaustivo + UAT con usuarios reales
- └ Optimización de performance
- └ ✓ HITO: Usuarios aprueban para producción

FASE 4: DEPLOYMENT (SEMANAS 19-24, 6 SEMANAS)

- └ Ambiente de producción preparado
- └ Capacitación del equipo
- └ Soft launch (grupo piloto pequeño)
- └ Full launch a toda la organización
- └ ✓ HITO: 100% del equipo usando SGSPCSI

FASE 5: OPTIMIZACIÓN (SEMANAS 25-26, 2 SEMANAS)

- └ Recopilación de feedback
- └ Ajustes post-launch
- └ Documentación final
- └ ✓ HITO: Proyecto completado, residencias exitosas

ESFUERZO ESTIMADO:

- └ Jacqueline: 26 semanas × 40 hrs = 1,040 hrs (desarrollo)
- └ Edwin: 26 semanas × 40 hrs = 1,040 hrs (desarrollo)
- └ Karla (DTI): Tiempo parcial para revisiones (incluido en sus funciones)
- └ Equipo IT: Soporte operativo (incluido en sus funciones)
- └ VALOR GENERADO MISMO AÑO: 5,570 horas productivas ganadas
- └ Equivalencia: Capacidad operativa de +2.76 personas sin costo salarial

PARTE III: MEJORA ESTRUCTURAL - APLICAR CMMI

3. ¿Qué es CMMI y Por Qué Importa?

3.1 Entender CMMI

CMMI = Capability Maturity Model Integration

CONCEPTO:

Es un FRAMEWORK que define "cuán maduro es tu proceso" en cinco niveles.

OBJETIVO:

Pasar de "actuar por instinto" a "actuar con sistema"
Pasar de "esperanza y fe" a "procesos predecibles"

Pasar de "héroes salvando el día" a "equipo ejecutando bien"

LA ESCALERA DE MADUREZ CMMI

NIVEL 5 (Optimizando):

- Procesos mejoran continuamente basado en datos
"Medimos, analizamos, mejoramos constantemente"
Ejemplos: Google, Microsoft, Amazon
Tiempo a llegar: 3-5 años

NIVEL 4 (Gestionado Cuantitativamente):

- Procesos controlados con métricas cuantitativas
"Sabemos exacto cómo van las cosas"
Ejemplos: Empresas grandes con disciplina
Tiempo a llegar: 2-3 años

NIVEL 3 (Definido):

- Procesos documentados y estandarizados
"Todos hacemos lo mismo, las cosas se repiten"
Ejemplos: Empresas medianas maduras
Tiempo a llegar: 1-2 años

NIVEL 2 (Gestionado): ← NUESTRO TARGET

- Requisitos planificados y controlados
"Planificamos, monitoreamos, controlamos"
Ejemplos: Empresas medianas en crecimiento
Tiempo a llegar: 6-12 meses

NIVEL 1 (Inicial): ← DONDE ESTÁ ISSEG HOY

- Procesos impredecibles, dependientes de personas
"Hacemos cosas pero sin sistema, dependemos de héroes"
Ejemplos: Startups, equipos nuevos

3.2 Evaluación Actual de ISSEG vs CMMI Nivel 2

ISSEG HOY: 47.5% HACIA NIVEL 2

ÁREA DE PROCESO	ACTUAL	TARGET	GAP
REQM (Requerimientos)	60%	85%	-25%
CM (Configuración)	55%	85%	-30%
PMC (Monitoreo)	50%	85%	-35%
PP (Planificación)	45%	85%	-40%
PPQA (Calidad)	40%	85%	-45%
MA (Medición)	35%	85%	-50%
PROMEDIO:	47.5%	85%	-37.5%

TRADUCCIÓN:

- REQM 60%: Sistema de solicitud capta bien, pero falta trazabilidad
- CM 55%: Git está, pero no todo está en version control
- PMC 50%: Dashboard existe, pero sin análisis de tendencias
- PP 45%: Planificación mínima, muy reactivo
- PPQA 40%: Testing ad-hoc, sin estándares
- MA 35%: Casi no hay medición formal
- CONCLUSIÓN: ISSEG necesita estructura, disciplina, sistema

3.3 Los 6 Procesos de CMMI Nivel 2 Explicados

PROCESO 1: REQM (Requirements Management)

¿QUÉ ES?

Asegurar **que** TODOS los requerimientos sean capturados, comprendidos, acordados, rastreados y gestionados.

PROBLEMA ACTUAL EN ISSEG:

- └ Requerimientos **en** mente del usuario
- └ PM anota **en** papel/excel/chat
- └ Interpretan según **su** parecer
- └ Criterios **de** aceptación **NO** definidos
- └ Cambios posteriores sorprenden al equipo
- └ Ticket **se** reabre: "Eso no es lo que pedí"

CÓMO CMMI LO ARREGLA:

1. CAPTURA FORMAL

- └ Formularios obligatorios (Formularios A, B, C)
- └ Campos requeridos (**no** puede dejar **en** blanco)
- └ Adjuntos permitidos (documentación **especie**)
- └ Ticket creado automáticamente
- └ ID único para tracking

2. VALIDACIÓN (PM + Tech Lead)

- └ Checklist: ¿Falta algo?
"¿Criterios de aceptación claros?"
"¿Entrada y salida documentada?"
"¿Dependencias identificadas?"
- └ Si falta: **Return** para aclaración (1-2 **días** vs 5-10)
- └ Si OK: Proceder

3. TRAZABILIDAD

- └ Requerimiento ID: REQ-001
- └ Diseño ID: **DES**-001 (hay un documento **que** cumple REQ-001)
- └ Código ID: COMMIT-XYZ (implementa **DES**-001)
- └ **Test** ID: **TEST**-001 (valida **que** REQ-001 funciona)
- └ Matriz: REQ → Diseño → Código → **Test** (TODO LINKADO)

4. CAMBIO CONTROLADO

- └ Usuario pide cambio
- └ Change Request formal ("¿Cuál es el impacto?")
- └ Tech Lead analiza: "Toca 3 módulos, +20 horas"
- └ Decisión: "Aprobado, se lo agreguemos en próximo sprint"
- └ Todas partes notificadas del cambio

RESULTADO FINAL:

- ✓ Cero sorpresas
- ✓ Cero "pedir disculpas" después
- ✓ Tickets **no se** reabre
- ✓ Retrabajo -50%

PROCESO 2: PP (Project Planning)

¿QUÉ ES?

