



Actividad 1: Mapa de rutas y contenedores funcionales



Historia de usuario

COMO usuario que entra en la aplicación,
QUIERO entender qué pantallas existen y moverme entre ellas,
PARA acceder rápidamente al chat, a mis conversaciones, a la Pokédex y a los ajustes.



Tareas

- **Tarea 1: Definir el mapa de rutas (documentado en el repo).**
Rutas objetivo (nombres de vistas; sin código):
 - Inicio/Chat (vista principal),
 - Conversaciones (listado),
 - Conversación (detalle con historial),
 - Pokédex (demostrador POKEAPI existente),
 - Ajustes (preferencias del asistente/modelo).
- **Tarea 2: Contenedor visual común (layout funcional).**
Cabecera persistente + área de contenido. Sin diseño “bonito”, sólo estructura funcional y clara.
- **Tarea 3: Elementos de navegación.**
Enlaces/botones que disparen cambio de vista. Deben ser visibles y funcionar con teclado.



Estructura esperada

```
src/  
├── App.jsx  
├── main.jsx  
├── components/  
│   ├── Layout/  
│   │   ├── AppLayout.jsx      # cabecera + área de contenido  
│   ├── Navigation/  
│   │   ├── NavBar.jsx        # enlaces funcionales  
│   └── Views/                 # contenedores de “pantalla”  
│       ├── ChatView.jsx  
│       ├── ConversationsView.jsx  
│       ├── ConversationView.jsx  
│       ├── PokedexView.jsx  
│       └── SettingsView.jsx  
├── services/  
├── styles/  
│   └── layout.css             # estilos funcionales mínimos  
public/  
└── assets/
```

Pruebas funcionales

- ✓ Hacer clic en cada enlace cambia la vista correctamente.
- ✓ La cabecera se mantiene estable.
- ✓ **GIF**: recorrido por todas las vistas usando la propia navegación.



Actividad 2: Enrutado con parámetros, queries y estados

✖ Historia de usuario

COMO persona que comparte enlaces,
QUIERO que la URL refleje la vista actual y sus parámetros (id de conversación, filtros),
PARA abrir directamente contenido concreto.

🧠 Tareas

- **Tarea 1: Parámetros de ruta.**
“ConversationView” debe aceptar un `id` de conversación (p. ej., `?` o segmento de URL simulado).
- **Tarea 2: Query params funcionales (sin estética).**
“ConversationsView” interpreta `?q=` (búsqueda) y `?sort=` (orden) para listar conversaciones.
- **Tarea 3: Sincronía URL ⇌ Estado.**
Cambiar los controles de búsqueda/orden actualiza la URL; abrir la URL restaura el estado.

📁 Estructura esperada

```
src/  
├── components/  
│   └── Views/  
│       ├── ConversationsView.jsx # lee/aplica q y sort  
│       └── ConversationView.jsx  # lee/aplica id  
└── services/  
    └── url-state.md                # doc: qué params se usan y por qué
```

🔧 Pruebas funcionales

- ✓ Escribir/limpiar búsqueda actualiza la URL.
- ✓ Abrir una URL con `?q=` y `?sort=` muestra el listado filtrado/ordenado.
- ✓ Abrir directamente una conversación por su `id` funciona.
- ✓ **GIF:** filtrar con query param → recargar → se mantiene.

Actividad 3: Rutas “protegidas” en cliente (guards) y redirecciones

Historia de usuario

COMO usuario no autenticado en este dispositivo,
QUIERO ver sólo las vistas públicas,
PARA evitar acceder a secciones que requieren sesión local.

Tareas

- **Tarea 1: Política de acceso (documentada).**
Definir qué vistas son públicas (p. ej., Login si la usas) y cuáles requieren sesión local (Chat, Conversaciones, Pokédex, Ajustes).
- **Tarea 2: Guarda funcional (cliente).**
Si no hay sesión local simulada, navegar a vista “pública” (o mostrar mensaje funcional).
- **Tarea 3: Redirecciones predecibles.**
Tras “acceder”, llevar a Chat; tras “salir”, volver a la pantalla pública.

