MARK III

· AI Process Architecture

BLUEPRINT · V2.0

Pipeline Fase-Gate para Desarrollo de Software Asistido por IA

Motor: Claude + herramientas de IA.

Martha Gabriela Carmona Campos
AI Process Architect

Marzo 2026

1. Visión General

MARK III v3.0 es un pipeline fase-gate para desarrollo de software asistido por IA. Cada fase tiene entradas, entregables y un criterio de salida (gate) que debe cumplirse para avanzar.

Filosofía

Garbage in, garbage out.

La IA no reemplaza experiencia ni pensamiento sistémico — amplifica las bases sólidas y expone las deficiencias. MARK1 pone IA donde más impacto tiene: en las fases más caras e ignoradas del SDLC.

Modelo Operativo

Un operador con conocimiento de ingeniería de software, DevOps e IA. Sin equipos grandes, sin ceremonias de Scrum. Proyectos completos de principio a fin con entregables definidos.

1.1 Pipeline Completo

Fase	Nombre	Función	Gate de Salida
F1	Entrevistador	Captura requerimientos adaptativos	¿Claude Code podría codear sin adivinar?
F2	Prototipado	Valida UI con stakeholder	¿Cada pantalla mapea a HUs?
F3	Arquitecto	Define stack, estructura, modelo, contratos	¿Todo decidido, nada ambiguo?
F4	Preparación	Scaffolding con Claude dentro del carril	¿docker-compose up funciona?
F5	Codeo	Claude Code implementa por waves	¿Compila y responde por wave?
F6	Validación	Tests E2E + SAST + auditoría	¿100% CAs, 0 críticos?
F7	Despliegue	Pipeline CI/CD + release	¿Corriendo en destino?

1.2 Catálogo de Stacks

TIER 1 — VALIDADO EN PRODUCCIÓN

Spring Boot + Angular + PostgreSQL — Probado con SIAD (14 días, 16/16 HUs, 0 fallas E2E, 0 vulnerabilidades críticas SAST).

TIER 2 — ALTA CONFIANZA, POR VALIDAR

Python/FastAPI + React + PostgreSQL — Para proyectos web modernos, MVPs, startups.

Electron + React/Angular — Para aplicaciones de escritorio.

TIER 3 — EXPERIMENTAL

Todo lo demás. Requiere validación con un proyecto real antes de incorporarse al catálogo. El output requiere revisión manual adicional.

Base de datos estándar: PostgreSQL siempre como default (PRIMERA VERSION DE MARKIII).

F1 — Entrevistador — Levantamiento de Requerimientos

Capturar el contexto completo del negocio para que los agentes downstream produzcan software **sin tener que adivinar nada.**

2.1 Modelo Adaptativo por Complejidad

Complejidad	Entregables	Ejemplo
Baja (CRUD)	HUs + CAs + modelo de datos básico	Sistema de registro deportivo
Media (reglas de negocio)	+ Reglas de dominio + diagrama de estados	Gestión de becas con topes presupuestales
Alta (integraciones)	+ Contratos de API + BPMN de procesos críticos	Sistema que consume servicios SOAP/REST

2.2 Entregables de F1

1. **Historias de Usuario (HUs) — Formato estándar 'Como [rol] quiero [acción] para [beneficio]'. Agrupadas por épicas.**
2. **Criterios de Aceptación (CAs) — Formato Dado/Cuando/Entonces. Mapeo 1:1 con tests E2E.**
3. **Modelo de datos conceptual — Entidades, atributos principales, relaciones.**
4. **Reglas de dominio — Explícitas, no implícitas. Complejidad media o superior.**
5. **Contratos de integración — APIs externas, sistemas legacy. Complejidad alta.**
6. **BPMN de procesos críticos — Diagramas de flujo con múltiples actores o estados.**

GATE DE SALIDA

¿Claude Code podría producir este sistema sin tener que adivinar nada? Si la respuesta es no, falta información y el paquete no avanza.

F2 — Prototipado

Validar la interfaz de usuario con el stakeholder antes de codear — reducción de retrabajo — y generar input visual para los agentes downstream.

3.1 Doble Función del Prototipo

Validación con el cliente: '¿Esto es lo que quieres?' Algo visual que un no-técnico entienda.

