Plan de Gestión del Proyecto de Software (SPMP)

Producto: Plataforma de Servicios Locales (marketplace cliente–contratista)
Versión: 1.1 (ajustado en base al nuevo cronograma)
Fecha: 11/09/2025
Estado: Borrador para aprobación
Project Manager: Fernando Ramos
Equipo núcleo: Backend Dev (Mateo García), Frontend Dev (Leonardo Daniel HG.), QA (Luis González)
Horizonte: 7 sprints de 1 semana
Ritmo de trabajo: Sync de devs Lun-Mié-Vie
Herramienta de gestión: Azure DevOps (backlog, RFC/CCB, pipelines, decision log)

0. Control de cambios

Versión	Fecha	Autor	Descripción
1.1	10/09/2025	Equipo	Ajustes al documento al nuevo cronograma, basado en las semanas restantes del semestre.
1.0	28/08/2025	Equipo	Primera versión del SPMP.

1. Propósito, alcance y entregables

Propósito: Construir un MVP web que implemente todos los requerimientos funcionales definidos, cumpla criterios de aceptación y RNF de desempeño, seguridad y privacidad, y sea escalable para soportar picos de demanda con balanceo/capacidad de crecimiento.

Alcance del MVP.

- **Canal web responsive** (ZMG como mercado inicial)**.**
- **Flujos principales:** búsqueda y exploración, registro/login, publicación de servicios y disponibilidad, reserva y checkout con anticipo (Stripe Checkout + Connect Express), estados y notificaciones via webhooks, mensajería entre usuarios, calificaciones, liquidación al contratista y panel admin básico.
- **RNF clave:** latencias P95 por flujo según SRS; seguridad (OWASP básico, sin almacenar PAN/CVV), privacidad (ARCO, retención acotada), observabilidad y registro de auditoría.

Fuera de alcance: Apps móviles nativas, internacionalización amplia y cualquier elemento adicional ya excluido en los artefactos (SRS/SDD/presentación).

Criterios de aceptación del proyecto (Go/No-Go).

1. Todos los RF del MVP operativos en PROD.
2. Cumplimiento de CA/RNF mínimos (latencias P95 por flujo, error-rate bajo umbral, accesibilidad básica en flujos críticos, OWASP básico).
3. Pasarelas de pago integradas sin almacenamiento de PAN/CVV y con idempotencia de webhooks.
4. Dataset de prueba “cientos” (≥300 servicios, ≥200 usuarios, ≥200 reservas) y evidencia de pruebas E2E.
5. Firma de aceptación por Sponsor (y, cuando aplique, validación de cliente evaluador).

2. Organización del proyecto y gobierno

Estructura y roles.

- **Sponsor / Aprobador final: Profesor** (sponsor del curso).
- **Project Manager: Fernando Ramos** (planificación, scope, CCB/RFC, reporting, riesgos, releases).
- **Backend Dev: Mateo García** (servicios, DB, integraciones, webhooks, performance back).
- **Frontend Dev: Leonardo Daniel HG.** (UI/UX, flujos FE, accesibilidad, performance front).
- **QA: Luis González** (estrategia y ejecución de pruebas, UAT, métricas de calidad).

Responsabilidad y colaboración: La responsabilidad es compartida por rol; la toma de decisiones formales y aprobaciones pasan por Sponsor y PM.

Matriz RACI:

- **R** (Responsable)
- **A** (Aprueba)
- **C** (Consulta)
- **I** (Informa)

Actividad	R	A	C	I
Definir alcance MVP	PM	Sponsor	BE/FE/QA	Cliente
Priorización del backlog	PM	Sponsor	Equipo	Cliente
Arquitectura y DB	BE	Sponsor	PM/FE	QA
Integración Stripe	BE	Sponsor	PM/QA	FE
Front/UI/Accesibilidad	FE	Sponsor	PM/QA	BE
Pruebas y UAT	QA	Sponsor	PM/BE/FE	Cliente

Releases/Go-live	PM	Sponsor	BE/FE/QA	Cliente
Cambios (RFC/CCB)	PM	Sponsor	Cliente	Equipo

3. Proceso de gestión y cadencia

Ciclo de vida: Ágil ligero con sprints de 1 semana (7 en total) y 1 épica por sprint.

Ceremonias:

- **Sync de devs:** Lunes, Miércoles y Viernes (30–45 min, foco en bloqueadores y handoffs).
- **Planning del sprint:** Lunes (definición de objetivo, selección de historias, estimación T-shirt, capacidad semanal).
- **Review con Sponsor:** Viernes (demo, decisiones, aprobación/rechazo, próximos cambios).

Backlog, requisitos y trazabilidad: Azure DevOps como fuente operativa; SRS como single source of **truth de requisitos.**

Trazabilidad: RF ↔ Épica ↔ Historias ↔ Tasks.