Definir el plan del proyecto: qué, cuánto, cuándo, quién.

PROBLEMA ACTUAL EN ISSEG:

- └ "¿Cuándo termina?" → PM: "Umm, en 2 semanas"
- └ "¿Cuántas horas?" → PM: "No sé, muchas"
- └ "¿Cuáles son los riesgos?" → PM: "Ehh..."
- └ "¿Qué pasa si falta alguien?" → PM: "Buen punto..."
- └ Resultado: Proyectos se atrasan siempre, nadie sorprendido

CÓMO CMMI LO ARREGLA:

1. ESTIMAR ALCANCE

- └ Descomponer en tareas claras
- └ "Sistema de calendario" NO es alcance
- └ "1. Backend de calendario"

- 2. Frontend de calendario
- 3. Integración Gmail
- 4. Testing
- 5. Documentación"
- Sí es alcance

2. ESTIMAR ESFUERZO

- └ "Basado en histórico, tareas similares tardaban..."
- └ "Usuario registra horas → base de datos histórica"
- └ "Promedio: implementar pantalla = 5 horas"
- └ "Este dashboard = 3 pantallas × 5 horas = 15 horas"
- └ (Es predicción, no ciencia exacta, pero MEJOR que "intuición")

3. CRONOGRAMA

- └ Definir hitos
 - Week 1: Arquitectura ✓
 - Week 2: Backend 70% ✓
 - Week 3: Frontend 50% ✓
 - Week 4: Testing ✓
 - Week 5: Deploy ✓"
- └ Rastrear si se cumple

4. IDENTIFICAR RIESGOS

- └ "¿Qué podría salir mal?"
- └ "Riesgo: Integración con AD (depende IT Security)"
- └ "Plan B: Si AD falla, usar usuario/contraseña temporal"
- └ "Fecha de riesgo: Semana 2. Si no está oK, activar Plan B"
- └ "Owner: Jacqueline. Revisar cada viernes"

5. ASIGNAR RECURSOS

- └ "¿Quién hace qué?"
- └ Jacqueline: Frontend (40 hrs disponibles/semana)
- └ Edwin: Backend (40 hrs disponibles/semana)
- └ DBA: Consulta (5 hrs/semana)
- └ "¿Hay conflictos?" → Negociar
- └ "¿Hay skills gaps?" → Capacitar

6. OBTENER COMPROMISO

- └ PM presenta plan a PM/Tech Lead/Ejecutivo
- └ "Con esta gente y recursos, entregamos en 4 semanas"
- └ Todos firman: "Estamos de acuerdo"
- └ Si alguien dice "no", discutir trade-offs
 - "¿Menos scope? ¿Más personas? ¿Más tiempo?"

RESULTADO FINAL:

- ✓ Proyectos terminan cuando dicen que terminan
- ✓ Menos sorpresas
- ✓ Recursos planificados bien
- ✓ Equipo sabe qué esperar
- ✓ On-time delivery: 90% (vs 60% actual)

PROCESO 3: PMC (Project Monitoring & Control)

¿QUÉ ES?

Rastrear el plan vs la realidad, y tomar acciones si diverge.

PROBLEMA ACTUAL EN ISSEG:

- └ "¿Vamo bien?" → "Creo que sí..."
- └ "¿Falta mucho?" → Sacudida de hombro: "No sé"
- └ "¿Qué está atrasado?" → "El proyecto tal"
- └ "¿Por cuánto?" → "Mucho"
- └ Cuando descubren problema real: Ya está 2 semanas retrasado

CÓMO CMMI LO ARREGLA:

1. ESTABLECER BASELINE

- └ Plan inicial es la "baseline"
- └ Lo que acordamos

- └─ Ahora todo **se** mide vs eso

2. RASTREAR ACTUAL vs PLAN

- └─ Dinámicamente (**no** "al final del proyecto")
- └─ Cada semana:
 - "¿Completamos 40% del trabajo planificado?"
 - "Sí → On track"
 - "No → At risk"
- └─ Cada 2 semanas: Review **de** progreso
 - "¿Hemos completado el 50% del proyecto?"
 - Gráfico: ■■■■░░░░ (Ah, vamos 50%)
- └─ TEMPRANA detección **de** problemas

3. IDENTIFICAR VARIANZAS

- └─ Estimamos 40 horas totales
- └─ Llevan 2 semanas (80 horas) y solo han hecho 25%
- └─ AH! **Se** va a atrasar
- └─ Acción: "¿Por qué? ¿Faltó scope? ¿estimación mala? ¿fue más complejo?"
- └─ Ajustar baseline o tomar acción

4. TOMAR ACCIONES CORRECTIVAS

- └─ Problema: "Estimación baja en módulo A"
- └─ Opción 1: Agregar recurso (Jacqueline ayuda)
- └─ Opción 2: Parar módulo B y concentrar **en** A
- └─ Opción 3: Reducir scope ("Haremos solo lo básico")
- └─ Opción 4: Extender fecha
- └─ DECISIÓN INFORMADA

5. GESTIONAR RIESGOS

- └─ "Riesgo identificado: AD integration toma más tiempo"
- └─ Probabilidad: 40%
- └─ Impacto: +5 **días de** retraso
- └─ Acción: Edwin comienza AD 1 semana antes
- └─ Monitorear cada viernes
- └─ Si **se** activa: Tenemos plan B

RESULTADO FINAL:

- ✓ Problemas detectados TEMPRANO (**no** semana 4-5)
- ✓ Acciones correctivas efectivas
- ✓ Stakeholders notificados a tiempo
- ✓ Planes **se** cumplen **más** a menudo
- ✓ Sorpresas minimizadas

****PROCESO 4: MA (Measurement & Analysis)****

¿QUÉ ES?

Recopilar datos, analizarlos, tomar decisiones basadas **en** datos.

