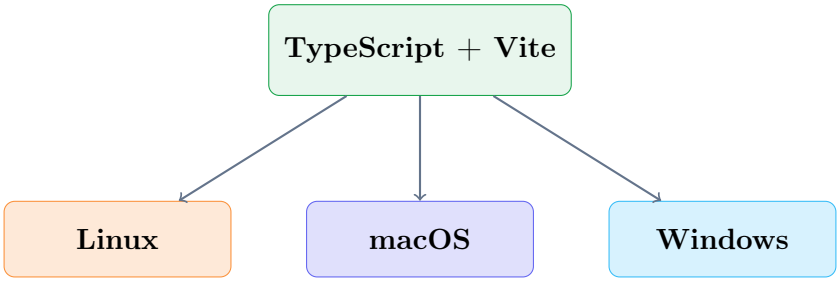
DesignLibre
Deployment Guide

Cross-Platform Desktop Application
with Tauri



Application: DesignLibre
Framework: Tauri 2.x
Frontend: TypeScript + Vite
Platforms: Linux, macOS, Windows

Version 1.0.0

January 2026

Contents

Part I
Architecture Overview

1 Introduction

DesignLibre is a 2D vector design application built with TypeScript and deployed as a native desktop application using Tauri. This document covers the complete deployment workflow from development to production releases across Linux, macOS, and Windows.

1.1 Why Tauri

Aspect	**Tauri**	**Electron**
Bundle Size	3–10 MB	150–300 MB
RAM Usage	30–80 MB	150–500 MB
WebView	System native	Bundled Chromium
Backend	Rust (compiled)	Node.js (interpreted)
Security	Process isolation	Full Node.js access
GPU Performance	Native WebView	Good, but heavier

Table 1: Tauri vs Electron Comparison

1.2 Platform WebView Backends

Platform	**WebView**	**Notes**
Linux	WebKitGTK	Based on WebKit, requires `libwebkit2gtk-4.1`
macOS	WKWebView	Native Safari/WebKit engine, pre-installed
Windows	WebView2	Chromium-based, pre-installed on Win 10/11

