# Guía Técnica de Implementación Frontend (PWA React + Vite)

## 1. Estructura basada en Diseño Atómico

La interfaz sigue la metodología **Atomic Design**, descomponiendo la UI en Átomos, Moléculas, Organismos, Plantillas y Páginas para asegurar consistencia y reutilización. A continuación se definen cada nivel y sus componentes principales:

**Átomos:** Son los elementos básicos de la UI, con estilo definido según los tokens de diseño. Incluyen componentes simples como botones, iconos, campos de texto y etiquetas tipográficas. También se consideran átomos las **variables de estilo** globales (colores, tipografías, espaciados) que conforman la base visual.

| **Átomo** | **Descripción** |
| --- | --- |
| **Color (Token)** | Valor de color fijo usado en la UI. Se definen paletas para distintos roles (primario, secundario, fondo, texto, estados) y **se usan los hex exactos proporcionados** (ej. verde primario #0A7D2B para botones, azul info #2196F3 para indicadores offline)[.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Primario) |
| **Tipografía (Token)** | Estilos de fuente base según Material Design 3: fuente Roboto en pesos 400/500/700; tamaños base Título ~20–22px, Subtítulo ~16px, Cuerpo ~14-16px, Caption ~12px[. Se aplican line-heights de 1.3–1.5 para legibilidad.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Tamaños%20base%3A) |
| **Icono** | Ícono individual (Material Icons Outlined). Tamaño estándar 24×24dp (24px)[. Usa color primario para iconos activos (](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Estilo%3A%20Material%20Icons%20Outlined)#0A7D2B), color secundario atenuado para inactivos (#666666 al 60%) y color info azul para estado offline (#2196F3)[.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Estilo%3A%20Material%20Icons%20Outlined) |
| **Texto** | Nodo de texto o componente tipográfico (por ejemplo <Text>). Aplica un estilo de fuente específico (ej. **Título** 20px/700, **Cuerpo** 16px/400, etc.) usando los tokens de tipografía definidos (colores #333 para primario, #666 para secundario)[.](file://file_000000004bdc61f490771690e57e336d#:~:text=%22text%22%3A%20%7B%20%22primary%22%3A%20%22,) |
| **Botón (Base)** | Botón clicable sin estilo específico. En código puede ser un <button> o componente base con estilos heredables. Sirve de base para construir botones primarios, secundarios, etc. |
| **Input (Base)** | Campo de entrada de texto nativo (<input>/<textarea>) con estilo básico. Incluye los sub-elementos mínimos (sin label ni iconos). Se extiende para crear campos de formulario completos. |
| **Spinner de carga** | Indicador de carga simple (animación circular) para mostrar estados de espera. Tamaño ~24dp. Usará color primario o neutro según el contexto de fondo. |

**Moléculas:** Son grupos simples de átomos que funcionan como una unidad. Por ejemplo, un campo de texto completo con su etiqueta y mensaje de error, o una tarjeta/list item que combina icono + textos + botón. Son componentes reutilizables que encapsulan una pequeña funcionalidad.

| **Molécula** | **Descripción** |
| --- | --- |
| **Botón Primario** | Botón estilizado principal (<PrimaryButton>), construido del átomo base. Presenta fondo color primario verde #0A7D2B y texto blanco, con radio de borde 12dp[. Se usa para acciones CTA destacadas (e.g. “Guardar”, “Finalizar”).](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Dimensiones%3A%20hasta%20312%20×%2048,altura%20mínima%2048%20dp) |
| **Botón Secundario** | Variante de botón para acciones secundarias. Estilo de fondo claro o borde delineado (1px #0A7D2B) y texto color primario. Mismo tamaño (48dp alto) que el primario para mantener área táctil mínima[.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Dimensiones%3A%20hasta%20312%20×%2048,altura%20mínima%2048%20dp) |
| **Chip/Pill** | Etiqueta visual pequeña con texto, usada como filtro o indicador. Ej: chips de filtro “Pendientes (3) / Todos / Activos” en la pantalla principal. Tienen padding horizontal ~8dp, altura ~32dp, bordes redondeados (16dp) y colores según estado (ej. fondo verde suave para chip seleccionado, texto blanco). |
| **Campo de Texto (Input)** | Campo de formulario completo (<TextField>), combinando átomo de input + etiqueta + ayuda. Altura 56dp[, incluye:](file://file_0000000043dc61fa9eefb470199a33df#:~:text=5) **Label** (12px, color #666) encima o como placeholder, **Input** editable con borde inferior gris claro #F5F5F5[, y texto de ayuda/error (12px) debajo en gris #666 o rojo #D8262C para errores.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=5) |
| **Ítem de Lista de Cliente** | Tarjeta/row para mostrar un cliente en la lista principal. Combina: ícono/avatar circular con inicial del cliente, nombre del cliente (bold 16px), subtítulo con razón de visita (14px) y un indicador de prioridad si aplica. Fondo blanco con borde/redondez leve (radio 8–12dp), separado por margen de la lista. Al hacer click se navega al detalle. |
| **Ítem de Recomendación** | Componente para mostrar una recomendación de IA en la pantalla de detalle. Incluye: título corto de la acción (ej. “Sugerir surtido XX Lager 355” – tipografía 14px bold), descripción/resumen breve (14px regular), etiqueta de vigencia (ej. “Hoy”, “Esta semana”) y un botón secundario “Registrar feedback”. Se presenta como una sección con fondo claro (blanco) dentro de la tarjeta de detalle, con cada recomendación separada por un divisor o espaciado. |
| **Banner Offline** | Franja informativa mostrada en la parte superior cuando no hay conexión. Altura 48dp[, color de fondo azul info (](file://file_0000000043dc61fa9eefb470199a33df#:~:text=match%20at%20L265%20interacción,“Usar%20remoto”%2C%20detalles%20mínimos)#2196F3) y texto blanco centrado (“Sin conexión. Trabajando localmente.”)[. Aparece fijamente sobre la UI para notificar estado offline, desapareciendo al reconectar (pero no removible manualmente mientras persista offline)](file://file_0000000043dc61fa9eefb470199a33df#:~:text=match%20at%20L270%20Microcopy%20,¿Qué%20versión%20quieres)[.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=match%20at%20L317%20No%20ocultar,el%20banner%20offline%20cuando%20offline) |
| **Indicador Local (Offline)** | Icono estático que indica datos guardados localmente. Representado por un **disco verde** (#4CAF50) en una esquina de la pantalla[. Por ejemplo, se ubica en la esquina inferior derecha para indicar que los cambios se han guardado en el dispositivo](file://file_0000000043dc61fa9eefb470199a33df#:~:text=interacción,“Usar%20remoto”%2C%20detalles%20mínimos)[. Este icono aparece cuando hay datos pendientes de sincronizar, y desaparece al confirmar sincronización exitosa.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=match%20at%20L265%20interacción,“Usar%20remoto”%2C%20detalles%20mínimos) |

**Organismos:** Son componentes complejos compuestos por múltiples moléculas y/o átomos, formando secciones completas de la interfaz. Normalmente encapsulan una funcionalidad autónoma dentro de la página.

