

MindMapper - Fase 1: Core Implementation



Resumen

Se ha implementado exitosamente el **core completo** de MindMapper, una aplicación de mapas mentales local-first construida con Electron, React y TypeScript.

Funcionalidades Implementadas

1. Modelo de Datos TypeScript (`src/renderer/types/mindmap.ts`)

- Interface `MindMapNode` con todas las propiedades:
- `id`, `text`, `children`, `parentId`
- `style` (`backgroundColor`, `textColor`, `icon`, `status`)
- `collapsed`, `position`, `order`
- Interface `NodeStyle` con estilos personalizables
- Interface `MindMap` con estructura completa
- Tipos para acciones y estado del viewport
- Paleta de colores por defecto

2. Store Zustand Completo (`src/renderer/store/mindMapStore.ts`)

- Estado global con mapa actual, nodos, selección
- **Acciones CRUD completas:**
 - `createNode` - Crear nodo hijo o hermano
 - `deleteNode` - Eliminar nodo recursivamente
 - `updateNodeText` - Actualizar texto
 - `updateNodeStyle` - Actualizar estilos
 - `moveNode` - Mover nodo (drag & drop)
 - `toggleCollapse` - Colapsar/expandir subárboles
- **Historial completo de Undo/Redo**
 - Implementado con stack de estados
 - `undo()`, `redo()`, `canUndo()`, `canRedo()`
 - Gestión de viewport (zoom, pan)
 - Creación de mapas nuevos con nodo raíz

3. Layout Jerárquico con Dagre (`src/renderer/utils/layout.ts`)

- Cálculo automático de posiciones con algoritmo dagre
- Layout de árbol vertical (Top-to-Bottom)
- Manejo de nodos colapsados
- Cálculo de conexiones entre nodos
- Cálculo de bounds para fit-to-screen
- Generación de paths curvos para conexiones

4. Componente Canvas (`src/renderer/components/Canvas.tsx`)

- **Renderizado SVG** con nodos y conexiones
- **Nodos visuales** con:
 - Rectángulos redondeados con color personalizable
 - Iconos de Lucide React
 - Texto truncado si es muy largo
 - Indicador de estado (pendiente/en progreso/hecho)
 - Botón de colapsar/expandir para nodos con hijos
- **Conexiones** entre nodos padre-hijo con curvas bezier
- **Zoom interactivo** con rueda del ratón
- **Pan (desplazamiento)** arrastrando el fondo
- **Selección de nodos** con click
- **Edición inline** con doble click
- **Drag & Drop básico** para reorganizar nodos
- Indicador visual de drop target
- Botón “Fit to Screen” flotante

5. Componente NodeEditor (`src/renderer/components/NodeEditor.tsx`)

- **Panel lateral** con propiedades del nodo seleccionado
- **Información del nodo:** texto, número de hijos
- **Botones de acción:**
 - Crear hijo (Tab)
 - Crear hermano (Enter)
 - Eliminar nodo (Delete)
- **Selector de color** con paleta de 8 colores
- **Selector de iconos** con grid de 28 iconos comunes
- **Selector de estado** con 3 opciones:
 - Pendiente (gris)
 - En Progreso (amarillo)
 - Hecho (verde)
- **Guía de atajos de teclado** integrada

6. Componente Toolbar (`src/renderer/components/Toolbar.tsx`)

- **Botones principales:**
 - Nuevo Mapa
 - Undo/Redo con estado deshabilitado
 - Zoom In/Out
 - Reset viewport
 - Indicador de nivel de zoom (%)
- **Nombre del mapa** en el centro
- Botón de colapsar todo (preparado para futuro)

7. Drag and Drop Completo

- Arrastrar nodos manteniendo click
- Feedback visual durante arrastre

- Highlight del nodo objetivo
- Actualización automática de parentId al soltar
- Validación (no se puede arrastrar el nodo raíz)

