## **Roadmap Definitivo de SOUP** Market **(MVP)**

Este documento detalla el estado actual del proyecto SOUP Emprendimientos, la arquitectura implementada, las funcionalidades futuras planificadas por capítulos con sus respectivas tareas de UI/UX, backend y frontend, una visión general de la arquitectura completa del MVP, y un cronograma de desarrollo.

### 1. Funcionalidades Completadas a la Fecha

Hemos establecido una base sólida para la aplicación, cubriendo la gestión esencial para usuarios y emprendedores, y un listado público.

* **Capítulo 1: Gestión de Usuarios y Negocios**
  + **Backend:** Modelos Usuario y Negocio. Esquemas Pydantic para registro, login, perfil y CRUD de negocios. Lógica CRUD para usuarios y negocios. Routers para autenticación, perfil de usuario y negocios.
  + **Frontend:** Pantallas de LoginScreen, RegisterScreen, DashboardScreen, ProfileScreen, ManageBusinessesScreen, CreateBusinessScreen, EditBusinessScreen. Integración con API de autenticación y negocios.
  + **Funcionalidad:** Registro y login de usuarios. Edición de perfil. Creación, listado, edición y eliminación de negocios por parte de usuarios microemprendimiento o freelancer.
* **Capítulo 2: Gestión de Productos y Servicios**
  + **Backend:** Modelo Producto. Esquemas Pydantic para CRUD de productos. Lógica CRUD para productos (incluyendo asociación opcional a Negocio). Router para productos.
  + **Frontend:** Pantallas ManageProductsScreen, CreateProductScreen, EditProductScreen. Integración con API de productos. Selección de tipo de producto y asociación a negocio.
  + **Funcionalidad:** Creación, listado, edición y eliminación de productos/servicios por parte de usuarios microemprendimiento o freelancer.
* **Capítulo 3: Listado Público y Búsqueda Básica**
  + **Backend:** Esquema UserPublicResponse. Funciones CRUD para obtener *todos* los negocios y productos. Router público (/public) para exponer negocios, productos y perfiles de usuario sin autenticación.
  + **Frontend:** PublicListingScreen como página de inicio (/). Consumo de APIs públicas para mostrar listados de negocios y productos. Botones de login/registro en la página pública.
  + **Funcionalidad:** Acceso público a un listado de todos los negocios y productos. Visualización de perfiles de usuario públicos (información no sensible).

### 2. Arquitectura de Software Ya Implementada

La arquitectura actual está diseñada para ser modular, escalable y mantenible, siguiendo las mejores prácticas para aplicaciones web modernas.

* **Backend (FastAPI - Python):**
  + **Framework:** FastAPI (asíncrono, alto rendimiento).
  + **Base de Datos:** PostgreSQL.
  + **ORM:** SQLAlchemy (declarative models).
  + **Validación/Serialización:** Pydantic (para esquemas de datos).
  + **Autenticación:** JWT (JSON Web Tokens) gestionado con python-jose y passlib.
  + **Estructura de Archivos:**
    - app/main.py: Punto de entrada de la aplicación, configuración CORS, inclusión de routers.
    - app/database.py: Configuración de la conexión a la base de datos y sesión.
    - app/models.py: Definición de los modelos de base de datos (tablas).
    - app/schemas.py: Definición de los esquemas de validación y respuesta de datos (Pydantic).
    - app/dependencies.py: Funciones de inyección de dependencias (ej. get\_db, get\_current\_user).
    - app/crud/: Módulos con la lógica de operaciones CRUD para cada entidad (user.py, business.py, product.py).
    - app/routers/: Módulos con la definición de los endpoints API para cada sección (auth\_router.py, user\_router.py, business\_router.py, product\_router.py, public\_router.py).
* **Frontend (React - JavaScript):**
  + **Framework:** React.js.
  + **Routing:** React Router DOM.
  + **Estilos:** Tailwind CSS (para utilidades de CSS) y Shadcn UI (para componentes de UI accesibles y personalizables).
  + **Gestión de Estado:** React Context API (para autenticación).
  + **Comunicación con API:** fetch API nativa con funciones de ayuda en módulos dedicados.
  + **Estructura de Archivos:**
    - src/App.js: Configuración principal de rutas.
    - src/context/AuthContext.js: Contexto global para la autenticación.
    - src/api/: Módulos con funciones para interactuar con el backend (authApi.js, userApi.js, businessApi.js, productApi.js, publicApi.js).
    - src/components/ui/: Componentes de UI de Shadcn (botones, tarjetas, inputs, selects, etc.).
    - src/screens/: Componentes de pantalla para cada vista de la aplicación (Login, Register, Dashboard, Profile, ManageBusinesses, CreateBusiness, EditBusiness, ManageProducts, CreateProduct, EditProduct, PublicListing).