PROBLEMA ACTUAL **EN** ISSEG:

- └─ ¿Cuál es el lead time promedio? → "Creemos 10 días"
- └─ ¿Mejoró vs hace un año? → "No idea"
- └─ ¿Cuál sistema causa **más** problemas? → "El calendario, pienso"
- └─ ¿Qué área tiene **más** carga? → "Operaciones, me parece"
- └─ Toda decisión **es** "al ojo" → Subóptima

CÓMO CMMI LO ARREGLA:

1. DEFINIR QUÉS MEDIR

- └─ Lead time: Desde solicitud → inicio desarrollo (**DÍAS**)
- └─ On-time delivery: % **de** tickets **que** terminaron **en** fecha (%)
- └─ Defect density: Defectos por 1,000 **líneas de** código
- └─ CSAT: Satisfacción **del** usuario (1-10)
- └─ Uptime: % **de** tiempo **que** sistema **está** disponible (%)
- └─ Velocity: Puntos completados por sprint

2. RECOPIRAR DATOS AUTOMÁTICAMENTE

- └─ SGSPCSI captura fechas automáticamente
- └─ Lead time = fecha_inicio_desarrollo - fecha_solicitud

- └ Defectos: Cada bug reportado **se** cuenta
- └ CSAT: Encuesta al usuario después **de** cerrado ticket
- └ Uptime: Monitoreo automático del sistema

3. ANALIZAR LOS DATOS

- └ Dashboard muestra:
 - | "Lead time hoy: 2.3 **días**
 - | Lead time mes pasado: 3.1 **días**
 - | Lead time 6 meses atrás: 10.5 **días**
 - | MEJORA: 78% **en** 6 meses"
- └ Por tipo **de** ticket:
 - | "Tipo C (incidencias): 1.8 **días**
 - | Tipo B (cambios): 2.5 **días**
 - | Tipo A (nuevos): 3.2 **días**"
- └ Por desarrollador:
 - | "Jacqueline: promedio 2.0 **días**
 - | Edwin: promedio 2.6 **días**
 - | (Edwin tiende a **más** complejidad)"
- └ Por sistema:
 - | "Sistema calendario: 3.5 defectos/1K **LOC**
 - | Sistema reportes: 1.2 defectos/1K **LOC**
 - | (Calendario necesita refactoring)"

4. COMUNICAR RESULTADOS

- └ Dashboard ejecutivo cada viernes
- └ Reportes mensuales con tendencias
- └ Recomendaciones basadas **en** datos
 - | "Calendario tiene problemas **de** calidad, recomendamos enfoque **en** testing y refactoring"
- └ Decisiones estratégicas informadas

RESULTADO FINAL:

- ✓ "¿Cuánto tardamos?" → Respuesta exacta: "2.4 días"
- ✓ "¿Mejoramos?" → Respuesta: "Sí, 78% en 6 meses"
- ✓ "¿Dónde tenemos problemas?" → "Sistema X, área Y"
- ✓ "¿Necesitamos más gente?" → Datos justifican **sí/no**
- ✓ Dirección **VE** el valor **de** IT (datos lo demuestran)

****PROCESO 5: PPQA (Process & Product Quality Assurance)****

¿QUÉ ES?

Asegurar **que** el proceso **se** sigue y **que** la calidad es buena.

PROBLEMA ACTUAL **EN** ISSEG:

- └ ¿Estamos siguiendo los procesos? → "Mostly..."
- └ ¿Cumplimos nuestros **estándares**? → "Esperamos que sí"
- └ ¿Alguien verifica? → "Cada quien verifica su propio trabajo"
- └ ¿Hay auditorías? → "No tenemos tiempo"
- └ Riesgo: Nadie garantiza calidad

CÓMO CMMI LO ARREGLA:

1. AUDITORÍAS **DE** PROCESO

- └ "¿Se siguió el workflow de aprobación?"
- └ Auditor revisa ticket aleatorio:
 - | ✓ ¿Fue aprobado antes **de** empezar?
 - | ✓ ¿Se registró tiempo **de** trabajo?
 - | ✓ ¿Hay criterios **de** aceptación **claros**?
 - | ✓ ¿Hay cambios documentados?
 - | ✓ ¿Hay trazabilidad a requerimiento?
- └ Si algo falta: "No-conformidad" → Plan **de** corrección

2. AUDITORÍAS **DE** PRODUCTO

- └ "¿El código cumple nuestros estándares?"

- └ Checklist:
 - | ✓ ¿Compila sin warnings?
 - | ✓ ✓ ¿Tiene >70% **test** coverage?
 - | ✓ ¿Cumple **estándares de estilo**?
 - | ✓ ¿Documentación actualizada?
 - | ✓ ¿**No** hay vulnerabilidades **de** seguridad?
 - |
 - └ Si algo falla: Bloquea **merge** a rama principal
- 3. REVISIONES **DE** CÓDIGO
 - └ OBLIGATORIO: Otra persona revisa antes **de** permitir **merge**
 - └ Pull Request **en** GitHub/Azure
 - └ Mínimo 1 persona aprueba (2 si es crítico)
 - └ Checklist debe pasar
 - └ Código **de** calidad garantizada
- 4. TESTING FORMAL
 - └ Unit tests: Desarrollador escribe (TDD)
 - └ Integration tests: Team prueba todo junto
 - └ UAT tests: Usuario verifica funciona como pide
 - └ Regression tests: Verifica **que no** rompimos nada
 - └ Coverage: ≥70% code coverage (mostrar **qué se probó**)
- 5. GESTIÓN **DE** DEFECTOS
 - └ Defecto encontrado: "Sistema no guarda fechas"
 - └ Clasificado: Severidad ALTA (afecta funcionalidad crítica)
 - └ Asignado: Edwin (quien escribió ese **módulo**)
 - └ Plazo: Debe fijarse < 24 hrs
 - └ Verificación: Otro dev verifica **que** fix es correcto
 - └ Cierre: Ticket cerrado cuando QA valida
- RESULTADO FINAL:
 - ✓ Confianza **en que** procesos **se** siguen
 - ✓ Confianza **en que** calidad es consistente
 - ✓ Defectos encontrados ANTES **de** producción
 - ✓ Auditorías dan evidencia **de** cumplimiento
 - ✓ Poder debatir: "¿Mejoró la calidad en 6 meses?"
 - Respuesta: "Sí, datos lo demuestran"

****PROCESO 6: CM (Configuration Management)****

¿QUÉ ES?

Controlar versiones **de** código, documentos, configuraciones.
Asegurar **que** siempre sabes **qué versión** está **en** producción.