| **Organismo** | **Descripción** |
| --- | --- |
| **Header (Barra Superior)** | Cabecera fija de la app (Top App Bar). Contiene el título de pantalla (ej. “Hoy • Vie 26 Sep” en la principal, o nombre del cliente en el detalle) y puede incluir iconos de estado (ej. icono de nube tachada para offline en la barra)[. Diseño M3: altura ~56dp, color de fondo blanco, texto título 20px #333.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=match%20at%20L247%20Offline%3A%20icono,solo%20indicativo%2C%20no%20CTA) |
| **Lista de Clientes (Hoja de Ruta)** | Lista scrollable que agrupa múltiples ítems de cliente (moléculas). Implementa un patrón de lista master: muestra los clientes del día con su prioridad. Puede incluir un filtro o tabs para segmentar (Pendientes/Todos/Activos). Cada ítem de cliente muestra nombre y contexto principal (“Razón: ...”). La lista utiliza un fondo neutro (#F5F5F5) y cada tarjeta de cliente con elevación leve (sombra nivel e1) para separarlas visualmente. |
| **Detalle de Visita (Cliente)** | Vista de detalle para un cliente específico. Contiene organismos secundarios: sección **Razón de visita** (motivo principal o prioridad de la visita, e.g. “Cliente con Alto Riesgo de Abandono” destacada en chip rojo de prioridad), sección **Recomendaciones** (lista de componentes Ítem de Recomendación, usualmente 2–3 acciones concretas a ejecutar)[, posiblemente pestañas para alternar sub-secciones (Resumen / Recomendaciones / KPIs). Incluye un botón flotante o fijo “Iniciar” para marcar comienzo de visita (registro de llegada), y al finalizar se habilita opción de terminar visita.](file://file_00000000009061f993ee5e4e33fed8dc#:~:text=Nivel%201%3A%20Prioridad%20de%20la,con%20Alto%20Riesgo%20de%20Abandono) |
| **Formulario de Feedback** | Módulo que aparece al registrar feedback de una recomendación. Puede ser un modal o pantalla aparte. Contiene campos para el resultado de la interacción (ej. opciones “Aceptó”, “Rechazó” con sus motivos)[, calificación (ej. escala 1-5 estrellas) y comentario opcional](file://file_00000000009061f993ee5e4e33fed8dc#:~:text=Adherencia%20a%20la%20Recomendación%3A%20Botones,“¿Ofreció%20el%20producto%20X%3F”%3A%20Sí%2FNo)[. Incluye botones para cancelar o guardar. Este organismo utiliza componentes atómicos (checkbox/radio, textarea, etc.) combinados, y valida entrada requerida antes de permitir guardar.](file://file_00000000009061f993ee5e4e33fed8dc#:~:text=Calificación%20de%20la%20Recomendación%3A%20Una,la%20utilidad%20de%20la%20recomendación) |
| **Módulo Offline** | Conjunto de elementos UI relacionados con la conectividad. Incluye el **Banner Offline** y **Indicador Local** (descritos arriba), así como un posible indicador de sincronización en progreso (icono animado o spinner junto a un texto “Sincronizando…”) cuando haya envíos pendientes[. Estos elementos funcionan en conjunto para informar al usuario del estado de red y guardado local en todo momento, cumpliendo el principio de](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Estados%20previstos%201,“Sincronizando…”%29%20sin%20bloquear) **Visibilidad de Estado**[.](file://file_00000000b21461f690ca97d7085c6574#:~:text=de%20forma%20clara%20,marcar%20una%20visita%20como%20completada) |

**Plantillas:** Estructuras de diseño que colocan organismos en un layout específico, sirviendo como base para páginas. Por ejemplo, una plantilla de **Pantalla Principal** podría definir: header fijo con título + filtros, contenido central con la lista de clientes, y footer con botones de acciones rápidas (ej. acceso a formularios de inteligencia de mercado y botón de sincronizar). Otra plantilla, la de **Detalle de Cliente**, organiza el header con nombre del cliente, secciones de pestañas, y un footer con botón primario (ej. “Finalizar visita”) fijo en la zona inferior (thumb-friendly).

**Páginas:** Son instancias concretas de plantillas con datos reales. Para este PWA se identifican páginas clave del flujo del asesor: - **Pantalla Principal (Hoy)** – Muestra la “Hoja de Ruta” del día con la lista de clientes priorizados y accesos a funciones adicionales (ver sección 5). - **Pantalla de Detalle de Cliente** – Muestra la información específica de la visita a un cliente (motivo y recomendaciones de IA) y permite registrar interacciones. - **Pantalla de Registro de Visita** – Interfaz para cerrar una visita: resumen de lo ocurrido, validación de ubicación y envío de feedback (equivale al formulario Finalizar Visita). - **Pantallas de Inteligencia de Mercado** – Formularios para capturar Quiebres de stock, Inventario, Competencia, etc., accesibles desde la principal (cada uno con sus campos específicos según definido en la propuesta[).](file://file_00000000009061f993ee5e4e33fed8dc#:~:text=Presencia%20e%20Inventario%20de%20la,Competencia)

Cada página está compuesta por los organismos y moléculas antes descritos, combinados según la plantilla. Los *elementos “héroe” visuales* en cada pantalla son aquellos de mayor importancia informativa: por ejemplo, en la Pantalla Principal son la lista de clientes y sus razones principales (lo primero que ve el usuario)[, en la Pantalla de Detalle son las recomendaciones de IA destacadas en tarjetas con botones de acción prominentes](file://file_00000000cb0461fb9d47c6cd19c32044#:~:text=,menús%20profundos%20y%20texto%20largo)[, y en la Pantalla de Registro de Visita es el resumen final con el botón “Finalizar visita” claramente visible.](file://file_00000000cb0461fb9d47c6cd19c32044#:~:text=,Mostrar%20al%20cliente)

## 2. Especificaciones Detalladas por Componente

A continuación se detallan componentes clave con sus propiedades, estilos y ejemplos de uso en JSX. Estos componentes se implementarán respetando los tamaños, colores y tipografías de los tokens definidos (sin desviaciones)[.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=match%20at%20L291%20333333%20y,Usa%20exactamente%20estos%20hex)

#### Componente: Botón Primario (<PrimaryButton>)

* **Descripción:** Botón de acción principal de la app. Se usa para confirmar o ejecutar la acción más importante en una pantalla (CTA), por ejemplo “Entrar”, “Guardar”, “Finalizar visita”.
* **Estilos:** Altura **48dp** (mínimo táctil)[, ancho variable hasta ~312dp max. Relleno horizontal estándar 16px. Radio de borde](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Dimensiones%3A%20hasta%20312%20×%2048,altura%20mínima%2048%20dp) **12dp** para esquinas redondeadas[. Color de fondo](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Dimensiones%3A%20hasta%20312%20×%2048,altura%20mínima%2048%20dp) **verde primario** (#0A7D2B) y texto en **blanco** (#FFFFFF)[. Tipografía del texto: 16px, peso 500 (Medium), todo en mayúsculas o Title Case según guía Material. Sombra/elevación nivel 1 (desplazamiento z=1dp) para estado normal.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Dimensiones%3A%20hasta%20312%20×%2048,altura%20mínima%2048%20dp)
* **Estados:** *Normal* (fondo verde, sombra leve), *Hover/Focus* (puede aclarar ligeramente el fondo o elevar sombra), *Pressed* (ligero oscurecimiento y desplazamiento -2dp de elevación[),](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Ubicación%3A%20dentro%20del%2030%20,thumb%20zone) *Disabled* (opacidad reducida al 38% [sobre color primario, quedando un verde grisáceo #0A7D2B60).](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Ubicación%3A%20dentro%20del%2030%20,thumb%20zone) *Loading*: opcionalmente puede mostrar un spinner a la izquierda del texto cuando está en estado “cargando”[.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Estados%3A%20normal%20%2F%20presionado%20,spinner%20alineado%20left%20%2B%20label)
* **Props:**

| Prop | Tipo | Descripción |
| --- | --- | --- |
| **label** | string | Texto a mostrar en el botón (corta acción, p.ej. "Guardar"). |
| **onClick** | function | Callback al hacer click/tap (acción a ejecutar). |
| **disabled** | bool | Si está true, se muestra deshabilitado (estilo opaco) e ignora eventos. |
| **loading** | bool | Si true, indica estado de carga: deshabilita click y muestra spinner de carga junto al texto. |

