Product Requirements Document (PRD): Project Showcase App

Autor: Esteban de Santiago García

Estado: Planificado (MVP)

Versión: 1.1

Última actualización: 2025-10-20

---

1. Propósito

Crear una aplicación pública para exhibir agentes de IA y los equipos que los construyeron, mostrando propósito, capacidades y autoría de manera clara y navegable.

2. Alcance y Contexto

- Frontend: Flutter.
- Backend: Java + Spring Boot (API REST, solo lectura en MVP).
- Datos: Supabase (PostgreSQL/BaaS).
- Fuera de alcance (MVP): métricas individuales, grabaciones, explorador de carpetas, autenticación.

3. Visión (alineada al Agente Product Owner)

Maximizar valor y transparencia: priorizar visualización clara de agentes y equipos, con historias testables y backlog visible.

4. Objetivos

- Centralizar información de agentes y equipos.
- Aumentar visibilidad del trabajo.

- Facilitar comprensión de capacidades y alcance.

5. Usuarios y Casos de Uso

Perfiles: visitante público, stakeholder/colaborador. Casos (MVP):

- CU1: Ver lista/galería de agentes.
- CU2: Ver detalle de agente.
- CU3: Ver lista de equipos.
- CU4: Ver detalle de equipo y miembros.
- CU5: Acceder a ficha básica del desarrollador principal desde el agente.

6. Historias y Criterios de Aceptación

Épica Agentes:

- HU-1.1: Ver lista de agentes.
  - CA: Muestra nombre y equipo; maneja vacío/error.
- HU-1.2: Ver detalle de agente.
  - CA: Muestra descripción, rol, capacidades, equipo y desarrollador principal.
- HU-1.3: Abrir especificación si existe.
  - CA: Si hay specUrl, botón que abre enlace.

Épica Equipos/Desarrolladores:

- HU-2.1: Ver lista de equipos.
  - CA: Muestra nombre y conteo de agentes.
- HU-2.2: Ver detalle de equipo.
  - CA: Lista miembros (nombre, rol) y agentes del equipo.
- HU-2.3: Ver ficha de desarrollador principal.
  - CA: Muestra nombre y rol.

7. Requisitos Funcionales

- RF-1: Listado de agentes con paginación simple o lazy load.

- RF-2: Detalle de agente con chips de capacidades.
- RF-3: Listado de equipos con conteo de agentes.
- RF-4: Detalle de equipo con miembros y agentes.
- RF-5: Enlace a especificación (XML/MD) si existe.

8. Requisitos No Funcionales

- RNF-1: Disponibilidad backend >= 99% durante demo.
- RNF-2: Carga inicial < 2s en red estable.
- RNF-3: Accesibilidad básica (contraste, tamaños).
- RNF-4: Seguridad básica (CORS, sanitización).
- RNF-5: Observabilidad mínima (logs y captura de errores).

9. Datos (Supabase) y Semillas

Tablas: agents, teams, members, team_members; attachments (post-MVP). Detalle de campos en `MVP_Descripcion.md` . Semillas: ≥3 equipos, ≥5 agentes, ≥6 miembros; ≥1 agente con specUrl a `product-owner-agent.xml` .

10. API REST (Spring Boot)

GET /api/agents → lista con team y owner básicos. GET /api/agents/{id} → detalle completo. GET /api/teams → lista con agentsCount. GET /api/teams/{id} → detalle con miembros y agentes. Errores: { code, message }.

11. Arquitectura e Integración

Flutter consume API REST. Spring Boot orquesta acceso a Supabase (PostgreSQL). Despliegue back con perfiles dev/prod; front web o móvil.

12. UX de Alto Nivel

Pantallas: HomeAgentes, AgenteDetalle, Equipos, EquipoDetalle, FichaDesarrollador. Estados: cargando, vacío, error con reintento.

13. Métricas de Éxito

Adopción (visitantes/instalaciones), engagement (vistas de detalle, tiempo de sesión), feedback cualitativo de stakeholders.

14. Roadmap (tentativo)

S1: Esquema Supabase, contrato API, esqueleto Flutter. S2: Agentes (lista/detalle). S3: Equipos y desarrolladores. S4: QA, hardening, docs y release.

15. Gobernanza (Agente Product Owner)

Principios: value_maximization (crítico), clarity_and_transparency (alto). Backlog: visible y priorizado; refinamiento semanal. DoR: historia clara, criterios, datos ejemplo y dependencias. DoD: integrado, manejo de errores/estados, pruebas básicas y documentación actualizada. Rol: PO define qué/por qué; equipo define cómo (no dicta arquitectura).