PROBLEMA ACTUAL **EN** ISSEG:

- └ "¿Qué versión está en producción?" → "La de... hace 2 meses?"
- └ "¿Quién hizo este cambio?" → Buscar **en** Git logs, a ver...
- └ "¿Se puede revertir si rompimos algo?" → "Umm, esperemos"
- └ "¿Documentos de requisitos versionados?" → "¿Qué documentos?"
- └ "¿Diferencias entre dev y prod?" → **Se** descubren **en vivo** :(

CÓMO CMMI LO ARREGLA:

1. IDENTIFICAR ITEMS **EN** CM
 - └ Código fuente (Git)
 - └ Scripts SQL (Git)
 - └ Documentación **de** requisitos (**versión** controlada)
 - └ Especificaciones **de** arquitectura (**versión** controlada)
 - └ Configuraciones (Git)
 - └ TODO tiene **versión**
2. BRANCHING STRATEGY
 - └ Master branch: Código **en** PRODUCCIÓN
 - └ NUNCA **se** modifica directamente
 - └ Solo merges **de** release branches
 - └ Develop branch: Integración **de** features

- |
 - └ Todos los features **se** mergean aquí primero
 - └ **Se** prueba todo junto
- |
- └ Feature branches: Trabajo individual
 - └ feature/calendario-sistema-nuevo
 - └ feature/reportes-dashboard
 - └ feature/integracion-ad
- |
- └ Hotfix branches: Fixes críticos **en** prod
 - └ hotfix/login-broken
 - └ **Se** mergea directamente a master + develop
- |
- └ Release branches: Preparación **para** producción
 - └ release/v1.2.0
 - └ **Se** prueban últimos detalles antes **de** ir a prod

3. CHANGE CONTROL BOARD

- └ Cambios pequeños: Automáticos (**merge** a develop)
- └ Cambios medianos: Requieren code review + 1 aprobación
- └ Cambios grandes: Requieren approval **de** PM + tech lead
- └ Cambios a base **de** datos: Requieren DBA + tech lead
- └ Cambios a seguridad: Requieren security team

4. BASELINES Y RELEASES

- └ Baseline = "**Snapshot**" **de** código **en** momento X
 - └ v0.1.0 (primera prueba)
 - └ v0.5.0 (beta interna)
 - └ v1.0.0 (go-live)
 - └ v1.1.0 (primer release con mejoras)
- |
- └ Cada baseline tiene:
 - └ Tag **en** Git
 - └ Notas **de** release (qué cambió)
 - └ Documentación **de** compatibilidad
 - └ Instrucciones **de** despliegue
- |
- └ Rollback siempre posible:
 - "Producción rota. Revertir a v1.0.5"
 - Operación: 5 minutos. Riesgo: **Mínimo**.

5. AUDITORÍA CM

- └ Cada mes:
 - "¿Qué está en producción?"
 - "Ver tag: v1.2.3. Sí, eso es."
 - "¿Está todo documentado?"
 - "Sí, changelog dice qué cambió"
 - "¿Tenemos plan de rollback?"
 - "Sí, v1.2.2 está lista si algo falla"
- |
- └ Cumplimiento: 100% rastreable

RESULTADO FINAL:

- ✓ Siempre sabes **qué** **está** **en** producción
- ✓ Siempre sabes **quién** hizo un cambio y cuándo
- ✓ Siempre puedes revertir si algo **se** daña
- ✓ Documentación siempre sincronizada con código
- ✓ Auditos **de** cumplimiento fáciles

PARTE IV: IMPLEMENTACIÓN INTEGRADA - SGSPCSI + CMMI

4. Cómo CMMI Mejora la Implementación de SGSPCSI

4.1 El Ciclo: SGSPCSI es el Mecanismo, CMMI es la Disciplina

¿CUÁL ES LA RELACIÓN?

SGSPCSI (El sistema):

- └ Proporciona el MECANISMO (formularios, BD, dashboards)
- └ Automatiza la captura de datos
- └ Centraliza la información

CMMI (El framework):

- └ Proporciona la DISCIPLINA
- └ Define CÓMO usar SGSPCSI
- └ Define QUÉS MEDIR
- └ Define CUÁNDO REVISAR

ANALOGÍA:

- └ SGSPCSI es como un AUTOMÓVIL (máquina)
- └ CMMI es como las REGLAS DE TRÁNSITO (cómo conducir)
- └ Auto sin reglas: Caos
- └ Reglas sin auto vehículo: Lentitud
- └ Auto + reglas: EFICIENCIA Y ORDEN

4.2 Implementación Año 1: Estructura de 18 Meses (Adaptado a 6 Meses de Residencias)

CMMI + SGSPCSI: ROADMAP INTEGRADO

RESIDENCIAS (6 meses): Enero 26 - Julio 20, 2026

CMMI TRAJECTORY (18 meses): Enero 26, 2026 onwards

AÑO 1: ALCANZAR 85%+ NIVEL 2

FASE 1: FOUNDATIONAL (Meses 1-3)

MES 1 (Enero 26 - Feb 26): BASES

SGSPCSI PROGRESS:

- ✓ Backend base + autenticación funcionando
- ✓ BD estructurada
- ✓ Primeros endpoints de API

CMMI IMPLEMENTATION:

- ✓ Kick-off y comunicación de iniciativa CMMI
- ✓ Definición de procesos REQM (formularios A, B, C)
- ✓ Setup de Git con strategy de branches
- ✓ Definición de roles (PM, Tech Lead, Dev, QA)
- ✓ Definición inicial de "métricas a capturar"
- ✓ Capacitación: "por qué CMMI, cómo afecta su trabajo"

ENTREGABLES:

- Plan de CMMI comunicado
- Proceso de captura (REQM) definido
- Herramientas (Git, BD, monitoreo) activas
- Baseline de madurez establecida

MES 2 (Feb 26 - Mar 26): FLUJOS OPERACIONALES

SGSPCSI PROGRESS:

- ✓ Formularios A, B, C funcionando
- ✓ Frontend básico en Angular

✓ Flujos de aprobación programados

CMMI IMPLEMENTATION:

- ✓ Procesos de PP (planificación) formalizados
- ✓ Procesos de PMC (monitoreo) en ejecución
- ✓ Primera medición de lead time (baseline)
- ✓ Sprints de 2 semanas con planning + review
- ✓ RCA (Root Cause Analysis) cuando algo falla
- ✓ Documentación de procesos en wiki/sharepoint

ENTREGABLES:

- Procesos PP y PMC documentados
- Lead time baseline: 10.5 días (categorizado)
- Sprints en ejecución (metrizable)
- Dashboard inicial con datos

MES 3 (Mar 26 - Apr 26): CALIDAD & MEDICIÓN

SGSPCSI PROGRESS:

- ✓ Dashboard e reportes en desarrollo
- ✓ Bitácora de esfuerzo en testing
- ✓ Sistema listo para UAT