Input técnico para IA: La estructura de pantallas define componentes, el flujo define rutas, los campos definen el modelo de datos.

3.2 Entregables de F2

1. Prototipo clickeable navegable — todas las pantallas del sistema con navegación funcional.

2. Mapa de pantallas ↔ HUs — cada pantalla referencia las HUs que implementa.

GATE DE SALIDA

¿Cada pantalla mapea a una o más HUs, y cada campo visible corresponde a un atributo del modelo de datos de F1?

F3 — Arquitecto

Producir un paquete técnico completo que Claude Code pueda consumir sin ambigüedad. Todas las decisiones arquitectónicas quedan resueltas aquí.

4.1 Entregables del Arquitecto

Entregable	Descripción	Consumidor
Stack definido	Selección del catálogo (Tier 1/2). Sin opciones, decidido.	Claude Code + operadora
Estructura de proyecto	Árbol de carpetas completo según template del stack.	Claude Code
Modelo de datos técnico	Tablas, columnas, tipos, relaciones, constraints, índices.	Claude Code
Contratos de API	Endpoints, métodos, request/response, códigos de error.	Claude Code + testing
ADRs ligeros	Autenticación, patrón arquitectónico, manejo de errores, naming.	Claude Code + operadora
CLAUDE.md	Archivo auto-generado con contexto completo del proyecto.	Claude Code
Infra base	Dockerfiles, docker-compose, nginx.conf. Templated por stack.	Pipeline de deploy

4.2 Orden de Implementación (Waves)

Wave	Contenido	Justificación
Wave 1	Autenticación + entidades base	Siempre primero: login, roles, entidades raíz
Wave 2	CRUDs principales	Operaciones básicas sobre entidades core
Wave 3	Reglas de negocio y flujos	Lógica que depende de entidades existentes
Wave 4	Reportes, dashboards, features secundarios	Funcionalidad que consulta/agrega datos

GATE DE SALIDA

¿Puedo abrir Claude Code, darle este paquete + una HU, y que produzca código que respete la arquitectura sin que yo tenga que intervenir en decisiones estructurales?

F4 — Preparación para Codeo

Materializar las decisiones del Arquitecto en un repositorio real listo para que Claude Code comience a implementar.

5.1 Modelo: Claude en Carril

Claude opera con libertad para adaptar el template al proyecto específico, pero dentro de un marco inviolable. Como un tren, no como un coche.

LO QUE CLAUDE NO PUEDE HACER

✗ Cambiar la estructura de carpetas de primer nivel.

✗ Agregar dependencias fuera de la lista aprobada del stack.

✗ Modificar configuración de Docker, Nginx o infraestructura.

✗ Tomar decisiones arquitectónicas no definidas en F3 (debe preguntar).

✗ Violar naming conventions (PascalCase clases, kebab-case endpoints, snake_case tablas).

LO QUE CLAUDE SÍ PUEDE HACER

✓ Generar entidades JPA / modelos SQLAlchemy a partir del modelo de datos de F3.

✓ Crear migraciones Flyway/Alembic.

✓ Generar puertos (interfaces) y adaptadores vacíos por entidad.

✓ Crear DTOs a partir de los contratos de API.

✓ Generar feature modules de frontend a partir de las épicas.

✓ Adaptar el CLAUDE.md con rutas y configuraciones reales.

> GATE DE SALIDA
>
> ¿docker-compose up levanta sin errores, la app arranca vacía, y Claude Code puede empezar a trabajar HU por HU?

F5 — Codeo con Claude Code

Implementar el software completo usando Claude Code como motor de desarrollo, wave por wave.

6.1 Flujo de Implementación

1. **Claude Code abre el proyecto scaffoldeado por F4.**
2. **Lee el CLAUDE.md (contexto completo del proyecto).**
3. **Toma la primera HU de la wave actual.**
4. **Implementa backend + frontend para esa HU.**
5. **Siguiente HU de la wave. Repite hasta completar la wave.**
6. **Checkpoint de refactor entre waves.**
7. **Siguiente wave. Repite hasta completar.**

6.2 Controles de Calidad en Codeo

Checkpoint de Refactor (entre waves): Claude Code revisa el código acumulado y refactoriza servicios con más de 200 líneas, lógica duplicada, métodos que violan single responsibility, imports sin usar.