### 3. Nuevas Funcionalidades a Implementar (Por Capítulos)

Continuaremos con el desarrollo incremental, añadiendo valor en cada etapa.

#### Capítulo 4: Gestión de Insumos y Cálculo de Costos (Próximo)

* **Descripción UI/UX:**
  + **Dashboard:** Un nuevo botón "Gestionar Insumos" para emprendedores.
  + **Pantalla de Gestión de Insumos (ManageInsumosScreen):** Una tabla o lista de los insumos del usuario, con botones para "Crear Insumo", "Editar" y "Eliminar".
  + **Pantalla de Creación/Edición de Insumo (CreateInsumoScreen/EditInsumoScreen):** Formulario con campos para nombre, cantidad disponible, unidad de medida (ej. "kg", "litro", "unidad"), costo unitario de compra.
  + **Pantallas de Creación/Edición de Producto (CreateProductScreen/EditProductScreen):**
    - Sección "Insumos del Producto" con un botón "Añadir Insumo".
    - Al añadir, un modal o sección expandible que permita seleccionar un insumo existente (de la lista del usuario) y especificar la "cantidad necesaria" para *este producto*.
    - Visualización dinámica (o al guardar) del **COGS** y el **Precio Sugerido** para el producto, basados en los insumos asociados y el margen de ganancia.
* **Tareas de Backend:**
  + **backend/app/models.py:**
    - Crear modelo Insumo (id, nombre, cantidad\_disponible, unidad\_medida\_compra, costo\_unitario\_compra, usuario\_id, fecha\_creacion, fecha\_actualizacion).
    - Crear modelo ProductoInsumo (tabla de asociación id, producto\_id, insumo\_id, cantidad\_necesaria).
    - Modificar modelo Producto para incluir relaciones con ProductoInsumo y campos cogs y precio\_sugerido.
  + **backend/app/schemas.py:**
    - Definir InsumoCreate, InsumoUpdate, InsumoResponse.
    - Definir ProductoInsumoCreate, ProductoInsumoResponse.
    - Modificar ProductoCreate, ProductoUpdate, ProductoResponse para manejar la lista de ProductoInsumoCreate/Response.
  + **backend/app/crud/insumo.py:** Implementar CRUD para Insumo.
  + **backend/app/crud/product.py:**
    - Añadir lógica para crear/actualizar ProductoInsumo al crear/actualizar productos.
    - Implementar cálculo de cogs y precio\_sugerido basado en los insumos asociados y sus costos.
  + **backend/app/routers/insumo\_router.py:** Crear router con endpoints protegidos para el CRUD de insumos.
  + **backend/app/main.py:** Incluir insumo\_router.
* **Tareas de Frontend:**
  + **frontend/src/api/insumoApi.js:** Crear funciones para interactuar con los endpoints de insumos.
  + **frontend/src/screens/ManageInsumosScreen.js:** Crear componente para listar insumos.
  + **frontend/src/screens/CreateInsumoScreen.js:** Crear componente para formulario de creación de insumos.
  + **frontend/src/screens/EditInsumoScreen.js:** Crear componente para formulario de edición de insumos.
  + **frontend/src/screens/CreateProductScreen.js / EditProductScreen.js:** Modificar para integrar la selección de insumos, cantidad necesaria, y mostrar COGS/Precio Sugerido.
  + **frontend/src/App.js:** Añadir rutas para las pantallas de insumos.
  + **frontend/src/screens/DashboardScreen.js:** Añadir botón "Gestionar Insumos".