Estructura esperada

```
src/
├── components/
│   ├── Auth/
│   │   ├── LoginView.jsx      # si decidís usar login simulado
│   │   └── SessionIndicator.jsx # muestra estado de “sesión local”
│   └── Navigation/
│       └── Guards.md           # doc: reglas de acceso y redirecciones
├── services/
│   └── storage.js              # persistencia simple del “estado de sesión”
├── styles/
│   └── auth.css                # estilos funcionales mínimos
```

Pruebas funcionales

- ✓ Sin sesión: el intento de abrir Conversaciones/Chat te lleva a la vista pública.
- ✓ Con sesión: acceder a cualquier vista “protegida” funciona.
- ✓ Tras cerrar sesión: redirección a vista pública.
- ✓ **GIF:** intento sin sesión → bloqueo funcional → login simulado → acceso → logout.

Actividad 4: Gestión del historial, restauración de scroll y foco

Historia de usuario

COMO usuario que usa “atrás/adelante” del navegador,
QUIERO que la app respete el historial, el scroll y el foco,
PARA volver exactamente donde estaba al navegar.

Tareas

- **Tarea 1: Comportamiento del historial.**
Verificar que “Atrás/Adelante” recupera la vista y parámetros previos (sin recálculos sorpresa).
- **Tarea 2: Restauración de scroll.**
Volver al mismo punto de scroll en listados largos (ConversationsView).
- **Tarea 3: Foco funcional.**
Al entrar en Chat, el foco va al input; al cambiar de vista, el foco cae en un elemento significativo (accesibilidad funcional).

Estructura esperada

```
src/  
├── components/  
│   ├── Views/  
│   │   └── ConversationsView.jsx # lista larga para probar scroll  
│   └── Feedback/  
│       └── ScreenAnnouncer.jsx # opcional: anuncia cambios de vista  
└── styles/  
    └── layout.css
```

Pruebas funcionales

- ✓ Navegar entre vistas y usar Atrás/Adelante mantiene estado/params.
- ✓ El scroll se restaura al volver a listados.
- ✓ El foco cae en el campo esperado.
- ✓ **GIF:** navegar a listado → bajar scroll → ir a detalle → volver → scroll restaurado.

Actividad 5: Errores, 404 y estados de carga unificados

Historia de usuario

COMO usuario,
QUIERO que los errores y pantallas inexistentes se manejen de forma clara y consistente,
PARA entender qué ocurre sin inspeccionar la consola.

Tareas

- **Tarea 1: Vista 404 funcional.**
Cualquier URL no declarada muestra una vista 404 sencilla con enlace para volver a Inicio/Chat.
- **Tarea 2: Estados de carga homogéneos.**
“ConversationsView” y “PokedexView” muestran un estado de “Cargando...” mientras resuelven datos.
- **Tarea 3: Captura de error por vista.**
Ante fallo en datos (p. ej., POKEAPI), mostrar mensaje funcional en la propia vista sin romper navegación.

Estructura esperada

```
src/  
├── components/  
│   ├── Views/  
│   │   └── NotFoundView.jsx  
│   └── Feedback/  
│       ├── Loading.jsx  
│       └── ErrorBlock.jsx  
├── styles/  
│   ├── layout.css  
│   └── chatbot.css  
public/  
└── assets/
```

Pruebas funcionales

- ✓ Escribir una URL desconocida muestra 404.
- ✓ Estados de carga visibles durante peticiones reales.
- ✓ Error de POKEAPI produce un bloque de error sin romper la app.
- ✓ **GIF:** URL inexistente → 404 → volver a inicio; forzar un error de API y ver el manejo.



Actividad 6: Deep-linking de conversación y acciones de navegación programática



Historia de usuario

COMO usuario que comparte trabajo,
QUIERO abrir directamente una conversación concreta y que la aplicación lo respete,
PARA retomar un hilo sin pasos intermedios.