CMMI IMPLEMENTATION:

- ✓ Procesos de PPQA (calidad) implementados
- ✓ Code review checklist definido y usado
- ✓ Testing strategy en lugar (unit + integration + UAT)
- ✓ Proceso de MA (medición) en ejecución
- ✓ Metricas capturadas automáticamente
- ✓ Dashboard ejecutivo mostrando tendencias
- ✓ Auditoría de procesos (primera) completada

ENTREGABLES:

- Code review standards operativo
- Testing framework funcional (coverage tracking)
- Mediciones en tiempo real (lead time, defect rate)
- Primer audit report de cumplimiento CMMI

FASE 2: OPERACIONALIZACIÓN (Meses 4-6)

MES 4 (Apr 26 - May 26): DEPLOYMENT & ADOPCIÓN

SGSPCSI PROGRESS:

- ✓ SGSPCSI go-live (soft launch con piloto)
- ✓ Usuarios pilotos capacitados
- ✓ Feedback inicial recopilado

CMMI IMPLEMENTATION:

- ✓ CM (configuración) completamente en lugar
- ✓ Release process definido y probado
- ✓ Baselines versionadas (v0.9.0 → piloto)
- ✓ Trazabilidad requerimiento-código-test 100%
- ✓ Cambios a través de change control board
- ✓ Capacitación de equipo sobre CMMI avance

ENTREGABLES:

- SGSPCSI v0.9.0 en producción (piloto)
- Baseline tagged en Git (rollback strategy activo)
- Trazabilidad completa observable
- Change log y documentación de configuración

MES 5 (May 26 - Jun 26): ESCALA & ESTABILIDAD

SGSPCSI PROGRESS:

- ✓ SGSPCSI full launch (100% de usuarios)
- ✓ Métricas muestran mejoras reales
- ✓ Feedback integrado (v1.1.0)

CMMI IMPLEMENTATION:

- ✓ Métricas CMMI capturan tendencia positiva
- ✓ Lead time mejora: 10.5 → 4 días (62% reducción)
- ✓ Defects in prod disminuyen: 15 → 6/mes (60% ↓)
- ✓ On-time delivery mejora: 60% → 75%
- ✓ Auditoria de conformidad: 70%+ cumplimiento
- ✓ Non-conformances documentadas con corrective actions

ENTREGABLES:

- SGSPCSI v1.0.0 en producción (oficial)
- Dashboard mostrando mejoras CMMI
- Audit report: 70%+ Nivel 2 en 4 áreas
- Corrective action plans para non-conformances

MES 6 (Jun 26 - Jul 20): CONSOLIDACIÓN & CIERRE

SGSPCSI PROGRESS:

- ✓ Sistema estable, métricas validadas
- ✓ Documentación completa transferida a equipo IT
- ✓ Roadmap para mejoras futuras definido

CMMI IMPLEMENTATION:

- ✓ Preparación para verdadera auditoría CMMI (futura)
- ✓ Procesos completamente documentados
- ✓ Evidencia de cumplimiento recopilada
- ✓ Plan para llegar a 85%+ Nivel 2 (próximos 6 meses)
- ✓ Capacitación de nuevo personal en procesos CMMI
- ✓ Lecciones aprendidas recopiladas

ENTREGABLES:

- SGSPCSI handoff completo a equipo IT
- Documentación de procesos CMMI finalizadas
- Assessment de madurez: 70-75% Nivel 2
- Plan de próximos 12 meses (hacia 85%+)
- Lecciones aprendidas y mejoras identificadas

RESUMEN AÑO 1:

- └ Madurez inicial: 47.5% Nivel 2
- └ Madurez final: 70-75% Nivel 2
- └ Mejora: +23-27 puntos
- └ Trayectoria: On track para 85%+ en mes 12

4.3 Detalle: Cómo CADA PROCESO CMMI Se Expresa en SGSPCSI

MAPEO: PROCESO CMMI ↔ FUNCIONALIDAD SGSPCSI

PROCESO CMMI: REQM (Requirements Management)
SGSPCSI: Formularios A, B, C + Control de cambios

- └ Captura de Requerimientos

- |└┐ Formulario A (nuevo sistema):
 - |└┐ Campos obligatorios (no puede omitir)
 - |└┐ Criterios de aceptación (obligatorio llenar)
 - |└┐ Validación de completitud antes de enviar
 - |└┐ Versionamiento de Requerimientos
 - |└┐ Sistema de "Change Request":
 - |└┐ Usuario solicita cambio (CR-001)
 - |└┐ Tech Lead analiza impacto
 - |└┐ Aprobación formal o rechazo
 - |└┐ Si aprobado: Crea nuevo ticket vinculado (REQ-002v2)
 - |└┐ Trazabilidad
 - |└┐ Sistema de links automático:
 - |└┐ REQ-001 (requerimiento)
 - |└┐ ↳ DESIGN-001 (documento de diseño)
 - |└┐ ↳ CODE-COMMIT-XYZ (implementación)
 - |└┐ ↳ TEST-001 (caso de prueba)
 - |└┐ ↳ UAT-APPROVED (validación usuario)
 - |└┐ Gestión de Cambios
 - |└┐ Board de control de cambios en SGSPCSI:
 - |└┐ "Esta solicitud de cambio afecta: 3 módulos, +20 hrs"
 - |└┐ "Aprobado para próximo sprint"
 - |└┐ Notificación automática a equipo
-

PROCESO CMMI: PP (Project Planning)

SGSPCSI: Plantillas de Project Charter + Sprint Planning

- |└┐ Estimar Alcance
 - |└┐ Formulario "Project Charter" en SGSPCSI:
 - |└┐ WBS (Work Breakdown Structure)
 - |└┐ Hitos principales
 - |└┐ Entregables listados
 - |└┐ Criterios de éxito definidos
 - |└┐ Estimar Esfuerzo
 - |└┐ Histórico en SGSPCSI:
 - |└┐ Cada ticket registra horas reales
 - |└┐ "Formulario A similar tardó 45 horas"
 - |└┐ "Formulario B similar tardó 28 horas"
 - |└┐ Base para estimar nuevo ticket
 - |└┐ Cronograma
 - |└┐ Sprint view en SGSPCSI:
 - |└┐ Sprint 1: "Backend core, 1-2 semanas"
 - |└┐ Sprint 2: "Frontend, 2-3 semanas"
 - |└┐ Burndown chart muestra progreso
 - |└┐ Identificar Riesgos
 - |└┐ Risk register en SGSPCSI:
 - |└┐ Riesgo: "Integración con AD compleja"
 - |└┐ Probabilidad: 40%, Impacto: +1 semana
 - |└┐ Mitigation: "Comenzar 1 semana antes"
 - |└┐ Status: Tracked semanalmente
-