#### Capítulo 5: Gestión de Encargos (Pedidos)

* **Descripción UI/UX:**
  + **Dashboard:** Botón "Ver Encargos".
  + **Pantalla de Gestión de Encargos (ManageEncargosScreen):** Lista de encargos con estados (Pendiente de pago, En producción, Listo para envío, Completado, Cancelado). Botones para ver detalles, actualizar estado.
  + **Pantalla de Creación de Encargo (CreateEncargoScreen):** Formulario para registrar un nuevo encargo (seleccionar cliente, producto, cantidad, precio total, fecha estimada de entrega, notas, dirección de envío).
  + **Pantalla de Detalle/Edición de Encargo (EncargoDetailScreen):** Vista detallada del encargo, con historial de estados y opciones para actualizarlo.
  + **Integración WhatsApp (PMV):**
    - En la pantalla de detalle/edición de encargo, opciones para enviar notificaciones de estado por WhatsApp al cliente (ej. "Encargo Aceptado", "Listo para Retirar/Enviado").
    - En las pantallas de detalle de Producto/Negocio (públicas), un botón "Compartir por WhatsApp" que genere un enlace al item.
* **Tareas de Backend:**
  + **backend/app/models.py:**
    - Modelo Encargo (id, cliente\_id, producto\_id, cantidad, precio\_total, fecha\_encargo, fecha\_entrega\_estimada, estado, notas, direccion\_envio, progreso\_envio, usuario\_id).
    - Relaciones con Usuario (gestor) y Producto.
  + **backend/app/schemas.py:** Definir EncargoCreate, EncargoUpdate, EncargoResponse.
  + **backend/app/crud/encargo.py:** Implementar CRUD para Encargo.
  + **backend/app/routers/encargo\_router.py:** Crear router con endpoints protegidos para el CRUD de encargos.
  + **Integración WhatsApp (Backend):**
    - Configuración para almacenar credenciales de WhatsApp Business API (ej. en settings.py o nueva tabla WhatsAppConfig).
    - Lógica para interactuar con la API de WhatsApp Business para enviar mensajes de plantilla.
    - Integrar llamadas a esta lógica en las funciones CRUD de Encargo (al cambiar de estado).
* **Tareas de Frontend:**
  + **frontend/src/api/encargoApi.js:** Crear funciones para interactuar con los endpoints de encargos.
  + **frontend/src/screens/ManageEncargosScreen.js:** Crear componente para listar encargos.
  + **frontend/src/screens/CreateEncargoScreen.js:** Crear componente para formulario de creación de encargos.
  + **frontend/src/screens/EncargoDetailScreen.js:** Crear componente para ver y editar detalles de encargo, incluyendo cambio de estado.
  + **Integración WhatsApp (Frontend):**
    - Botones para enviar notificaciones de WhatsApp en EncargoDetailScreen.
    - Botones "Compartir por WhatsApp" en PublicListingScreen (en tarjetas de producto/negocio) y futuras pantallas de detalle público.
  + **frontend/src/App.js:** Añadir rutas para las pantallas de encargos.
  + **frontend/src/screens/DashboardScreen.js:** Añadir botón "Ver Encargos".

#### Capítulo 6: Reseñas y Calificaciones

* **Descripción UI/UX:**
  + **Pantalla de Detalle de Encargo:** Después de un encargo "Completado", un botón "Dejar Reseña" que lleve a un formulario.
  + **Formulario de Reseña (CreateReviewScreen):** Campo para calificación (estrellas 1-5) y comentario.
  + **Pantallas de Detalle Público (Negocio/Producto):** Mostrar el rating\_promedio y reviews\_count.
* **Tareas de Backend:**
  + **backend/app/models.py:** Modelo Review (id, encargo\_id, rating, comentario, fecha\_review, usuario\_id).
  + **backend/app/schemas.py:** Definir ReviewCreate, ReviewResponse.
  + **backend/app/crud/review.py:** Implementar CRUD para Review.
  + **backend/app/crud/negocio.py / backend/app/crud/product.py:** Lógica para actualizar rating\_promedio y reviews\_totales/reviews\_count al crear una nueva reseña.
  + **backend/app/routers/review\_router.py:** Crear router con endpoints protegidos para la creación de reseñas.
* **Tareas de Frontend:**
  + **frontend/src/api/reviewApi.js:** Crear funciones para interactuar con los endpoints de reseñas.
  + **frontend/src/screens/EncargoDetailScreen.js:** Añadir botón "Dejar Reseña" condicional.
  + **frontend/src/screens/CreateReviewScreen.js:** Crear componente para el formulario de reseña.
  + **Pantallas de Detalle Público (futuras):** Mostrar rating\_promedio y reviews\_count.
  + **frontend/src/App.js:** Añadir rutas para las pantallas de reseña.