* **Uso (JSX):**

// Botón primario en estado normal  
<PrimaryButton label="Finalizar visita" onClick={finalizarVisita} />  
  
// Botón primario en estado de carga (deshabilitado automáticamente)  
<PrimaryButton label="Guardando..." loading={true} />

*(En el primer ejemplo, el botón verde llama a finalizarVisita cuando se pulsa. En el segundo ejemplo, el botón muestra un estado de carga con spinner y no permite interacción.)*

#### Componente: Campo de Texto (<TextField>)

* **Descripción:** Campo de entrada de texto de formulario, estilo Material Design. Incluye etiqueta, campo input y texto auxiliar/error.
* **Estilos:** Altura total **56dp**[. Label de 12px arriba del campo (cuando hay contenido) o como placeholder interno. Texto de entrada: 14px color texto primario (#333). Borde inferior de 1px color gris claro (#E0E0E0) por defecto](file://file_0000000043dc61fa9eefb470199a33df#:~:text=5)[; al enfocar cambia a borde](file://file_000000004bdc61f490771690e57e336d#:~:text=%22border%22%3A%20%7B%20%22default%22%3A%20%22,de%20inputs%2C%20tarjetas%2C%20elementos%20interactivos) **verde foco** (#0A7D2B) para indicar foco[. Icono opcional (ej. ícono de ojo para mostrar contraseña) de 24px puede colocarse a la derecha dentro del campo. Texto de ayuda: 12px color #666 debajo del campo; en caso de error se muestra en rojo #D8262C.](file://file_000000004bdc61f490771690e57e336d#:~:text=%22border%22%3A%20%7B%20%22default%22%3A%20%22,de%20inputs%2C%20tarjetas%2C%20elementos%20interactivos)
* **Estados:** *Enfocado* (borde/acento verde primario, label sube y se colorea verde), *Sin foco* (borde gris claro, label gris si contenido), *Relleno* (label pequeña arriba, texto presente), *Error* (borde rojo #D8262C, mensaje de error visible). *Deshabilitado* (texto y borde gris medio, opacidad reducida ~40%, sin interacciones).
* **Props:**

| Prop | Tipo | Descripción |
| --- | --- | --- |
| **label** | string | Etiqueta del campo (ej. "Contraseña"). |
| **placeholder** | string | Texto placeholder cuando está vacío (si no se usa label flotante). |
| **value** | string | Valor de texto controlado del campo. |
| **onChange** | function | Callback al cambiar texto (firma: func(nuevoValor) ). |
| **type** | string | Tipo de input (texto, email, password, etc.). |
| **error** | string | Mensaje de error a mostrar (cadena vacía si sin error). |
| **disabled** | bool | Si true, campo no editable. |

* **Uso (JSX):**

// Campo de texto para contraseña con estado de error  
<TextField   
 label="Contraseña"   
 type="password"   
 value={password}   
 onChange={setPassword}   
 error={ passwordValida ? "" : "Contraseña incorrecta" }   
/>

*(En este ejemplo, el campo muestra la etiqueta "Contraseña". Si la contraseña no es válida, aparece un mensaje de error en rojo debajo. El borde inferior se pondrá rojo cuando error no esté vacío.)*

#### Componente: Banner Offline (<OfflineBanner>)

* **Descripción:** Barra informativa fija que indica ausencia de conexión. Se muestra en la parte superior de la pantalla cuando la app está en modo offline.
* **Estilos:** Altura **48dp**, ocupa el ancho completo de la ventana[. Color de fondo](file://file_0000000043dc61fa9eefb470199a33df#:~:text=match%20at%20L265%20interacción,“Usar%20remoto”%2C%20detalles%20mínimos) **azul offline** (#2196F3) según token offline/info[. Texto centrado color blanco (#FFFFFF) en estilo Body/14px. Iconografía: puede incluir un pequeño ícono de nube con tache a la izquierda del texto para mayor claridad (24px, color blanco).](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Azul)
* **Contenido:** Mensaje breve indicando el estado offline, siguiendo la microcopy definida: por ejemplo: *“Sin conexión. Trabajando localmente.”*[. Este mensaje transmite que la app sigue operativa offline.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=match%20at%20L270%20Microcopy%20,¿Qué%20versión%20quieres)
* **Comportamiento:** Aparece automáticamente al perder conexión y desaparece al recuperar conexión (con una animación de transición suave en la aparición/desaparición). **No es interactivo** (no se cierra manualmente). Mientras se esté offline, permanece visible (no debe ocultarse)[.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=match%20at%20L317%20No%20ocultar,el%20banner%20offline%20cuando%20offline)
* **Props:** *Este componente no requiere props personalizadas*, solo se monta/desmonta según estado de red. Opcionalmente podría aceptar un texto personalizado si necesitáramos mostrar un mensaje diferente.
* **Uso (JSX):**

{/\* Mostrar banner si isOffline es true \*/}  
{ isOffline && <OfflineBanner /> }

*(En este fragmento, <OfflineBanner> se renderiza cuando la aplicación detecta modo offline. Al reconectar, isOffline sería false y el banner dejaría de mostrarse.)*

#### Componente: Ítem de Lista de Cliente (<ClienteListItem>)

* **Descripción:** Representa a un cliente en la pantalla principal (lista de visitas del día). Es un botón/grupo seleccionable que resume la info clave del cliente.
* **Estilos:** Contenedor de fondo blanco con borde redondeado 8dp y **sombra nivel 1** (elevación) para resaltarlo ligeramente del fondo[. Padding interno: 12px. Altura mínima ~72dp para contenido (siempre al menos un renglón de nombre y uno de detalle). Separa sus elementos en layout horizontal: icono + textos + flecha.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Dimensiones%3A%20hasta%20312%20×%2048,altura%20mínima%2048%20dp)
* **Icono/avatar:** Círculo de 40dp con fondo en color variante (por ejemplo, un color según inicial) o un ícono representativo. Letra inicial del cliente en 20px bold, color blanco.
* **Texto Nombre:** Nombre del cliente en 16px bold, color texto primario #333.
* **Texto Razón:** Descripción corta de la razón principal de visita en 14px regular, color texto secundario #666 (ej: "Razón: Reactivar compra XX Lager").
* **Prioridad Chip:** Si el cliente tiene una prioridad especial, se muestra un chip/etiqueta junto al nombre. Ej: un rectángulo con "Prioridad crítica" en texto blanco sobre fondo rojo #D8262C si es crítico.
* **Indicador navegación:** Icono de flecha (Material Icon “chevron\_right”) de 24px al extremo derecho indicando que al pulsar se entra al detalle.
* **Estados:** *Normal* (fondo blanco). *Hover/Pressed* en desktop o al tocar: se puede aplicar un leve fondo gris claro para feedback táctil. *Selected* no aplica, ya que la navegación es inmediata al tocar.
* **Props:**

| Prop | Tipo | Descripción |
| --- | --- | --- |
| **nombre** | string | Nombre del cliente. |
| **prioridad** | string | Etiqueta de prioridad (ej. "crítica"/"alta"/"" vacía si no hay). Determina si se muestra el chip de prioridad y su estilo (rojo para "crítica", amarillo para "alta", etc.). |
| **razon** | string | Razón principal de la visita (breve descripción). |
| **onClick** | function | Callback al hacer clic en el item (navegar a detalle). |