Memoria Persistente entre Sesiones: El CLAUDE.md contiene una sección 'Decisiones Tomadas' que se actualiza al final de cada sesión.

GATE DE SALIDA

¿Cada wave compila, la app levanta, y los endpoints/pantallas de esa wave responden

F6 — Validación y Testing

Verificar que el software funciona correctamente (F6a) y que el código tiene calidad profesional (F6b).

7.1 F6a — Tests Funcionales

Mapeo 1:1 entre criterios de aceptación y tests E2E con Playwright. Cada test se genera interactivamente con Claude Code + Playwright MCP y se guarda como artefacto permanente en el repositorio.

Principio: El test se ejecuta una vez manualmente con IA y queda automatizado para siempre.

7.2 F6b — Revisión Técnica / Calidad

Revisión	Herramienta	Criterio
Seguridad (SAST)	Semgrep	0 vulnerabilidades críticas, 0 altas sin justificación
Calidad de código	Claude (auditoría)	Sin clases >300 líneas, sin lógica duplicada, sin secrets
Cobertura de CAs	Matriz HU→CA→test	100% de CAs tienen test E2E asociado

GATE DE SALIDA

¿100% de CAs tienen test? ¿0 vulnerabilidades críticas? ¿Todos los tests pasan? ¿La auditoría no tiene hallazgos bloqueantes?

F7 — Despliegue y Liberación

Llevar el software validado a producción con un pipeline CI/CD automatizado.

8.1 Pipeline CI/CD por Stack

El pipeline no se diseña por proyecto, se instancia del template del stack. Stages fijos:

`build → test → sast → e2e → build-image → push-registry → deploy`

Stack	Build	Registry	Deploy
Spring+Angular+PG	Maven + ng build	Harbor / Docker Hub	K8s / Docker Compose
FastAPI+React+PG	pip + npm build	Harbor / Docker Hub	K8s / Docker Compose
Electron	npm + electron-builder	N/A (binario)	Distribución directa

8.2 Estrategias de Despliegue

Opción	Descripción	Cuándo usar
Infra del cliente	Entrega código + Dockerfiles + manifiestos. El cliente opera.	Arranque. Menor riesgo.
Cloud manejado	Railway, Render, Fly.io, o AWS/GCP del cliente. Tú configuras.	Proyectos medianos.
Tu infraestructura	Tú despliegas, monitoreas, cobras hosting.	Cuando haya volumen.

GATE DE SALIDA

¿La app está corriendo en el ambiente destino, el pipeline pasa completo sin intervención manual, y el cliente puede acceder a la aplicación funcionando?

Métricas de Observabilidad

Cada proyecto producido por MARK1 registra las siguientes métricas para medir eficiencia y mejorar el pipeline:

Métrica	Qué mide	Baseline (SIAD)
Tiempo total del pipeline	Día 0 (entrevista) a día N (deploy)	14 días
HUs entregadas	Cantidad de funcionalidades completadas	16/16
CAs validados	Criterios de aceptación con test E2E	133 (0 fallas)
Vulnerabilidades SAST	Hallazgos críticos + altos	0 críticos, 15 infra
Tiempo por fase	Duración de cada F1–F7	Por documentar
Retrabajo	HUs que requirieron corrección post-F6	Por documentar

Con 3-4 proyectos completados, estas métricas permiten demostrar que MARK1 es repetible, medible y mejora con cada iteración. Eso transforma una historia convincente en un caso de negocio.

Roadmap

CORTO PLAZO — PRÓXIMAS 4 SEMANAS

→ **Afinar prompts de agentes F1, F2, F3 con base en este blueprint.**

→ **Construir templates de F4 para Tier 1.**

→ **Ejecutar un proyecto demo completo F1–F7, grabado.**

→ **Publicar artículo técnico + video demo.**

MEDIANO PLAZO — 2-3 MESES

→ **Validar Tier 2 (FastAPI+React) con un proyecto real.**

→ **Completar 3 proyectos con métricas comparables.**

→ **Sacar CKA.**

→ **Posicionamiento activo en LinkedIn/YouTube.**

LARGO PLAZO — 6+ MESES

→ **MARK1 como servicio de consultoría vendible.**

→ **Modelo de delegación: otras personas operan F1–F3.**

→ **Gemelo Digital Operativo (SRE agent con contexto institucional completo).**