#### Capítulo 7: Publicidad y Monetización

* **Descripción UI/UX:**
  + **Dashboard:** Botón "Gestionar Publicidad" (para usuarios Premium).
  + **Pantalla de Gestión de Publicidad (ManageAdsScreen):** Lista de anuncios del usuario, con estado (activo/inactivo), visualizaciones, clics. Botones para crear, editar, pausar/activar.
  + **Pantalla de Creación/Edición de Anuncio (CreateAdScreen/EditAdScreen):** Formulario para seleccionar un negocio o producto a publicitar, tipo de publicidad, fechas de inicio/fin, costo.
  + **Pantallas Públicas (PublicListingScreen / Detalles):** Espacios designados para mostrar anuncios (ej. "Anuncios Destacados").
* **Tareas de Backend:**
  + **backend/app/models.py:** Modelo Publicidad (id, item\_publicitado\_id, tipo\_publicidad, fecha\_inicio, fecha\_fin, costo, usuario\_id, estado, visualizaciones, clics, conversiones, fecha\_creacion, fecha\_actualizacion).
  + **backend/app/schemas.py:** Definir PublicidadCreate, PublicidadUpdate, PublicidadResponse.
  + **backend/app/crud/publicidad.py:** Implementar CRUD para Publicidad.
  + **Lógica de Tracking:** Implementar lógica para registrar visualizaciones y clics en los anuncios.
  + **backend/app/routers/publicidad\_router.py:** Crear router con endpoints protegidos para la gestión de publicidad.
  + **backend/app/routers/public\_router.py:** Añadir endpoints públicos para obtener anuncios activos.
* **Tareas de Frontend:**
  + **frontend/src/api/publicidadApi.js:** Crear funciones para interactuar con los endpoints de publicidad.
  + **frontend/src/screens/ManageAdsScreen.js:** Crear componente para gestionar anuncios.
  + **frontend/src/screens/CreateAdScreen.js / EditAdScreen.js:** Crear componentes para formularios de anuncios.
  + **frontend/src/screens/PublicListingScreen.js / Pantallas de Detalle Público (futuras):** Integrar componentes para mostrar anuncios.
  + **frontend/src/App.js:** Añadir rutas para las pantallas de publicidad.
  + **frontend/src/screens/DashboardScreen.js:** Añadir botón "Gestionar Publicidad".

#### Capítulo 8: Integración de IA (SOUP Flow AI - Primeros Pasos)

* **Descripción UI/UX:**
  + **Dashboard/Panel de Control:** Una nueva sección "Asistente SOUP AI" con un campo de texto para preguntas o un botón para generar sugerencias.
  + **Generación de Descripciones:** En CreateProductScreen/EditProductScreen, un botón "Generar Descripción con IA" que use el nombre del producto y tipo para generar texto.
  + **Generación de Mensajes WhatsApp:** En EncargoDetailScreen, un botón "Sugerir Mensaje WhatsApp" que genere un borrador basado en el estado del encargo.
* **Tareas de Backend:**
  + **backend/app/services/ai\_service.py:** Nuevo módulo para encapsular la lógica de llamadas a la Gemini API.
  + **Endpoints de IA:** Nuevos endpoints en routers existentes (ej. product\_router, encargo\_router) o un nuevo ai\_router para:
    - POST /ai/generate-description (para productos/servicios).
    - POST /ai/suggest-whatsapp-message (para encargos).
    - POST /ai/chat (para un chatbot básico).
  + **Integración:** Utilizar la Gemini API (gemini-2.0-flash para texto) para generar contenido.
* **Tareas de Frontend:**
  + **frontend/src/api/aiApi.js:** Crear funciones para interactuar con los nuevos endpoints de IA.
  + **frontend/src/screens/CreateProductScreen.js / EditProductScreen.js:** Añadir botón y lógica para generar descripciones.
  + **frontend/src/screens/EncargoDetailScreen.js:** Añadir botón y lógica para sugerir mensajes de WhatsApp.
  + **frontend/src/screens/DashboardScreen.js:** Posiblemente un campo de texto para un chatbot básico o un área de sugerencias.