* **Uso (JSX):**

<ClienteListItem   
 nombre="Abarrotes La Paz"   
 prioridad="crítica"   
 razon="Reactivar compra de XX Lager"   
 onClick={() => abrirDetalle(clienteId)}   
/>

*(Este ejemplo renderiza un ítem de cliente llamado “Abarrotes La Paz” con prioridad crítica – se mostrará un chip rojo “Prioridad crítica” – y la razón de visita “Reactivar compra de XX Lager”. Al pulsarlo, llama a abrirDetalle pasando el ID del cliente.)*

#### Componente: Ítem de Recomendación (<RecomendacionItem>)

* **Descripción:** Muestra una recomendación de IA dentro de la pantalla de detalle del cliente. Incluye la acción sugerida y permite registrar el resultado de dicha recomendación.
* **Estilos:** Cada recomendación se presenta en una **tarjeta interna** dentro del detalle:
* Fondo blanco o ligeramente gris claro (para separarla del fondo general si corresponde).
* Margen inferior de 8dp entre recomendaciones.
* **Título de acción:** texto descriptivo corto, 14px bold, color #333. Ej: “Sugerir surtido XX Lager 355”.
* **Detalle/razón:** texto secundario con contexto, 14px regular #666 (puede incluir datos como “Cliente con 2 semanas sin comprar. Proponer 3 cajas.”).
* **Etiqueta de vigencia:** un pequeño chip de estado al lado del título o alineado a la derecha indicando cuándo aplica la recomendación (ej. “Hoy”, “Esta semana”), con color: verde #4CAF50 si es para hoy, amarillo #FFC107 si es esta semana, etc.
* **Botón registrar:** botón secundario alineado abajo (derecha o debajo del texto) con etiqueta "Registrar feedback"[. Es de estilo secundario: borde verde y texto verde #0A7D2B, fondo transparente (o fondo muy claro) para no destacar más que el botón principal de la pantalla.](file://file_00000000009061f993ee5e4e33fed8dc#:~:text=Módulos%20de%20Trazabilidad%20y%20Feedback%3A,podrá%20registrar%20de%20forma%20rápida)
* **Funcionalidad:** Al pulsar "Registrar feedback", abre el formulario de feedback correspondiente a esa recomendación (modal/página aparte). Mientras no se haya registrado feedback, la recomendación puede marcarse como *pendiente*. Tras registrar, podría cambiar el estado visual (por ejemplo, deshabilitar el botón o marcar como “Completada”).
* **Props:**

| Prop | Tipo | Descripción |
| --- | --- | --- |
| **titulo** | string | Título corto de la acción recomendada. |
| **descripcion** | string | Descripción o detalle explicativo de la recomendación. |
| **vigencia** | string | Texto de vigencia (ej. "Hoy", "Esta semana", "Próxima visita"). |
| **feedbackPendiente** | bool | Indica si el feedback de esta recomendación está pendiente. De ser true, el componente podría resaltar de alguna forma (por ej. icono 🕒). |
| **onRegistrar** | function | Handler al pulsar "Registrar feedback" (abre o navega al formulario de registro de feedback para esta recomendación). |

* **Uso (JSX):**

{recs.map(rec => (  
 <RecomendacionItem   
 key={rec.id}   
 titulo={rec.titulo\_accion}   
 descripcion={rec.descripcion\_detalle}   
 vigencia={rec.vigencia}   
 feedbackPendiente={!rec.feedbackEnviado}   
 onRegistrar={() => abrirFeedback(rec.id)}   
 />  
))}

*(En este fragmento, se itera sobre una lista de recomendaciones recs. Por cada una se crea un RecomendacionItem pasando el título de la acción, descripción, vigencia (por ejemplo, propiedad calculada), y se indica si el feedback está pendiente. Al hacer click en el botón "Registrar feedback", se llama a abrirFeedback con el ID de la recomendación.)*

## 3. Estrategia Offline-First (Modo Sin Conexión)

La aplicación está diseñada con una filosofía **Offline-First**, garantizando que el asesor pueda usarla sin conexión continua. A continuación se describen los mecanismos implementados:

**Service Worker & Caché de Recursos:** Se registra un *Service Worker* utilizando Workbox desde el primer acceso. Su tarea es interceptar solicitudes y almacenar en caché tanto la “shell” de la app (HTML, CSS, JS) como ciertos datos API, permitiendo que la PWA cargue y funcione offline[. El SW precachea los archivos estáticos durante la instalación (usando](file://file_00000000b21461f690ca97d7085c6574#:~:text=El%20Service%20Worker%20,para%20enviar%20datos) workbox-precaching) y luego atiende peticiones de navegación con esos recursos cacheados cuando no hay red[. Esto significa que si el asesor recarga la página estando offline, el SW servirá inmediatamente la interfaz desde cache en lugar de una página de error, proporcionando una experiencia continua](file://file_000000002cec61fba609c81630ec7a9d#:~:text=,No%20se%20perdió%20nada)[.](file://file_000000002cec61fba609c81630ec7a9d#:~:text=dentro%20de%20IndexedDB%2C%20marcado%20como,La%20página%20se)

**Almacenamiento Local Persistente:** Para no depender de la memoria volátil, la app utiliza **IndexedDB** como base de datos local en el navegador. Al realizar la **sincronización inicial** (cuando hay conexión al comienzo del día), la aplicación descarga el *Plan de Ruta Enriquecido* (lista de clientes y recomendaciones) y lo **almacena completo en IndexedDB**[. Este almacenamiento se realiza a través de una base de datos local (por ejemplo,](file://file_000000002cec61fba609c81630ec7a9d#:~:text=1.%20%20,de%20la%20aplicación) offlineDB) con tablas/colecciones como: - PlanDeRuta: guarda el objeto de ruta del día (clientes y sus recomendaciones). - FeedbackPendiente: cola de eventos de feedback/inteligencia pendientes de enviar[.](file://file_00000000b21461f690ca97d7085c6574#:~:text=La%20base%20de%20datos%20local,proceso%20de%20sincronización%20de%20fondo)