Estimación: T-shirt sizes (XS/S/M/L/XL) y conversión a capacidad semanal por rol.

Gestión de cambios: PM controla y gestiona cambios (RFC en ADO) y los valida con Sponsor/cliente.

4. Plan de trabajo: épicas, WBS y cronograma

Estrategia de implementación. Orden secuencial para reducir riesgo de terceros y maximizar valor temprano. Las épicas se despliegan como sigue:

- **S1: Autenticación & Perfiles Base.**
    - **WBS:** setup CI/CD, diseño base y navegación, registro/login, perfiles mínimos, políticas de contraseñas y sesiones, pruebas unitarias iniciales.
    - **Demo:** login/registro y UI base operativos.

- **S2: Catálogo & Búsqueda.**
    - **WBS:** modelos y seed inicial, listados, filtros, ordenamiento, índices de DB, caché ligera, pruebas de latencia de búsqueda.
    - Demo: búsqueda con P95 dentro de umbral.

- **S3 — Publicación & Disponibilidad del Contratista.**
    - **WBS:** alta/edición de servicios, agenda y slots, validaciones, pruebas de integración front-back.
    - **Demo:** publicar servicio y configurar disponibilidad.

- **S4: Reserva & Checkout (Anticipo) + Webhooks.**
    - **WBS:** intento de reserva, sesión de Checkout (Stripe), webhooks con idempotencia, estados post-pago, observabilidad específica de pagos, manejo de errores y reintentos.
    - **Demo:** reserva confirmada tras pago de anticipo, tiempos P95 en umbral.

- **S5: Estados de Reserva + Mensajería.**
    - **WBS:** máquina de estados, transición segura, mensajería entre usuarios, moderación básica.
    - **Demo:** flujo de estados y chat funcional.

- **S6: Calificaciones + Liquidación.**
    - **WBS:** cierre de reserva, ratings y comentarios, cálculo/liquidación al contratista, conciliación básica.
      Demo: liquidación y rating completados.

- **S7: Admin Básico + Hardening/UAT.**
    - **WBS:** vistas administrativas esenciales, auditoría, accesibilidad en flujos clave, endurecimiento de seguridad, UAT y correcciones.
    - **Demo final + acta de aceptación.**

5. Recursos, herramientas y presupuesto

Stack técnico.

- **Frontend:** React + Tailwind.
- **Backend:** Node.js.
- **Base de datos:** PostgreSQL + Prisma.
- **Pagos:** Stripe Checkout + Connect (Express).
- **Nube/Servicios:** AWS (S3 para medios, SES/SMTP para correo, Amazon Location para geocodificación/rutas), CDN para estáticos.
- **CI/CD:** GitHub Actions o Azure Pipelines (según repos).

Entornos.

- **dev** (ramas feature/*; manejo por devs) y **main** como rama protegida (staging→prod ligeros). Checks sugeridos: build, lint, tests.

6. Calidad y pruebas

Estrategia de pruebas.

- **Unitarias (BE/FE)** para lógica y utilidades.
- **Integración (API, DB, Stripe webhooks, Location).**
- **End-to-End (E2E) en flujos críticos:** Búsqueda → detalle → checkout → webhook → estado → liquidación.

7. Seguridad, privacidad y cumplimiento

- **Auth y control de acceso:** JWT.
- **Pagos: sin almacenar PAN/CVV;** idempotencia en webhooks; registro de eventos.
- **Seguridad app:** Top 10 OWASP.
- **Privacidad:** derechos ARCO; retención acotada (p.ej., mensajes 7 dí

8. Riesgos y respuesta

Matriz de riesgo.

ID	Riesgo	Prob.	Impacto	Dueño	Mitigación	Disparadores	Contingencia
R1	Dependencia Stripe/CDN/Mapas	M	A	BE	Circuit-breakers, idempotencia	timeouts elevados, 5xx, latencia P95 ↑	degradar pagos, cola offline
R2	Capacidad equipo (agenda)	M	M	PM	buffers, foco en bloqueadores	ausencias, entregas académicas	recortar alcance táctico
R3	Créditos AWS/tokens	M	M	BE/FE	caching/compresión /limits	costos ↑, throttling	desactivar features no críticas
R4	Moderación/disputas	B	M	PM/QA	reglas/SLAs, admin básico	tasa de reportes ↑	reforzar admin, plantillas

9. Comunicaciones

- **Syncs devs:** Lun-Mié-Vie (15-30 min).
- **Review con Sponsor:** Viernes (demo, decisiones, registro en decision log).
- **Canales:** WhatsApp para constante comunicación, Azure DevOps (work items, RFC, decision log), repos Git para issues técnicos.
- **Reportes semanales:** Estado por épica, quemado de capacidad, riesgos/top bloqueadores, métricas P95 y error-rate, decisiones tomadas/pending y próximos pasos.