#### Capítulo 9: Notificaciones y Mensajería (Consolidación)

* **Descripción UI/UX:**
  + **Dashboard:** Un icono de campana para notificaciones.
  + **Panel de Notificaciones:** Un modal o página para ver una lista de notificaciones (ej. "Tarea de proyecto actualizada", "Nuevo encargo", "Reseña recibida").
  + **Mensajería Básica (dentro de SOUP Projects):** Un hilo de comentarios o chat simple dentro de la vista de proyecto.
* **Tareas de Backend:**
  + **backend/app/models.py:** Modelo Notificacion (id, usuario\_id, tipo, contenido, leida, fecha).
  + **backend/app/schemas.py:** Definir esquemas para Notificacion.
  + **backend/app/crud/notification.py:** Implementar CRUD para notificaciones.
  + **Disparadores de Notificaciones:** Modificar lógica CRUD en otros módulos (ej. encargo.py, review.py, insumo.py, project.py - futuro) para crear notificaciones.
  + **backend/app/routers/notification\_router.py:** Crear router para obtener y marcar notificaciones como leídas.
  + **Consolidación WhatsApp:** Asegurar que la lógica de WhatsApp del Capítulo 5 esté bien encapsulada y sea reutilizable.
* **Tareas de Frontend:**
  + **frontend/src/api/notificationApi.js:** Crear funciones para interactuar con los endpoints de notificaciones.
  + **Componente de Notificaciones:** Un componente reutilizable para mostrar un icono de campana con contador y un panel de notificaciones.
  + **frontend/src/screens/DashboardScreen.js:** Integrar el componente de notificaciones.
  + **frontend/src/screens/ProjectDetailScreen.js (futuro):** Implementar el hilo de comentarios/chat.

#### Capítulo 10: Búsqueda Avanzada y Filtrado (Público)

* **Descripción UI/UX:**
  + **PublicListingScreen:** Añadir una barra de búsqueda con autocompletado y filtros (por rubro, tipo de producto, ubicación, rango de precios, rating).
  + **Resultados de Búsqueda:** Mostrar los resultados de forma clara y paginada.
* **Tareas de Backend:**
  + **backend/app/routers/public\_router.py:** Modificar los endpoints GET /public/businesses y GET /public/products para aceptar parámetros de consulta (query parameters) para búsqueda y filtrado.
  + **backend/app/crud/business.py / backend/app/crud/product.py:** Modificar las funciones get\_all\_businesses y get\_all\_products para aplicar los filtros y la lógica de búsqueda.
* **Tareas de Frontend:**
  + **frontend/src/screens/PublicListingScreen.js:** Implementar la barra de búsqueda y los controles de filtro.
  + **Lógica de Estado:** Gestionar el estado de los filtros y la consulta de búsqueda.
  + **Actualización de API:** Modificar las llamadas a publicApi.js para incluir los parámetros de búsqueda y filtro.

#### Capítulo 11: Refinamiento de Roles de Usuario y Permisos

* **Descripción UI/UX:**
  + **Registro:** Asegurar que la selección de tipo\_tier sea clara.
  + **Dashboard/Navegación:** Asegurar que los botones y enlaces solo sean visibles para los roles de usuario adecuados (ej. "Gestionar Publicidad" solo para Premium).
  + **Mensajes de Error:** Mostrar mensajes claros si un usuario intenta acceder a una funcionalidad sin los permisos adecuados.
* **Tareas de Backend:**
  + **backend/app/dependencies.py:** Crear funciones de dependencia para verificar roles (ej. require\_microemprendedor\_or\_freelancer).
  + **Routers:** Aplicar estas dependencias a los endpoints API para asegurar que solo los roles autorizados puedan acceder a ciertas funcionalidades.
  + **Lógica de Negocio:** Refinar cualquier lógica de negocio que dependa del tipo\_tier del usuario.
* **Tareas de Frontend:**
  + **Componentes de Navegación/Botones:** Implementar lógica condicional para renderizar elementos de UI basados en el user.tipo\_tier del AuthContext.
  + **Manejo de Errores:** Mostrar mensajes amigables al usuario si intenta acceder a rutas no autorizadas.

#### Capítulo 12: Despliegue y Consideraciones de Escalabilidad

* **Descripción UI/UX:** N/A (principalmente tareas de infraestructura).
* **Tareas de Backend:**
  + **Contenedorización:** Crear Dockerfile para la aplicación FastAPI.
  + **Configuración de Producción:** Variables de entorno para base de datos, JWT secrets, etc.
  + **Optimización:** Considerar Gunicorn/Uvicorn workers.
* **Tareas de Frontend:**
  + **Optimización de Build:** npm run build.
  + **Configuración de Producción:** Variables de entorno para la URL del backend.
* **Consideraciones Generales:**
  + **Base de Datos:** Optimización de consultas, índices.
  + **Seguridad:** Hardening de la API, protección contra ataques comunes.
  + **Monitoring:** Herramientas básicas de monitoreo de rendimiento y errores.