PROCESO CMMI: PMC (Project Monitoring & Control)

SGSPCSI: Dashboard + Sprint Reviews + Status Reports

- |└┐ Rastrear vs Plan
 - |└┐ Dashboard automático en SGSPCSI:
 - |└┐ "Semana 2 de 4: 45% completado (plan: 50%)"
 - |└┐ Status: 🟡 On track (con warning)
 - |└┐ Acción: "Revisar si hay bloqueadores"
- |└┐ Identificar Varianzas
 - |└┐ Análisis automático:

- | | | Estimado: 40 hrs, Actual: 50 hrs (20% over)
 - | | | Causa: "Complejidad subestimada en módulo X"
 - | | | Acción: Agregar 1 persona, extender sprint
 - | | Tomar Acciones Correctivas
 - | | | Workflow en SGSPCSI:
 - | | | | Problema identificado
 - | | | | Acción propuesta y aprobada
 - | | | | Responsable asignado + plazo
 - | | | | Tracking hasta cierre
 - | | Gestionar Riesgos
 - | | | Risk monitoring en SGSPCSI:
 - | | | | Cada viernes: ¿Se activó el riesgo?
 - | | | | "Riesgo AD complexity: 50% probable ahora"
 - | | | | "Activar plan B: usar usuario/pwd local"
-

PROCESO CMMI: MA (Measurement & Analysis)
SGSPCSI: Dashboards de métricas + reportes automáticos

- | | Recopilar Datos
 - | | | Automático en SGSPCSI:
 - | | | | Fecha de solicitud → inicio → fin
 - | | | | Horas registradas en bitácora
 - | | | | Defectos reported desde usuarios
 - | | | | Encuestas CSAT al cerrar
 - | | Analizar Datos
 - | | | Reportes automáticos:
 - | | | | "Lead time: promedio 2.4 días (vs 10.5 hace 6 meses)"
 - | | | | "Defect rate: 3/mes (vs 15 hace 6 meses)"
 - | | | | "CSAT: 8.6/10 (vs 7.2 hace 6 meses)"
 - | | | | Gráficos de tendencia
 - | | Comunicar Resultados
 - | | | Dashboard ejecutivo en SGSPCSI:
 - | | | | KPIs principales visibles
 - | | | | Tendencias month-over-month
 - | | | | Recomendaciones basadas en datos
 - | | | | Notificaciones si KPI "rojo"
-

PROCESO CMMI: PPQA (Process & Product Quality Assurance)
SGSPCSI: Code review + testing + audits

- | | Evaluar Procesos
 - | | | Auditoría en SGSPCSI:
 - | | | | Cada mes: ¿Se sigue el workflow?
 - | | | | Checklist: 20 ítems de cumplimiento
 - | | | | Resultado: "15/20 → 75% cumplimiento"
 - | | | | Non-conformance: Plan de corrección
- | | Evaluar Productos
 - | | | Code review en SGSPCSI:
 - | | | | Checklist: Compila, >70% coverage, sin warnings
 - | | | | Otro dev aprueba (no el autor)
 - | | | | Si falla: Bloquea merge
 - | | | | Documentación de revisión en SGSPCSI
- | | Testing
 - | | | Estrategia en SGSPCSI:
 - | | | | Unit tests (desv escribe, mínimo 70% coverage)
 - | | | | Integration (team prueba todo junto)
 - | | | | UAT (usuario valida funcionalidad)
 - | | | | Regression (verifica que no rompimos nada)
- | | Gestión de Defectos
 - | | | Tracking en SGSPCSI:

- └ Defect report: Severidad, descripción, impacto
- └ Asignado a dev responsable
- └ RCA (root cause) documentada
- └ Verificación de fix por otro dev

PROCESO CMMI: CM (Configuration Management)
SGSPCSI: Integración con Git + release management

- └ Identificar Items
 - └ En SGSPCSI y Git:
 - └ Código fuente (Git)
 - └ Scripts SQL (Git)
 - └ Documentación (versionada)
 - └ Configuraciones (como código, en Git)
- └ Baselines
 - └ Tags en Git, documentados en SGSPCSI:
 - └ v1.0.0: Primera release en producción
 - └ v1.1.0: Features nuevas + bugfixes
 - └ Cada baseline tiene "release notes"
- └ Change Control
 - └ Integración Git ↔ SGSPCSI:
 - └ Pull request vinculado a ticket SGSPCSI
 - └ "PR #123 implements REQ-045"
 - └ Code review antes de merge
 - └ Merge solo por PM/Tech Lead
- └ Auditoría CM
 - └ Reporte en SGSPCSI:
 - └ "¿Qué está en producción?" v1.2.3 ✓
 - └ "¿Tenemos rollback?" v1.2.2 ready ✓
 - └ "¿Está documentado?" Release notes ✓
 - └ "¿Completitud?" 100% items versionados ✓

PARTE V: MODELO FUTURO - VISIÓN DE EXCELENCIA

5. Beneficios Consolidados: Año 1 vs Situación Futura

5.1 Comparativa: ANTES → DESPUÉS (Después de 6-12 meses)

IMPACTO CUANTIFICABLE TOTAL

ANTES (ACTUAL) → DESPUÉS (MES 12)

MÉTRICA 1: LEAD TIME (Solicitud → Inicio desarrollo)

Actual: 10.5 días
Objetivo: 2.5 días
Mejora: 75% reducción

- └ Impacto: Usuarios reciben respuesta 8 días antes
- └ Impacto: Equipo puede atender 40-50% más tickets
- └ Efecto: ~170 días de delay acumulado ELIMINADO/año (~4.25 semanas productivas ganadas)

MÉTRICA 2: SOLICITUDES PERDIDAS

Actual: 18% (5-8 por mes)

Objetivo: 0%
Mejora: 100% eliminación

- └ Impacto: Cero frustración por "¿Dónde está mi solicitud?"
- └ Impacto: Cero trabajo duplicado
- └ Efecto: ~120 horas/año de búsquedas ELIMINADAS + reputación restaurada