Cada vez que se obtiene un nuevo plan o datos actualizados del servidor, se **actualiza IndexedDB**. Así, si el usuario cierra o recarga la app, en el arranque la lógica de la app primero verifica si hay datos locales en IndexedDB para mostrar inmediatamente la información antes de intentar llamar a la red[.](file://file_000000002cec61fba609c81630ec7a9d#:~:text=,No%20se%20perdió%20nada) **Nada crítico se borra en un refresh**, logrando que al recargar offline, toda la información permanezca intacta en pantalla (lista de clientes, recomendaciones, etc.)[. Esto es fundamental:](file://file_000000002cec61fba609c81630ec7a9d#:~:text=3.%20%20,la%20ruta%20y%20el%20feedback) *“la app, al iniciarse, primero busca los datos en esta base local antes de intentar internet”*[.](file://file_000000002cec61fba609c81630ec7a9d#:~:text=dentro%20de%20IndexedDB%2C%20marcado%20como,La%20página%20se)

**Operación Offline y Cola de Sincronización:** Mientras el asesor trabaja *sin conexión*, todas las acciones que normalmente se envían al backend se **guardan en local** marcadas como pendientes. Por ejemplo, cada vez que se registra un feedback de recomendación o un formulario de inteligencia de mercado, la app: 1. Guarda el evento de feedback en FeedbackPendiente (IndexedDB) con un estado “pendiente”. 2. Refleja inmediatamente en la UI que el registro fue tomado – por ejemplo, podría marcar la recomendación como registrada con un icono de reloj/pendiente (sin esperar la respuesta del servidor). 3. Intenta enviar el feedback en segundo plano. Si no hay conexión, simplemente queda en la cola.

El Service Worker puede utilizar la API de **Background Sync** o técnicas de reconexión para procesar esta cola. Por ejemplo, aprovechamos **Workbox BackgroundSync**: cuando se hace un POST /feedback y falla por falta de red, el SW encola esa petición y la reintentará automáticamente cuando vuelva la conectividad[.](file://file_00000000b21461f690ca97d7085c6574#:~:text=el%20asesor%20solicita%20su%20ruta,reintentos)

Para implementar esto con Workbox se define una ruta de cacheo para dichas peticiones. Un fragmento ilustrativo de la configuración del SW con Workbox es:

// Cachear respuestas de API de manera offline-first  
registerRoute(  
 ({ url }) => url.pathname.startsWith('/api/'),  
 new NetworkFirst({ cacheName: 'api-cache' })  
);  
  
// Encolar automáticamente envíos de feedback al estar offline  
registerRoute(  
 ({ url }) => url.pathname === '/feedback',  
 new NetworkOnly({  
 plugins: [   
 new BackgroundSyncPlugin('feedbackQueue', { maxRetentionTime: 24 \* 60 /\* min \*/ })   
 ]  
 }),  
 'POST'  
);

*// En este ejemplo, cualquier POST a /feedback que falle por falta de conexión se guarda en una cola feedbackQueue (persistente en IndexedDB). Workbox reintentará enviarla hasta por 24 horas. La estrategia NetworkFirst en rutas /api asegura que, si disponible, se usen datos cacheados.*

Cuando la conexión se restablece, el Service Worker despierta y **reintenta sincronizar** los datos pendientes[. Los eventos en](file://file_00000000b21461f690ca97d7085c6574#:~:text=el%20asesor%20solicita%20su%20ruta,reintentos) FeedbackPendiente se envían al servidor (por la ruta API correspondiente) y, si el envío es exitoso, se: - Borra la entrada de la cola local. - Notifica a la interfaz. La UI escucha resultados de sync (p.ej. vía postMessage del SW o chequeando periódicamente) y **actualiza el estado visual**: por ejemplo, quita el icono de pendiente y podría mostrar un pequeño mensaje o cambiar el estado de la recomendación a “Completada”. El **indicador local (disco verde)** se ocultará si la cola quedó vacía, indicando que ya no hay cambios locales sin enviar[.](file://file_00000000b21461f690ca97d7085c6574#:~:text=almacenada%20en%20la%20sincronización%20anterior,reintentos)

**Manejo de Conflictos:** Dado que la app puede operar offline, existe la posibilidad de conflictos de datos cuando vuelve la conexión (por ejemplo, si el asesor modifica algo que el servidor también cambió). La estrategia es optimizar la experiencia para el asesor: - Siempre que sea posible, la última acción del usuario tiene prioridad. Si al sincronizar el servidor indica un conflicto (datos desactualizados), la app mostrará un **modal no destructivo** dando a elegir: *“Mantener datos locales”* vs *“Usar versión del servidor”*, con detalle mínimo del conflicto[. Por ejemplo:](file://file_0000000043dc61fa9eefb470199a33df#:~:text=interacción,“Usar%20remoto”%2C%20detalles%20mínimos) *“Detectamos cambios en el servidor. ¿Qué versión quieres conservar?”*[. Esto permite al usuario decidir de forma informada. - Mientras no se resuelva el conflicto, los datos locales permanecen (no se sobreescriben automáticamente). La decisión del usuario desencadenará la acción correspondiente: si escoge “mantener local”, se reenviará forzadamente el cambio; si escoge “usar remoto”, se descarta el cambio local pendiente. - Este modal de conflicto solo aparece en casos excepcionales. En la mayoría de interacciones (feedback de recomendaciones) es poco probable conflicto porque cada registro es independiente, pero el mecanismo está previsto para seguridad.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Microcopy%20%28tono%20claro%2C%20empático%29%20,¿Qué%20versión%20quieres%20conservar%3F”)