#### Capítulo 13: SOUP Projects (Gestión Colaborativa Simple)

* **Descripción UI/UX:**
  + **Pantalla de Detalle de Encargo/Producto:** Opción "Crear Proyecto Colaborativo" o "Invitar Colaboradores".
  + **Pantalla de Proyecto (ProjectDetailScreen):**
    - Vista de detalles del proyecto (descripción, plazos).
    - Lista de "Tareas" con checkboxes de estado (Pendiente, En Proceso, Completada).
    - Sección de "Notas" o "Comentarios" con un feed cronológico.
    - Botón "Invitar Colaborador" (para el propietario del proyecto).
  + **Correo Electrónico:** Plantilla de email para invitaciones a colaboradores.
* **Tareas de Backend:**
  + **backend/app/models.py:**
    - Modelo Project (id, encargo\_id (opcional), owner\_id, name, description, status, due\_date).
    - Modelo Task (id, project\_id, description, assigned\_to\_user\_id, status).
    - Modelo ProjectCollaborator (project\_id, user\_id, role).
    - Modelo ProjectNote (id, project\_id, user\_id, content, timestamp).
  + **backend/app/schemas.py:** Definir esquemas para Project, Task, ProjectCollaborator, ProjectNote.
  + **backend/app/crud/project.py:** Implementar CRUD para Project, Task, ProjectCollaborator, ProjectNote.
  + **Lógica de Invitación:** Generación de tokens de invitación, envío de emails.
  + **backend/app/routers/project\_router.py:** Crear router con endpoints protegidos para la gestión de proyectos.
* **Tareas de Frontend:**
  + **frontend/src/api/projectApi.js:** Crear funciones para interactuar con los endpoints de proyectos.
  + **frontend/src/screens/EncargoDetailScreen.js / ProductDetailScreen.js:** Añadir botón "Crear Proyecto".
  + **frontend/src/screens/ProjectDetailScreen.js:** Crear componente para la vista de proyecto (tareas, notas, invitación).
  + **frontend/src/App.js:** Añadir rutas para las pantallas de proyectos.
  + **frontend/src/screens/DashboardScreen.js:** Añadir botón "Mis Proyectos".

#### Capítulo 14: Historias de Negocio y Contenido Multimedia

* **Descripción UI/UX:**
  + **Dashboard/Panel de Negocio:** Un botón "Añadir Historia" o una sección "Mis Actualizaciones".
  + **Formulario de Creación de Historia (CreateStoryScreen):** Campo de texto corto, botón para subir una única foto.
  + **Perfil Público del Negocio (Pantalla de Detalle de Negocio):** Una nueva sección "Historias" o "Novedades" con un feed cronológico de tarjetas de historias (foto + texto corto).
* **Tareas de Backend:**
  + **backend/app/models.py:** Modelo Story (id, business\_id, user\_id, text\_content, image\_url, location\_text (opcional), timestamp, is\_public).
  + **backend/app/schemas.py:** Definir esquemas para Story.
  + **backend/app/crud/story.py:** Implementar CRUD para Story.
  + **Servicio de Almacenamiento de Archivos:** Integración con un servicio de almacenamiento en la nube (ej. Google Cloud Storage, Firebase Storage) para subir y gestionar imágenes.
  + **backend/app/routers/story\_router.py:** Crear router con endpoints protegidos para la gestión de historias.
  + **backend/app/routers/public\_router.py:** Añadir endpoint público para obtener las historias de un negocio.
* **Tareas de Frontend:**
  + **frontend/src/api/storyApi.js:** Crear funciones para interactuar con los endpoints de historias.
  + **frontend/src/screens/CreateStoryScreen.js:** Crear componente para el formulario de creación de historias, incluyendo subida de imagen.
  + **frontend/src/screens/BusinessDetailScreen.js (futura pantalla de detalle público de negocio):** Integrar un componente para mostrar el feed de historias.
  + **frontend/src/App.js:** Añadir rutas para las pantallas de historias.
  + **frontend/src/screens/DashboardScreen.js:** Añadir botón "Mis Historias".

### 4. Arquitectura General del PMV Completo y Valor de Referencia

El MVP completo de SOUP Emprendimientos se construirá sobre una arquitectura de microservicios lógicos (aunque implementados en una única aplicación por simplicidad inicial) con una separación clara de responsabilidades entre el frontend y el backend.