8. Integración en App.tsx

- **Layout completo:**
 - Toolbar en la parte superior
 - Canvas en el centro
 - NodeEditor en panel lateral derecho
- **Atajos de teclado globales:**
 - Tab → Crear nodo hijo
 - Enter → Crear nodo hermano
 - Delete → Eliminar nodo
 - Ctrl+Z → Undo
 - Ctrl+Y → Redo
- **Mapa de ejemplo inicial** con estructura demo
- Estado de carga durante inicialización

9. Estilos CSS Completos

- **Tema oscuro profesional** (slate/blue)
- App.css - Layout principal responsive
- Canvas.css - Estilos para nodos, conexiones, animaciones
- NodeEditor.css - Panel lateral, pickers, botones
- Toolbar.css - Barra de herramientas
- index.css - Reset y estilos globales
- **Animaciones suaves:**
 - Transiciones hover
 - Animación de drop target (dash)
 - Transformaciones de escala
- **Scrollbar personalizado** para panel lateral
- **Responsive** (preparado para mobile)



Paleta de Colores

- Background principal: #0f172a (slate-900)
- Background secundario: #1e293b (slate-800)
- Bordes: #334155 (slate-700)
- Texto primario: #e2e8f0 (slate-200)
- Texto secundario: #94a3b8 (slate-400)
- Acento: #3b82f6 (blue-500)



Cómo Ejecutar

```
# Instalar dependencias (ya hecho)
npm install

# Modo desarrollo
npm run dev

# Build para producción
npm run build

# Empaquetar aplicación
npm run package
```



Estructura de Archivos

src/renderer/	
└── components/	
├── Canvas.tsx	# Renderizado visual del mapa
├── NodeEditor.tsx	# Panel de edición de nodos
└── Toolbar.tsx	# Barra de herramientas
└── store/	
└── mindMapStore.ts	# Estado global con Zustand
└── styles/	
├── App.css	
├── Canvas.css	
├── NodeEditor.css	
└── Toolbar.css	
└── index.css	
└── types/	
└── mindmap.ts	# Tipos del dominio
└── electron.d.ts	
└── utils/	
└── layout.ts	# Cálculo de layout con dagre
└── App.tsx	# Componente principal
└── main.tsx	# Entry point



Características Destacadas

1. **Layout Automático:** Dagre calcula posiciones óptimas automáticamente
2. **Undo/Redo Ilimitado:** Historial completo de cambios
3. **Drag & Drop Intuitivo:** Reorganiza el árbol visualmente
4. **Edición Inline:** Doble click para editar texto directamente
5. **Zoom y Pan:** Navegación fluida del canvas
6. **Colapsar Subárboles:** Oculta ramas para simplificar vista
7. **Estilos Personalizables:** Colores, iconos y estados
8. **Atajos de Teclado:** Flujo de trabajo rápido
9. **Tema Oscuro:** Profesional y moderno
10. **Responsive:** Funciona en diferentes tamaños de pantalla

Estado del Proyecto

Fase 1: COMPLETADA

Todas las funcionalidades core están implementadas y funcionando:

-  Modelo de datos TypeScript
-  Store Zustand con undo/redo
-  Layout jerárquico con dagre
-  Canvas interactivo (SVG)
-  Editor de nodos
-  Toolbar
-  Drag and drop
-  Zoom/pan/fit-to-screen
-  Colapsar/expandir
-  Atajos de teclado
-  Estilos completos

Próximos Pasos (Fase 2)

1. **Persistencia:** Guardar/cargar archivos JSON vía Electron IPC
2. **Export:** Exportar a PNG/SVG
3. **Búsqueda:** Buscar nodos por texto
4. **Temas:** Selector de temas claro/oscuro
5. **Más íconos:** Ampliar biblioteca de íconos
6. **Notas:** Añadir notas detalladas a nodos
7. **Links:** Enlaces entre nodos no jerárquicos

Testing

La aplicación compila sin errores de TypeScript:

```
npx tsc -p tsconfig.renderer.json --noEmit
#  Sin errores
```

Build de producción exitoso:

```
npm run build
#  Compilado correctamente
```

Commit Git

```
b9545e9 - Implementar core completo de MindMapper Fase 1
```

¡MindMapper Fase 1 está completa y lista para usar! 