**Flujo Offline/Online Resumido:** *Diagrama de flujo conceptual de sincronización:*

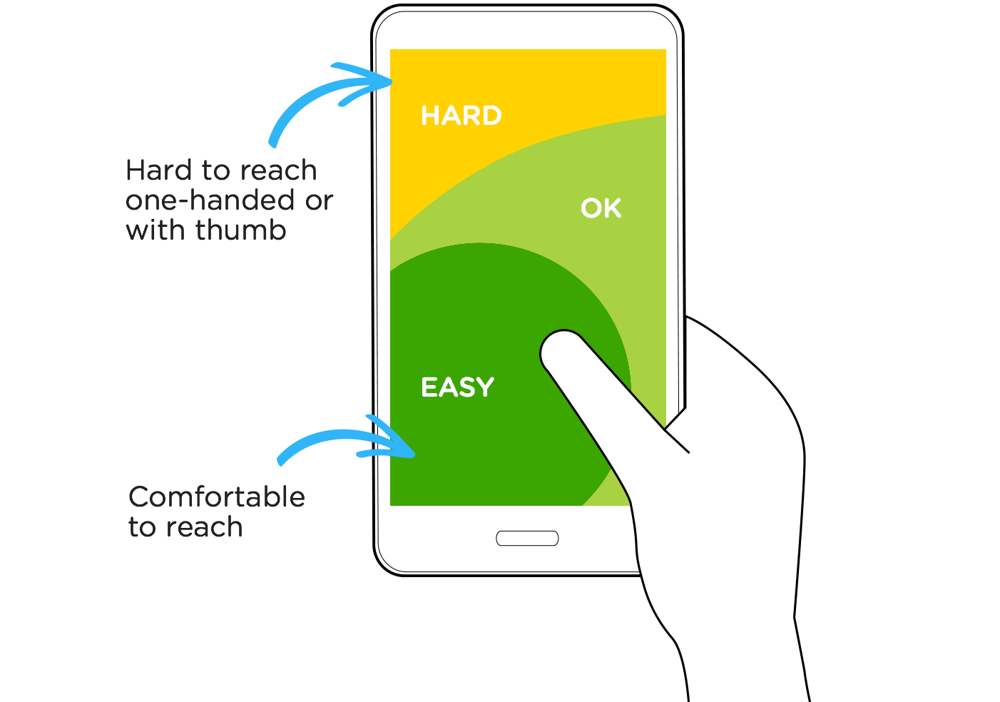
1. **Inicio del día (Online):** El asesor inicia sesión y la app realiza la sincronización inicial. -> [*La lista de visitas y recomendaciones se obtienen del backend y se almacenan en IndexedDB.*](file://file_00000000b21461f690ca97d7085c6574#:~:text=En%20cualquiera%20de%20los%20casos%2C,asesor%20la%20lista%20de%20clientes)
2. **Operación en campo (Offline):** El asesor navega por sus clientes y registra datos. -> *La app utiliza los datos locales del plan para mostrar la info (no requiere red)*[*. Cada feedback o formulario se guarda en la cola local (IndexedDB) en estado pendiente.*](file://file_00000000b21461f690ca97d7085c6574#:~:text=con%20el%20plan%20completo,asesor%20la%20lista%20de%20clientes)
3. **Usuario recarga/navega (Offline):** Si por error se cierra o recarga la PWA sin conexión, -> *El Service Worker sirve inmediatamente la aplicación desde caché, y al iniciar, la app carga el último plan y pendientes desde IndexedDB, restaurando el estado tal cual estaba*[.](file://file_000000002cec61fba609c81630ec7a9d#:~:text=dentro%20de%20IndexedDB%2C%20marcado%20como,La%20página%20se)
4. **Reconexion y sincronización:** Cuando el dispositivo recupera internet, -> *El Service Worker detecta el evento y automáticamente envía todos los registros pendientes al servidor*[. La UI muestra un estado de “Sincronizando…” con un icono animado (por ejemplo, un spinner junto al texto)](file://file_00000000b21461f690ca97d7085c6574#:~:text=el%20asesor%20solicita%20su%20ruta,reintentos)[.](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Estados%20previstos%201,“Sincronizando…”%29%20sin%20bloquear)
5. **Confirmación (Online):** Si todos los datos se sincronizan correctamente, -> *Se ocultan los indicadores de pendiente (disco verde)*[*, se retira el banner offline, y se puede mostrar una notificación sutil de éxito (ej. “Sincronización completa. ¡Buen trabajo hoy!”)*](file://file_0000000043dc61fa9eefb470199a33df#:~:text=interacción,“Usar%20remoto”%2C%20detalles%20mínimos). La pantalla final incluso resume cuántos registros se enviaron exitosamente【25†look】.
6. **Error en sincronización:** Si alguna petición falla incluso tras reconexión (por ej. servidor no disponible), -> *Los datos permanecen en la cola. El SW seguirá reintentando automáticamente o la app ofrecerá reintentar manualmente.* El asesor será notificado si corresponde (ej. mostrando “Intentando reenviar...” hasta lograrlo o botón “Reintentar” en alguna alerta).

Esta estrategia *offline-first* asegura que **el trabajo nunca se pierde** y el usuario puede confiar en la herramienta incluso con conectividad intermitente[. Para el desarrollador, el enfoque implica mayor complejidad, pero es](file://file_000000002cec61fba609c81630ec7a9d#:~:text=Desde%20la%20perspectiva%20de%20la,máxima%20prioridad%20en%20el%20desarrollo) *no negociable* dado el caso de uso en campo[.](file://file_000000002cec61fba609c81630ec7a9d#:~:text=desarrollo%20frontend%20implemente%20de%20forma,IndexedDB)

Además, ciertas consideraciones adicionales de resiliencia se implementan: - Si la sincronización inicial falla (sin internet al inicio): la app mostrará un mensaje de error y permitirá reintentar; no pudiendo proseguir sin datos iniciales (aunque se podría mostrar última ruta cacheada como referencia, esto es opcional porque las rutas cambian a diario)[. - Si durante el día no se puede obtener algún dato no crítico (ej. llamada a una API secundaria), la app lo omite y registra el fallo para manejarlo luego, evitando bloquear al usuario. Siempre se prioriza mantener la app usable. - Al finalizar el día, se envía un resumen final (puede ser en batch) y el app muestra un comprobante de sincronización final con estadísticas de envíos completados y pendientes, asegurando al asesor que todo quedó registrado.](file://file_00000000b21461f690ca97d7085c6574#:~:text=8,con%20el%20equipo%20IA%20tener)

## 4. Pautas de Ergonomía y Accesibilidad

En la implementación frontend se aplicarán estrictamente las pautas de ergonomía y accesibilidad definidas en los principios de UX:

* **Diseño “Thumb-Friendly”:** La interfaz está optimizada para uso con una sola mano y el pulgar. Los controles más usados y los botones primarios se sitúan en la zona de fácil alcance inferior/central de la pantalla[. Por ejemplo, el botón “Finalizar visita” en la pantalla de cierre se coloca en la parte inferior derecha (para diestros) o centrado, dentro de la zona verde de alcance fácil (ver imagen). Asimismo, se evita ubicar elementos interactivos importantes en esquinas superiores difíciles de alcanzar.  
  ](file://file_00000000cb0461fb9d47c6cd19c32044#:~:text=,clave%20en%20las%20esquinas%20superiores)  
  *Zonas de alcance con una mano en pantallas móviles: en verde la más cómoda, amarillo requiere estirar, rojo difícil【20†look】. La app sitúa los CTAs principales en la zona verde inferior para facilitar el tap.*
* **Contraste y Legibilidad:** Todos los textos y elementos cumplen con el contraste mínimo AA (ratio >= 4.5:1) sobre su fondo[. Se utilizan fondos muy claros (blancos) y texto oscuro (gris muy oscuro #333) para máxima legibilidad a la luz del día](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Reglas%20,5%3A1%20sobre%20superficies)[.](file://file_00000000cb0461fb9d47c6cd19c32044#:~:text=,ser%20de%20un%20tamaño%20generoso) *No* se usan grises claros sobre blanco para texto secundario, en su lugar se emplea gris #666 como mínimo[. Las fuentes tienen tamaño suficiente: 14px o más para cuerpos de texto, 16px o más para elementos importantes, siguiendo las recomendaciones Material (ej. títulos 20px)](file://file_000000004bdc61f490771690e57e336d#:~:text=%22text%22%3A%20%7B%20%22primary%22%3A%20%22,)[. Se verificará en cada pantalla estos valores antes de aprobar el diseño](file://file_0000000043dc61fa9eefb470199a33df#:~:text=Tamaños%20base%3A)[. Además, se evita bloquear texto con fondos brillantes; los estados (error, warning, info) usan colores accesibles (rojo, ámbar, azul) con texto blanco o negro según contraste.](file://file_00000000b21461f690ca97d7085c6574#:~:text=Estos%20lineamientos%20de%20UX%20son,y%20la%20eficiencia%20de%20uso)
* **Espaciado y Agrupación:** Aplicamos la ley de proximidad para la usabilidad[. Elementos relacionados aparecen agrupados con un padding interno consistente (8px/16px según el componente) y separados de otros grupos por margen suficiente. Por ejemplo, en la pantalla de detalle, todos los datos de una recomendación (título, descripción, botón) están contenidos en la misma tarjeta con un fondo común, separados de la siguiente recomendación por un espacio claro. Esto ayuda a](file://file_00000000cb0461fb9d47c6cd19c32044#:~:text=,siguiente%20recomendación%20por%20un%20espacio) *scanability* y evita toques erróneos: los botones tienen al menos 8px de separación entre sí. Ningún botón pequeño está demasiado cerca de otro para no presionar el incorrecto[.](file://file_00000000b21461f690ca97d7085c6574#:~:text=de%20validación%20,y%20la%20eficiencia%20de%20uso)
* **Feedback Táctil Inmediato:** Cada interacción de usuario produce una respuesta visual o háptica rápida (<100ms) para confirmar que la acción fue registrada. Por ejemplo, los botones usan efectos de estado activo: al tocarlos se **resalta** con un cambio de color ligero o animación ripple material (un breve destello de color)[. Esto se puede lograr con CSS (](file://file_00000000cb0461fb9d47c6cd19c32044#:~:text=,que%20cargue%20la%20siguiente%20pantalla):active) para cambiar la opacidad/fondo brevemente o utilizando la librería de componentes (Material Ripple). En dispositivos que lo soportan (ej. Android PWA instalada), podemos emplear la API de vibración para un sutil *buzz* al pulsar botones importantes[. Asimismo, acciones prolongadas (e.g. cargar recomendaciones de IA) muestran inmediatamente un spinner o indicador de progreso dentro del botón o sección, dando feedback que algo está en proceso.](file://file_00000000cb0461fb9d47c6cd19c32044#:~:text=,que%20cargue%20la%20siguiente%20pantalla)
* **Manejabilidad de Interrupciones (Offline First):** El diseño considera que el usuario puede ser interrumpido (llamadas, pérdida de señal) y retorna a la app. La UI siempre muestra el estado actual del sistema claramente. Ejemplos: si la IA está procesando algo, se muestra un mensaje “Generando recomendaciones…” con animación; si la geolocalización está en progreso, se indica “Obteniendo ubicación…”[. Cuando se pierde conexión, aparece el banner offline azul al instante](file://file_00000000cb0461fb9d47c6cd19c32044#:~:text=,Ubicación%20guardada%20correctamente)[. Al reconectar, se muestra un aviso breve de sincronización. Esto mantiene al usuario informado (principio de](file://file_0000000043dc61fa9eefb470199a33df#:~:text=match%20at%20L265%20interacción,“Usar%20remoto”%2C%20detalles%20mínimos) *Visibilidad del Estado*) en todo momento[.](file://file_00000000b21461f690ca97d7085c6574#:~:text=de%20forma%20clara%20,marcar%20una%20visita%20como%20completada)
* **Navegación y Enfoques Accesibles:** Aunque es una app mobile-first, aseguramos compatibilidad con accesibilidad estándar:
* Orden de tabulador lógico en caso de uso con teclado (por ejemplo, si se accede vía desktop o con teclado Bluetooth en tablet). Los componentes interactivos (button, input, etc.) siguen un flujo ordenado DOM y muestran un *focus outline* claro al enfocarse vía teclado (usando :focus-visible se podría personalizar el outline en verde, 2px, para visibilidad).
* Etiquetas ARIA y roles semánticos: los iconos puramente decorativos se marcan con aria-hidden="true", mientras que los iconos que transmiten estado (offline, éxito, etc.) tendrán aria-label descriptivo. Ej: el ícono de nube tachada llevará aria-label="Sin conexión" para lectores de pantalla.
* Textos alternativos: cualquier imagen relevante tiene atributo alt. En nuestro caso casi no hay imágenes, pero por ejemplo el avatar con la inicial del cliente podría tener alt="Avatar con letra A". Los botones y enlaces utilizan textos claros; se evita usar “clic aquí” genérico, en su lugar la acción se nombra directamente (“Registrar llegada”, “Guardar cambios”)[.](file://file_00000000fc1c61f8a7d406ffcb3dfb6e#:~:text=The%20primary%20objective%20of%20your,are%20guaranteed%20a%20smooth%20ride)
* **Tamaño táctil:** mínimo **48×48dp** en todos los controles. Esto está alineado con las pautas de Android/Material y evita problemas a usuarios con baja precisión motora. Incluso chips y toggles pequeños se diseñan con área aumentada si es necesario (padding extra transparente) para alcanzar ese mínimo.
* **Modo de color y iluminación:** Dado el uso en exteriores, se valida la apariencia bajo mucha luz (se podría probar en modo alto brillo). También se prevé un modo oscuro en el futuro, pero inicialmente la paleta clara se eligió por visibilidad a la luz del sol.
* **Rendimiento percibido (UX):** La app se comporta de forma ágil: las transiciones entre pantallas ocurren en menos de ~300ms para que se perciban inmediatas. Donde haya carga de datos, se utilizan *placeholders* o esqueleto antes de mostrar contenido o bien indicadores de progreso, de modo que el usuario siempre vea una reacción inmediata a sus acciones. Por ejemplo, al abrir el detalle de un cliente, si las recomendaciones de IA tardan un poco en formatearse, se puede mostrar un skeleton card durante unos instantes. Los tiempos de respuesta se mantienen dentro de los límites recomendados de percepción (respuesta inmediata <0.1s para feedback táctil, <1s para carga simple, con *spinner* a partir de 1s) para mantener la sensación de fluidez.

En resumen, **antes de liberar cualquier pantalla se verificará**: (a) contrastes de color (AA), (b) tamaños de fuente y controles, (c) que la jerarquía visual priorice lo esencial (carga cognitiva baja)[, (d) que no se pueda provocar errores graves (ej. se deshabilitan botones de doble envío, se confirman acciones destructivas), y (e) que la app siga las convenciones familiares para los usuarios (iconografía estándar, navegación consistente)](file://file_00000000cb0461fb9d47c6cd19c32044#:~:text=,menús%20profundos%20y%20texto%20largo)[. De esta forma la aplicación será intuitiva, robusta y cómoda de usar en el campo.](file://file_00000000cb0461fb9d47c6cd19c32044#:~:text=,verde%2C%20que%20siempre%20lo%20sea)

# 5. Descripción de pantallas del prototipo.

En esta sección se resumen las 11 pantallas aprobadas en prototipo. Cada pantalla detalla su héroe visual, componentes clave, estados/indicadores y acciones principales. Colores de referencia: Primario #0A7D2B, Error #D8262C, Advertencia #FFC107, Offline #2196F3, Texto #333/#666.

## 1) Login — “Bienvenido”

**Descripción:** Pantalla inicial con formulario (ID de asesor + contraseña) y CTA primario para ingresar.

**Héroe visual:** Botón primario verde “Entrar” ubicado en la zona inferior (thumb‑friendly).

**Componentes:**

* Top App Bar mínima con logotipo.
* Campos `<TextField>`: ID de asesor, Contraseña (con eye‑toggle).
* Checkbox “Recordarme”; enlace “¿Olvidaste tu contraseña?”.
* Etiqueta secundaria con “Última sincronización …”.

**Estados/indicadores:**

* Validaciones inline; foco visible; CTA deshabilita mientras valida.
* Sin banner; posible indicador de carga en el botón al enviar.