* **Frontend:**
  + **Tecnologías:** React.js, React Router DOM, Tailwind CSS, Shadcn UI.
  + **Propósito:** Interfaz de usuario interactiva y responsiva para emprendedores y clientes.
  + **Despliegue:** Idealmente en un servicio de hosting de frontend como Vercel o Netlify.
* **Backend:**
  + **Tecnologías:** FastAPI (Python), PostgreSQL, SQLAlchemy, Pydantic, JWT.
  + **Propósito:** API RESTful para la lógica de negocio, gestión de datos y comunicación con la base de datos.
  + **Despliegue:** En un servidor virtual privado (VPS) o un servicio PaaS (Platform as a Service) como Render, Heroku, o Google Cloud Run.
* **Base de Datos:**
  + **Tecnología:** PostgreSQL.
  + **Propósito:** Almacenamiento persistente de todos los datos de la aplicación.
  + **Despliegue:** Puede ser autohospedada en el mismo VPS que el backend o como un servicio de base de datos gestionado (ej. AWS RDS, DigitalOcean Managed Databases).
* **Servicios Externos (Futuros/Opcionales):**
  + **Almacenamiento de Archivos:** Google Cloud Storage, AWS S3, Firebase Storage (para imágenes de productos/historias).
  + **Servicio de Email:** SendGrid, Mailgun (para invitaciones, notificaciones).
  + **API de WhatsApp Business:** Para notificaciones y mensajería.
  + **Modelos de IA:** Gemini API (para generación de texto, análisis).

**Valor del MVP:**

El valor de este MVP radica en proporcionar una **herramienta integral para microemprendedores y freelancers** que les permita:

* **Profesionalizar su presencia online:** Con perfiles públicos, listados de productos/servicios.
* **Optimizar la gestión interna:** Control de negocios, productos, insumos, costos y encargos.
* **Mejorar la comunicación con clientes:** A través de notificaciones y futuras integraciones.
* **Obtener insights:** Con cálculos de costos y futuras analíticas.
* **Reducir la fricción:** Al centralizar herramientas que usualmente están dispersas.

Para los **clientes**, el valor es un **directorio curado y transparente** donde pueden encontrar proveedores de confianza con historial de producción y reseñas.

Este MVP establece una base sólida para futuras expansiones hacia un ecosistema completo de emprendimiento, con potencial para monetización a través de planes premium y publicidad.

### 5. Cronograma de Desarrollo (Objetivo: MVP Online a Finales de Septiembre 2025)

Este cronograma es una estimación ambiciosa pero alcanzable, asumiendo un desarrollo enfocado. Se basa en el 7 de julio de 2025 como fecha de inicio.