Table 2: Platform WebView Engines

1.3 Architecture Diagram

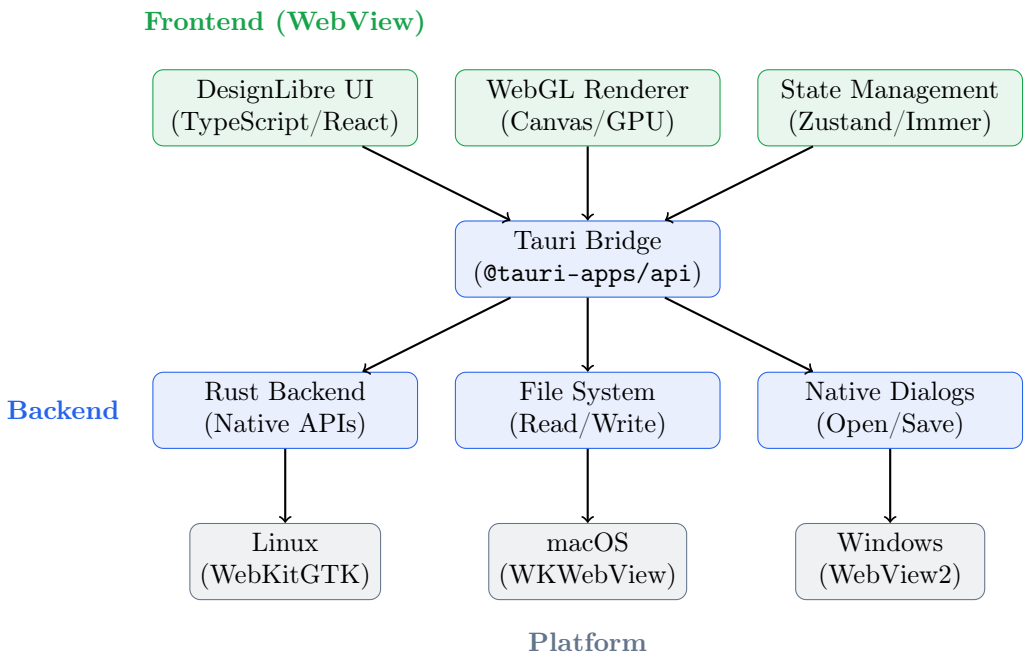


Figure 1: DesignLibre Architecture

2 Project Structure

```
designlibre/
          src/                              # TypeScript source
                main.tsx                      # Application entry
                App.tsx                       # Root component
                components/                   # UI components
                engine/                       # Rendering engine
                state/                        # State management
                utils/                        # Utilities
          public/                           # Static assets
                icons/                        # App icons
          src-tauri/                        # Tauri/Rust backend
                Cargo.toml                    # Rust dependencies
                tauri.conf.json               # Tauri configuration
                build.rs                      # Build script
                icons/                        # Platform icons
                      icon.ico                  # Windows
                      icon.icns                 # macOS
                      icon.png                  # Linux
                      32x32.png
                      128x128.png
                      128x128@2x.png
                src/
                    main.rs                   # Rust entry point
                    commands.rs               # Tauri commands
                    lib.rs                    # Library exports
          package.json                      # Node.js config
          tsconfig.json                     # TypeScript config
          vite.config.ts                    # Vite bundler config
```

```
            .github/
                  workflows/
                      release.yml            # CI/CD pipeline
            README.md
```

Listing 1: DesignLibre Project Structure

Part II
Development Environment

3 Prerequisites

3.1 All Platforms

```
# Node.js 20+ (LTS)
node --version   # v20.x.x or higher

# Rust toolchain
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
rustc --version  # 1.70.0 or higher

# Tauri CLI
cargo install tauri-cli
# Or via npm:
npm install -g @tauri-apps/cli
```

Listing 2: Universal Prerequisites

Linux:

Ubuntu/Debian Dependencies

```
sudo apt update
sudo apt install -y \
  libwebkit2gtk-4.1-dev \
  libappindicator3-dev \
  librsvg2-dev \
  patchelf \
  libssl-dev \
  libgtk-3-dev \
  libayatana-appindicator3-dev
```

Linux:

Fedora/RHEL Dependencies

```
sudo dnf install -y \
  webkit2gtk4.1-devel \
  openssl-devel \
  gtk3-devel \
```

```
    libappindicator-gtk3-devel \
    librsvg2-devel
```

Linux:

Arch Linux Dependencies

```
sudo pacman -S --needed \
  webkit2gtk-4.1 \
  base-devel \
  openssl \
  gtk3 \
  libappindicator-gtk3 \
  librsvg
```

macOS:

Xcode Command Line Tools

```
xcode-select --install

# For Apple Silicon Macs, also install Rosetta for x86 builds:
softwareupdate --install-rosetta
```

Windows:

Build Tools

```
# Install Visual Studio Build Tools 2022
# Download from: https://visualstudio.microsoft.com/downloads/

# Required workloads:
# - "Desktop development with C++"
# - Windows 10/11 SDK

# WebView2 Runtime (usually pre-installed)
# https://developer.microsoft.com/microsoft-edge/webview2/
```

4 Project Initialization

4.1 New Project Setup

```
# Create Vite project with TypeScript
npm create vite@latest designlibre -- --template react-ts
cd designlibre

# Install dependencies
npm install
```

```

# Initialize Tauri
npm install -D @tauri-apps/cli @tauri-apps/api
npx tauri init

# Answer prompts:
# - App name: DesignLibre
# - Window title: DesignLibre
# - Web assets path: ../dist
# - Dev server URL: http://localhost:5173
# - Dev command: npm run dev
# - Build command: npm run build
```