Tareas

- **Tarea 1: Abrir conversación por URL.**
Desde fuera (escribiendo la URL con el `id`), cargar el historial correspondiente.
- **Tarea 2: Acciones de navegación desde la UI.**
“Nueva conversación” navega y prepara el contenedor del chat; “Duplicar” crea nueva y navega; “Borrar” regresa de forma predecible (sin quedarse en vista huérfana).
- **Tarea 3: Estados vacíos.**
Si el `id` no existe, mostrar estado “no encontrada” con CTA para volver a Conversaciones.



Estructura esperada

```
src/  
├── components/  
│   ├── Chatbot/  
│   │   ├── ChatWindow.jsx  
│   │   ├── MessageList.jsx  
│   │   └── MessageInput.jsx  
│   └── Views/  
│       ├── ConversationsView.jsx  
│       └── ConversationView.jsx # gestiona id inexistente/duplicado  
└── services/  
    └── conversations.js      # contrato de lectura por id (doc)
```



Pruebas funcionales

- ✓ Abrir por URL una conversación válida funciona (historial visible).
- ✓ Nueva/Duplicar/Borrar navegan de forma correcta y predecible.
- ✓ Conversación inexistente → estado “no encontrada” + enlace de retorno.
- ✓ **GIF:** copiar URL → pegar en otra pestaña → conversación carga; borrar → volver a listado.

Criterios globales del bloque (sólo funcional, no estético)

- La **URL siempre refleja** la pantalla y su estado relevante (id, filtros).
- **Ningún flujo queda bloqueado**: siempre hay forma de volver a Inicio/Chat.
- El **historial del navegador** funciona (Atrás/Adelante) sin estados rotos.
- Los **estados de error/carga** son visibles y consistentes, sin depender de la consola.
- Todos los flujos tienen **GIF corto** mostrando la navegación.

Ejercicio 1	No se cumple con alguno de los criterios del formato de entrega o no se realizan las pruebas o no se realiza el gif que verifique el correcto funcionamiento del ejercicio. 0 puntos	Se cumple con lo establecido en los criterios del formato de entrega, se realizan las pruebas con éxito y se evidencia mediante un gif. Se encuentra todo documentado en el README.md 1 punto
Ejercicio 2	No se cumple con alguno de los criterios del formato de entrega o no se realizan las pruebas o no se realiza el gif que verifique el correcto funcionamiento del ejercicio. 0 puntos	Se cumple con lo establecido en los criterios del formato de entrega, se realizan las pruebas con éxito y se evidencia mediante un gif. Se encuentra todo documentado en el README.md 1 punto
Ejercicio 3	No se cumple con alguno de los criterios del formato de entrega o no se realizan las pruebas o no se realiza el gif que verifique el correcto funcionamiento del ejercicio. 0 puntos	Se cumple con lo establecido en los criterios del formato de entrega, se realizan las pruebas con éxito y se evidencia mediante un gif. Se encuentra todo documentado en el README.md 2 puntos
Ejercicio 4	No se cumple con alguno de los criterios del formato de entrega o no se realizan las pruebas o no se realiza el gif que verifique el correcto funcionamiento del ejercicio. 0 puntos	Se cumple con lo establecido en los criterios del formato de entrega, se realizan las pruebas con éxito y se evidencia mediante un gif. Se encuentra todo documentado en el README.md 2 puntos
Ejercicio 5	No se cumple con alguno de los criterios del formato de entrega o no se realizan las pruebas o no se realiza el gif que verifique el correcto funcionamiento del ejercicio. 0 puntos	Se cumple con lo establecido en los criterios del formato de entrega, se realizan las pruebas con éxito y se evidencia mediante un gif. Se encuentra todo documentado en el README.md 2 puntos
Ejercicio 6	No se cumple con alguno de los criterios del formato de entrega o no se realizan las pruebas o no se realiza el gif que verifique el correcto funcionamiento del ejercicio. 0 puntos	Se cumple con lo establecido en los criterios del formato de entrega, se realizan las pruebas con éxito y se evidencia mediante un gif. Se encuentra todo documentado en el README.md 2 puntos