**Acciones:**

* Enviar credenciales → navegación a Preparación inicial (si aplica) o Home.

## 2) Login — “Credenciales inválidas”

**Descripción:** Misma estructura del login con mensajes de error al validar.

**Héroe visual:** Alerta de error roja (#D8262C) con mensaje claro y aria‑live.

**Componentes:**

* Alert/Toast de error; campos con borde en estado error.
* Toggle “Mantener sesión iniciada”.

**Estados/indicadores:**

* CTA deshabilitado durante verificación.
* Soporte para reintentos; conteo de intentos opcional.

**Acciones:**

* Corregir y reintentar; recuperación segura de contraseña.

## 3) Preparación inicial (Sincronización)

**Descripción:** Wizard compacto que lista datasets a precargar y su progreso.

**Héroe visual:** Checklist de “Datasets descargados (7)” con barra al 100 % y CTA “Continuar”.

**Componentes:**

* Lista de datasets con tamaño, timestamp y estado (icono/spinner).
* Switch “Listo para continuar”.
* CTA “Continuar” (primario).

**Estados/indicadores:**

* Banner \*\*Offline\*\* #2196F3 si no hay red; reintentos por ítem.
* Bloquea continuar si faltan obligatorios.

**Acciones:**

* Al terminar → seed de IndexedDB y navegación a Home.

## 4) Home — Lista de clientes (Hoy)

**Descripción:** Vista diaria con tabs/filters y tarjetas de cliente priorizadas.

**Héroe visual:** Lista priorizada de clientes + razón de visita (elemento héroe).

**Componentes:**

* Header con fecha (“Hoy • Vie 26 Sep”) y chips: Pendientes / Todos / Activos.
* Botones rápidos: Quiebre stock, Inventario propios, Inventario competencia; botón \*\*Sincronizar\*\*.
* Items de cliente: avatar/inicial, nombre (bold), razón, chip “Prioridad crítica” si aplica.

**Estados/indicadores:**

* Banner Offline arriba; disco verde #4CAF50 si hay pendientes locales.
* Empty/state y skeleton en carga.

**Acciones:**

* Tap en cliente → Detalle; accesos a módulos de inteligencia.

## 5) Detalle de cliente — Razón + Recomendaciones

**Descripción:** Ficha del punto de venta con contexto y acciones sugeridas por IA.

**Héroe visual:** Tarjetas de \*\*Recomendaciones\*\* (2–3) con botón “Registrar feedback”.

**Componentes:**

* Sección “Razón de visita” con chips de prioridad y objetivo.
* Cards de recomendación: título, detalle, vigencia; botón secundario outline.
* Botón \*\*Iniciar\*\* (registro de llegada) con geo/hora.

**Estados/indicadores:**

* Banner Offline; marcas de feedback pendiente; indicador de geo en progreso.

**Acciones:**

* Abrir formulario de feedback; iniciar/continuar visita.

## 6) Registrar feedback (modal/pantalla)

**Descripción:** Formulario corto para registrar el resultado de una recomendación.

**Héroe visual:** Selector de \*\*Resultado\*\* (Aceptada / Rechazada / Parcial) y campo \*\*Añadir motivo\*\*.

**Componentes:**

* Segmented control o radios para resultado.
* Textarea de notas con contador; presets de motivo (dropdown).
* Botones \*\*Cancelar\*\* (secundario) y \*\*Guardar\*\* (primario).

**Estados/indicadores:**

* Cuando offline: mensaje “Se guardará localmente”.
* Validaciones inline; loading en \*\*Guardar\*\*.

**Acciones:**

* Guardar → a cola local si no hay red; retorno a Detalle con estado actualizado.

## 7) Sincronización final (Resumen del día)

**Descripción:** Pantalla de confirmación y métricas de envío.

**Héroe visual:** Tarjeta de éxito “Sincronización completa. ¡Buen trabajo hoy!” + \*\*Resumen de envío\*\*.

**Componentes:**

* KPIs: enviados, fallidos, duración, última sync; contadores por módulo.
* CTA \*\*Terminar\*\*.

**Estados/indicadores:**

* Si quedan pendientes: aviso y CTA \*\*Reintentar\*\*; banner Offline si aplica.

**Acciones:**

* Cerrar jornada; mantener cola y reintentar automáticamente.

## 8) Quiebre de stock (propios)

**Descripción:** Captura de quiebres en SKUs propios con parámetros de captura.

**Héroe visual:** Lista de SKUs con estados y último movimiento; filtros de parámetros arriba.

**Componentes:**

* Barra de parámetros: horizonte (días), selección de motivo, selector de tienda si aplica.
* Lista de productos con switch/check, cantidad/unidad, notas opcionales.
* CTA \*\*Guardar\*\* y \*\*Cancelar\*\* (sticky bottom).

**Estados/indicadores:**

* Validación de campos mínimos; banners de error si faltan datos.
* Offline: copia “Tus cambios se guardarán localmente”.

**Acciones:**

* Guardar → evento a cola; feedback visual por ítem marcado.

## 9) Inventario (propios)

**Descripción:** Formulario de captura de inventario de productos propios.

**Héroe visual:** Tabla simple por SKU con campos de cantidad y PVP; CTA \*\*Guardar\*\* accesible.

**Componentes:**

* Buscador/filtrado por SKU o presentación.
* Filas con SKU, controles +/‑ para cantidad, input de PVP, badges de estado.
* Barra inferior con \*\*Cancelar\*\* y \*\*Guardar\*\*.

**Estados/indicadores:**

* Validaciones inline (rango de números); ayuda contextual.
* Offline: indicador de persistencia local (disco verde).

**Acciones:**

* Guardar → encolar cambios; sincronizar al reconectar.

## 10) Inventario (competencia)

**Descripción:** Captura de observaciones de competencia por nivel y PVP.

**Héroe visual:** Listado de marcas/sku competencia con niveles (alto/medio/bajo) y PVP observado.

**Componentes:**

* Tabs \*\*Todos\*\* / \*\*Top 10\*\*; lista con tarjetas por producto.
* Controles tipo chip para nivel; input PVP; notas opcionales.
* Botones \*\*Cancelar\*\* / \*\*Guardar\*\* (sticky).

**Estados/indicadores:**

* Validaciones y avisos si falta nivel/PVP; estados de carga.
* Offline-first con cola de sincronización.

**Acciones:**

* Guardar → persistencia local y posterior sincronización.

## 11) Finalizar visita

**Descripción:** Cierre de visita con validaciones y resumen de pendientes.

**Héroe visual:** Bloque \*\*Resumen de visita\*\* y botón primario \*\*Finalizar visita\*\* en zona inferior.

**Componentes:**

* Avisos en amarillo (#FFC107): fuera de geocerca (±20 m), pendientes de feedback.
* Detalle de tiempos y acciones atendidas.
* Sección \*\*Validación de ubicación\*\*; campo \*\*Notas\*\* (contador 0/250).
* Botones \*\*Cancelar\*\* y \*\*Finalizar visita\*\*.

**Estados/indicadores:**

* Offline: mantener aviso “Enviaremos pendientes automáticamente al reconectar”.
* Errores de ubicación con opción \*\*Reintentar\*\*; conflicto de datos manejado por modal si aplica.

**Acciones:**

* Finalizar → registra local y dispara sync en background.