MÉTRICA 3: RETRABAJO / TICKETS REABIERTOS

Actual: 25% reabiertos
Objetivo: 10% (o menos)
Mejora: 60% reducción

- └ Impacto: 128 horas/mes ahorradas en equipo de 8 (~1,536 horas/año)
- └ Impacto: Mejor calidad primera vez
- └ Productividad: Equipo disponible para más proyectos

MÉTRICA 4: ON-TIME DELIVERY

Actual: 60% de tickets a tiempo
Objetivo: 90%
Mejora: 50% mejora

- └ Impacto: Usuarios creen en compromisos de IT
- └ Impacto: Mejor planeación de negocio
- └ Productividad: Reducción de tiempo en crisis management y escalaciones

MÉTRICA 5: SATISFACCIÓN DE USUARIO (CSAT)

Actual: 7.2/10
Objetivo: 8.5/10
Mejora: +18%

- └ Impacto: NPS (Net Promoter Score) +20 puntos
- └ Impacto: Usuarios confían en IT nuevamente
- └ Valor: Menos escalaciones, mejor relación

MÉTRICA 6: DEFECTOS EN PRODUCCIÓN

Actual: 15 defectos/mes (promedio)
Objetivo: 3-4 defectos/mes
Mejora: 75% reducción

- └ Impacto: Menos downtime de usuarios
- └ Impacto: Menos urgencias nocturnas (PM no interrumpido)
- └ Productividad: ~60 horas/mes NO interrumpidas por incidentes

MÉTRICA 7: PRODUCTIVIDAD EFECTIVA DEL EQUIPO

Actual: 60% (tiempo perdido: 40%)
Objetivo: 80% (tiempo perdido: 20%)
Mejora: +33% en productividad

- └ Impacto: 288 horas/mes de trabajo productivo adicional (~3,456 horas/año)
- └ Impacto: Equivalente a contratar 2.16 personas (sin costo)
- └ Productividad: 22.5% mejora en tiempo productivo del equipo

MÉTRICA 8: DISPONIBILIDAD DEL SISTEMA (SGSPCSI)

Actual: N/A (nuevo sistema)
Objetivo: 99.5% uptime
Mejora: Usuarios confían que funciona

- └ Impacto: Máx 10 horas/mes downtime (vs 30-40 horas actual)
- └ Impacto: 200 usuarios = menos interrupciones y frustraciones
- └ Productividad: 99.5% uptime = confiabilidad ganada

RESUMEN DE IMPACTO EN PRODUCTIVIDAD:

Reducción de tiempos de retraso:	170 días/año
Reducción de retrabajo:	1,536 hrs/año
Aumento de productividad:	3,456 hrs/año
Reducción de interrupciones:	288 hrs/año
Mejora de uptime:	99.5% (vs N/A)
Búsquedas de solicitudes eliminadas:	120 hrs/año
TOTAL HORAS PRODUCTIVAS GANADAS:	5,570 hrs/año
EQUIVALENTE:	+2.76 FTE
(Sin costo de salario adicional)	
(Mejoras se consolidan y crecen en años posteriores)	

INVERSIÓN PARA LOGRAR ESTO:

- └ Residencias (Jacqueline + Edwin): Recursos ya comprometidos (26 semanas)
- └ Herramientas: Licencias SaaS estándar (bajo costo)
- └ Capacitación: Dirigida por Karla (DTI-CSI)
- └ Tiempo de soporte interno: Absorbido por equipo existente
- └ COSTO TOTAL: Dentro del presupuesto existente (TBD formal)

IMPACTO EN NEGOCIO:

- └ Ganancia: 5,570 horas/año de productividad adicional (equivalente a +2.76 personas)
- └ Plazo de implementación: 26 semanas (enero-julio 2026)
- └ Sostenibilidad: Mejoras se mantienen y amplifican con CMMI (vea PARTE IV)

5.2 Impacto Estratégico a Largo Plazo

AÑO 2-3: Consolidación y Optimización Continua

Si ISSEG continúa mejorando (más allá del período de residencias):

ESCENARIO A: Continúan como están (sin CMMI)	
└ Madurez: Sigue en 47.5% Nivel 2	
└ Lead time: Vuelve a 8-10 días (sin disciplina)	
└ Beneficios: Se pierden gradualmente (falta disciplina)	
└ Capacidad: No escalan (siguen manualmente)	
└ COSTO en Productividad:	
└ ~200,000+ horas/3años de trabajo no productivo	
└ Equivalente: Mantener 100+ personas en tareas no valor	
└ RESULTADO: Ciclo eterno de "crisis management"	
Se pierde capacidad de crecimiento y modernización	
ESCENARIO B: Continúan inversión en CMMI	
(Llegar a 85%+ Nivel 2, Nivel 3)	

- └ Año 1: 70-75% Nivel 2 (gracias a residencias)
- └ Año 2: 85%+ Nivel 2, inicio Nivel 3 (10-15%)
- └ Año 3: 50%+ Nivel 3 (definido y estandarizado)
- └ Beneficios ACUMULADOS:
 - └ Lead time se estabiliza en 2 días
 - └ On-time delivery: 95%+ consistente
 - └ Defects: < 1 por mes (calidad extremadamente alta)
 - └ Capacidad: Fácilmente escalar a 12-15 personas
 - └ Conocimiento documentado (no perdido si rota)
 - └ Procesos predecibles y repetibles
- └ GANANCIA en Productividad (3 años):
 - └ Año 1: 5,570 horas ganadas
 - └ Año 2: 7,200 horas ganadas (consolidación)
 - └ Año 3: 8,640 horas ganadas (optimización)
 - └ TOTAL: ~21,410 horas productivas adicionales
 - Equivalente a +10.75 FTE durante 3 años
- └ RESULTADO: ISSEG se convierte en referencia de excelencia dentro de ISSEG.
 - Equipo motivado, usuarios satisfechos,
 - Dirección confía en planes con datos respaldados

DIFERENCIA CRÍTICA (3 AÑOS):

- └ Escenario A (sin CMMI): Pierde ~200,000 hrs productivas
- └ Escenario B (con CMMI): Gana ~21,410 hrs productivas
- └ DIFERENCIA: 221,410 horas = capacidad para escalar a 25-30 personas sin caos o burnout del equipo

5.3 Visión Final: ISSEG Como Centro de Excelencia

VISIÓN (18 MESES POSTERIORES):

SITUACIÓN DE ISSEG 2027:

1. OPERACIÓN SGSPCSI:

- └ 100% de solicitudes digitales (error rate: 0%)
- └ Lead time: 2-3 días (predecible)
- └ On-time delivery: 95%+ (plazo confiable)
- └ CSAT: 8.8/10 (usuarios muy satisfechos)
- └ Uptime: 99.5% (usuarios confían)

2. MADUREZ CMMI:

- └ Nivel 2: 85%+ alcanzado
- └ Procesos: Completamente documentados
- └ Equipo: Entiende y sigue procesos sin fricción
- └ Disciplina: Parte de la cultura (automatizada)
- └ Auditoría: Podría certificación CMMI si quisiera

3. CAPACIDAD:

- └ Equipo: Escaló de 8 a 12-15 personas (sin caos)
- └ Nuevos dev: Onboarding en 2-3 semanas (vs 2-3 meses)
- └ Conocimiento: Documentado, no depende solo de "héroes"
- └ Velocidad: 35-40 tickets/mes (vs 20 actual)
- └ Riesgo: Bajo (procesos absorben la incertidumbre)

4. VALOR PARA ISSEG:

- └ Proyectos se entregan a tiempo, dentro de presupuesto
- └ Usuarios confían en IT (relación restaurada)
- └ Dirección aprueba presupuesto IT sin cuestionamiento
- └ Riesgos estratégicos mitigados (documentación, knowledge)
- └ ISSEG es visto como "moderna y eficiente"

5. IMPACTO EN CAPACIDAD OPERACIONAL:

- └ Horas productivas ganadas: ~8,640 hrs/año (año 3)
- └ Equivalente a: +4.3 FTE adicionales sin costo salarial
- └ Costo de mantenimiento: Mínimo (procesos documentados)
- └ Payback de mejoras: Sostenido y creciente año a año

CONCLUSIÓN:

El proyecto de residencias no es solo "crear un sistema".
Es el CATALIZADOR para transformar ISSEG de:

CAÓTICO, REACTIVO, IMPREDECIBLE

a

DISCIPLINADO, PROACTIVO, PREDECIBLE

SGSPCSI = Herramienta (el vehículo)

CMMI = Camino (la disciplina)

Juntos: EXCELENCIA OPERACIONAL

CONCLUSIÓN EJECUTIVA

PROYECTO RESIDENCIAS: SGSPCSI + CMMI

Una oportunidad única para transformar un equipo
de "actitud heroica" a "actuación sistemática"

PROBLEMA:

- ✗ 15-20% de solicitudes se pierden
- ✗ Lead time 10.5 días (5x más lento que ideal)
- ✗ 25% de tickets reabiertos (retrabajo)
- ✗ 0% visibilidad para usuarios/dirección
- ✗ Decisiones sin datos (al ojo)
- ✗ Equipo frustrado, rotación 15-20%

SOLUCIÓN:

- ✓ SGSPCSI: Sistema digital centralizado
- ✓ Captura 100% de solicitudes
- ✓ Flujos estructurados y automatizados
- ✓ Trazabilidad y transparencia total
- ✓ Datos para decisiones

MEJORA:

- ✓ CMMI Nivel 2: Disciplina y sistema
- ✓ Procesos documentados y auditables
- ✓ Métricas para mejora continua
- ✓ Escalabilidad sin caos
- ✓ Preparado para certificación (si queremos)

BENEFICIOS AÑO 1:

- 🕒 Lead time: 75% más rápido (11.5 días → 2.5 días)
- 🕒 Productividad ganada: 5,570 horas/año
- 📄 On-time delivery: 90%+ (vs 60% actual)
- 😊 Satisfacción usuario: 8.5/10 (vs 7.2 actual)
- 👤 Reputación IT: Restaurada (+40 puntos de confianza)
- 🔍 Visibilidad: 100% de solicitudes rastreables

INVERSIÓN:

- 📁 Residencias: Recursos ya comprometidos (26 semanas)
- 📁 Herramientas: Licencias estándar (presupuesto operativo)
- 📁 Capacitación: Dirigida por equipo interno (Karla)
- 📁 Costo adicional: Mínimo (dentro presupuesto existente)

RETORNO (Medido en Productividad):

- 📅 Año 1: +5,570 horas productivas (≈+2.76 FTE)
- 📅 Año 2: +7,200 horas productivas (consolidación)
- 📅 Año 3: +8,640 horas productivas (optimización)
- 📅 Total 3 años: +21,410 horas = +10.75 FTE equivalentes

SUSTAINABILITY:

- ✓ Mejoras se institucionalizan (procesos documentados)
- ✓ No dependen de "héroes" individuales
- ✓ Nuevos empleados onboarding en 2-3 semanas
- ✓ Conocimiento capturado y transferible

TIMELINE:

- 📅 26 semanas (enero 26 - julio 20, 2026)
- 📅 5 fases bien definidas
- 📅 Hitos claros y medibles
- 📅 Residentes: Jacqueline + Edwin
- 📅 Asesor: Karla Arroyo

VIABILIDAD:

- ✓ Stack conocido (Angular, .NET, SQL)
- ✓ Requisitos claros
- ✓ Equipo capacitado
- ✓ Riesgos identificados y mitigados
- ✓ Plan detallado y realista

RECOMENDACIÓN:

APROBACIÓN INMEDIATA

Este proyecto es transformacional para ISSEG.
No es solo "un sistema", es catalizador de excelencia.

2026: Implementación (Residencias)
2027: Consolidación y optimización
2028+: Centro de excelencia operacional

ISSEG: De "crisis management" a "excelencia operacional"

Referencia de estimaciones monetarias

- Estas cifras son referenciales y se calculan a partir de tiempos perdidos convertidos a horas.
- Formula general: $\$ \text{horas perdidas/año} \times \text{tasa promedio por hora} = \text{estimado anual}$.
- Supuestos usados:
 - Tasa promedio: \$31 USD/hora (referencia de mercado para trabajo IT).
 - Tipo de cambio: $1 \text{ USD} = 17.0 \text{ MXN}$ (referencia Feb-2026).
- Ejemplo (retrasos): $170 \text{ días} \times 8 \text{ horas/día} = 1,360 \text{ horas/año}$; $1,360 \times 31 = 42,160 \text{ USD}$ (~714,000 MXN).

Documento Integral Completado

Investigación: Problema → Solución → Mejora → Implementación

Instituto de Seguridad Social del Estado de Guanajuato (ISSEG)

Febrero 20, 2026