* **Semana 1 (Jul 07 - Jul 13):**
  + **Capítulo 4: Gestión de Insumos y Cálculo de Costos (Backend completo)**
    - Modelos Insumo, ProductoInsumo y modificación de Producto.
    - Esquemas Pydantic para Insumos y ProductoInsumo, modificación de esquemas de Producto.
    - CRUD para Insumos.
    - Modificación CRUD de Productos (cálculo COGS/Precio Sugerido, manejo ProductoInsumo).
    - Routers para Insumos.
    - Integración en main.py.
    - **Deadline:** Backend de Capítulo 4 operativo y probado en Swagger UI.
* **Semana 2 (Jul 14 - Jul 20):**
  + **Capítulo 4: Gestión de Insumos y Cálculo de Costos (Frontend completo)**
    - API Services para Insumos.
    - Pantallas ManageInsumosScreen, CreateInsumoScreen, EditInsumoScreen.
    - Integración de Insumos en CreateProductScreen y EditProductScreen (UI para añadir/gestionar insumos, mostrar COGS/Precio Sugerido).
    - Actualización de Rutas y Navegación (App.js, DashboardScreen.js).
    - **Deadline:** Frontend de Capítulo 4 operativo y probado.
* **Semana 3 (Jul 21 - Jul 27):**
  + **Capítulo 5: Gestión de Encargos (Backend completo)**
    - Modelo Encargo.
    - Esquemas Pydantic para Encargos.
    - CRUD para Encargos.
    - Routers para Encargos.
    - Integración en main.py.
    - **Deadline:** Backend de Capítulo 5 operativo y probado en Swagger UI.
* **Semana 4 (Jul 28 - Ago 03):**
  + **Capítulo 5: Gestión de Encargos (Frontend completo)**
    - API Services para Encargos.
    - Pantallas ManageEncargosScreen, CreateEncargoScreen, EncargoDetailScreen.
    - Actualización de Rutas y Navegación.
    - **Capítulo 3: Actualización Frontend (Botones Compartir WhatsApp)**
      * Añadir botones "Compartir por WhatsApp" en PublicListingScreen (tarjetas).
    - **Deadline:** Frontend de Capítulo 5 operativo y probado.
* **Semana 5 (Ago 04 - Ago 10):**
  + **Capítulo 6: Reseñas y Calificaciones (Backend y Frontend)**
    - Modelos, esquemas, CRUD, routers para Review.
    - Lógica para actualizar ratings en Negocios/Productos.
    - API Services y Pantallas de Frontend para reseñas.
    - **Deadline:** Reseñas funcionales.
* **Semana 6 (Ago 11 - Ago 17):**
  + **Capítulo 7: Publicidad y Monetización (Backend y Frontend)**
    - Modelos, esquemas, CRUD, routers para Publicidad.
    - Lógica de tracking (visualizaciones/clics).
    - API Services y Pantallas de Frontend para gestión de publicidad.
    - Integración de anuncios en PublicListingScreen.
    - **Deadline:** Publicidad básica operativa.
* **Semana 7 (Ago 18 - Ago 24):**
  + **Capítulo 14: Historias de Negocio y Contenido Multimedia (Backend y Frontend)**
    - Modelos, esquemas, CRUD, routers para Story.
    - Configuración de almacenamiento de archivos (simulada o con un servicio real si se decide).
    - API Services y Pantallas de Frontend para crear/listar historias.
    - Integración de historias en la futura pantalla de detalle público de Negocio.
    - **Deadline:** Historias de negocio funcionales.
* **Semana 8 (Ago 25 - Ago 31):**
  + **Capítulo 13: SOUP Projects (Gestión Colaborativa Simple - Backend)**
    - Modelos Project, Task, ProjectCollaborator, ProjectNote.
    - Esquemas, CRUD, routers.
    - Lógica de invitación por email (simulada o con servicio real).
    - **Deadline:** Backend de Proyectos operativo.
* **Semana 9 (Sep 01 - Sep 07):**
  + **Capítulo 13: SOUP Projects (Gestión Colaborativa Simple - Frontend)**
    - API Services y Pantallas de Frontend para Proyectos (creación, detalle, tareas, notas).
    - Integración en EncargoDetailScreen.
    - **Deadline:** Proyectos colaborativos básicos funcionales.
* **Semana 10 (Sep 08 - Sep 14):**
  + **Capítulo 8: Integración de IA (Primeros Pasos - Backend y Frontend)**
    - Integración Gemini API para generación de descripciones de productos y sugerencia de mensajes WhatsApp.
    - UI para estas funcionalidades en las pantallas de producto y encargo.
    - **Capítulo 9: Notificaciones y Mensajería (Backend - Consolidación WhatsApp)**
      * Modelo Notificacion.
      * Disparadores de notificaciones.
      * Consolidación de la lógica de envío de WhatsApp.
    - **Deadline:** Funcionalidades de IA y Notificaciones Backend operativas.
* **Semana 11 (Sep 15 - Sep 21):**
  + **Capítulo 9: Notificaciones y Mensajería (Frontend)**
    - API Services y Componente de Notificaciones (campana, panel).
    - Integración en DashboardScreen.
  + **Capítulo 10: Búsqueda Avanzada y Filtrado (Backend y Frontend)**
    - Modificación de endpoints públicos para aceptar filtros.
    - Implementación de UI de búsqueda y filtros en PublicListingScreen.
    - **Deadline:** Notificaciones y Búsqueda Avanzada funcionales.
* **Semana 12 (Sep 22 - Sep 30):**
  + **Capítulo 11: Refinamiento de Roles de Usuario y Permisos (Backend y Frontend)**
    - Implementación de dependencias de roles en el backend.
    - Ajustes de visibilidad de UI basados en roles.
  + **Capítulo 12: Despliegue y Consideraciones de Escalabilidad**
    - Dockerización (si aplica).
    - Configuración de variables de entorno para producción.
    - Despliegue inicial del MVP en un entorno público.
  + **Revisión General y Bug Fixing:** Pruebas finales, corrección de errores, ajustes menores de UI/UX.
  + **Deadline:** **MVP Online a finales de Septiembre 2025.**

Este roadmap es un plan de alto nivel. Durante el desarrollo, pueden surgir desafíos inesperados que requieran ajustar las estimaciones. La clave será la comunicación constante y la priorización.