**Stack Tecnológico Completo y Recomendaciones para tu ERP Local**

Basado en tus requerimientos, aquí está la lista completa de herramientas y consideraciones para evitar los problemas que mencionas (especialmente los DOM issues con Electron):

Stack Definitivo Recomendado

**Core (ya definido por ti)**

* **Frontend**: HTML5, CSS3, JavaScript ES6+
* **UI Framework**: Tailwind CSS
* **Base de Datos**: MongoDB (local)
* **Empaquetado**: Electron

**Adiciones Esenciales**

1. **Framework Frontend**:
   * **Preact** (versión ligera de React) - Evita problemas de DOM con Virtual DOM
   * **Por qué**: Manejo eficiente de actualizaciones de UI, componentes reutilizables
2. **Gestión de Estado**:
   * **Zustand** (o Jotai) - Más simple que Redux y perfecto para Electron
   * **Ventaja**: Centraliza el estado evitando redundancias
3. **Build System**:
   * **Vite** + **electron-vite-plugin** - Configuración mínima y HMR rápido
   * **Beneficio**: Desarrollo ágil con hot-reload para ambos procesos (main/renderer)
4. **Base de Datos Local**:
   * **MongoDB Embedded** (con **MongoDB Node Driver**) o **NeDB** (alternativa más ligera)
   * **Extra**: Implementa **pouchdb** como capa intermedia para futura sincronización
5. **Manejo de Errores**:
   * **Boom** (para errores HTTP-like incluso en local) + tu BodegaError personalizado
   * **Sentry** (versión autoalojada opcional para logs)

Estructura de Proyecto para Evitar Problemas de DOM

Copy

Download

/bodega-erp

├── .electron-vite/ # Configuración build

├── src/

│ ├── main/ # Código del main process (Electron)

│ ├── renderer/ # TU frontend (Preact + Tailwind)

│ │ ├── assets/ # CSS/Imágenes

│ │ ├── core/ # Lógica compartida

│ │ │ ├── errors/ # Manejo centralizado de errores

│ │ │ ├── hooks/ # Custom hooks reutilizables

│ │ │ └── utils/ # Utilidades globales

│ │ ├── modules/ # Tus módulos ERP

│ │ └── stores/ # Zustand stores

│ └── database/ # Conexión a MongoDB local

│ ├── adapters/ # Para cambiar de DB fácilmente

│ └── models/ # Esquemas de datos

├── electron/ # Config Electron específica

│ ├── preload.js # Comunicación segura entre procesos

│ └── main.js # Punto de entrada Electron

└── scripts/ # Scripts para build/backup

Solución Específica para Problemas de DOM en Electron

1. **Regla de Oro**:
   * **Nunca** usar document o window directamente en React/Preact
   * **Solución**: Usar hooks seguros:

*// hooks/useElectronSafeDOM.js*

import { useEffect, useState } from 'preact/hooks';

export function useElectronSafeDOM() {

const [isReady, setIsReady] = useState(false);

useEffect(() => {

const checkDOM = () => {

if (typeof window !== 'undefined' && window.document) {

setIsReady(true);

}

};

checkDOM();

window?.addEventListener('load', checkDOM);

return () => window?.removeEventListener('load', checkDOM);

}, []);

return isReady;

}

*// Uso en componentes:*

function SafeComponent() {

const domReady = useElectronSafeDOM();

if (!domReady) return <div>Cargando...</div>;

return (

*// Tu JSX seguro aquí*

);

}

Configuración Clave de electron-vite.config.js

import { defineConfig } from 'electron-vite';

export default defineConfig({

main: {

*// Config para main process*

},

preload: {

*// Config para preload scripts*

},

renderer: {

*// Config especial para evitar problemas DOM*

esbuild: {

jsxFactory: 'h',

jsxFragment: 'Fragment'

},

resolve: {

alias: {

'react': 'preact/compat',

'react-dom': 'preact/compat'

}

}

}

});

Módulo Base para Evitar Redundancias

Ejemplo de estructura modular escalable para el módulo de inventario:

Copy

Download

/modules/inventario/

├── components/ # Componentes específicos

├── hooks/ # Hooks del módulo (useInventario, useProductos)

├── services/ # Lógica de negocio

├── stores/ # Estado del módulo (Zustand)

├── test/ # Pruebas del módulo

├── types/ # Tipos TypeScript o JSDoc

└── utils/ # Utilidades específicas

Checklist para Comenzar

1. **Antes de Codificar**:
   * Configurar Vite + Electron template
   * Definir estructura de errores centralizada
   * Configurar alias de imports (@/modules/\*)
2. **Primeros Componentes**:
   * SafeElectronProvider (Wrapper para seguridad DOM)
   * ErrorBoundary global (para capturar crashes en UI)
   * DBConnectionTester (Verificar estado de MongoDB local)
3. **Empaquetado Final**:
   * Usar electron-builder con:

"build": {

"appId": "com.tubodega.erp",

"win": {

"target": "portable",

"icon": "build/icon.ico"

},

"extraFiles": [

{

"from": "database/local",

"to": "db-data",

"filter": ["\*\*/\*"]

}

]

}