Listing 3: Initialize DesignLibre Project

4.2 Package.json Scripts

```
{
  "name": "designlibre",
  "version": "1.0.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "preview": "vite preview",
    "lint": "eslint src --ext ts,tsx",
    "typecheck": "tsc --noEmit",

    "tauri": "tauri",
    "tauri:dev": "tauri dev",
    "tauri:build": "tauri build",
    "tauri:build:debug": "tauri build --debug",

    "release:linux": "tauri build --target x86_64-unknown-linux-gnu",
    "release:mac-intel": "tauri build --target x86_64-apple-darwin",
    "release:mac-arm": "tauri build --target aarch64-apple-darwin",
    "release:windows": "tauri build --target x86_64-pc-windows-msvc"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "@tauri-apps/api": "^2.0.0"
  },
  "devDependencies": {
    "@tauri-apps/cli": "^2.0.0",
    "@types/react": "^18.2.0",
    "@types/react-dom": "^18.2.0",
    "typescript": "^5.3.0",
    "vite": "^5.0.0",
    "@vitejs/plugin-react": "^4.2.0"
  }
}
```

Listing 4: package.json Scripts Section

5 Tauri Configuration

5.1 Main Configuration

```
{
  "$schema": "https://schema.tauri.app/config/2.0.0",
  "productName": "DesignLibre",
  "version": "1.0.0",
  "identifier": "com.designlibre.app",
  "build": {
    "beforeDevCommand": "npm run dev",
    "devUrl": "http://localhost:5173",
    "beforeBuildCommand": "npm run build",
    "frontendDist": "../dist"
  },
  "app": {
    "windows": [
      {
        "title": "DesignLibre",
        "width": 1400,
        "height": 900,
        "minWidth": 800,
        "minHeight": 600,
        "resizable": true,
        "fullscreen": false,
        "center": true,
        "decorations": true,
        "transparent": false,
        "fileDropEnabled": true
      }
    ],
    "security": {
      "csp": "default-src 'self'; img-src 'self' data: blob:;
         script-src 'self' 'unsafe-eval'; style-src 'self'
         'unsafe-inline'"
    }
  },
  "bundle": {
    "active": true,
    "icon": [
      "icons/32x32.png",
      "icons/128x128.png",
      "icons/128x128@2x.png",
      "icons/icon.icns",
      "icons/icon.ico"
    ],
    "targets": "all",
    "linux": {
      "appimage": {
        "bundleMediaFramework": true
      },
      "deb": {
        "depends": ["libwebkit2gtk-4.1-0", "libgtk-3-0"],
        "section": "graphics",
        "priority": "optional"
      },
      "rpm": {
```

```
        "release": "1",
        "epoch": "0"
      }
    },
    "macOS": {
      "minimumSystemVersion": "10.15",
      "signingIdentity": null,
      "entitlements": null,
      "frameworks": []
    },
    "windows": {
      "certificateThumbprint": null,
      "digestAlgorithm": "sha256",
      "timestampUrl": "",
      "wix": null,
      "nsis": {
        "installMode": "currentUser",
        "languages": ["English"],
        "displayLanguageSelector": false
      }
    }
  },
  "plugins": {
    "fs": {
      "all": true,
      "scope": ["$HOME/**", "$DOCUMENT/**", "$DOWNLOAD/**"]
    },
    "dialog": {
      "all": true
    },
    "clipboard": {
      "all": true
    },
    "shell": {
      "open": true
    }
  }
}
```

Listing 5: src-tauri/tauri.conf.json

5.2 Rust Backend Entry Point

```
// Prevents additional console window on Windows in release
#![cfg_attr(not(debug_assertions), windows_subsystem = "windows")]

mod commands;

fn main() {
    tauri::Builder::default()
        .plugin(tauri_plugin_fs::init())
        .plugin(tauri_plugin_dialog::init())
        .plugin(tauri_plugin_clipboard_manager::init())
        .plugin(tauri_plugin_shell::init())
        .invoke_handler(tauri::generate_handler![
            commands::read_design_file,
```

```
            commands::write_design_file,
            commands::export_png,
            commands::export_svg,
            commands::get_system_fonts,
        ])
        .run(tauri::generate_context!())
        .expect("error while running DesignLibre");
}
```

Listing 6: src-tauri/src/main.rs

5.3 Custom Rust Commands

```
use std::fs;
use std::path::PathBuf;
use tauri::command;

#[derive(serde::Serialize)]
pub struct DesignFile {
    pub path: String,
    pub content: String,
}

#[command]
pub fn read_design_file(path: String) -> Result<DesignFile, String> {
    let content = fs::read_to_string(&path)
        .map_err(|e| format!("Failed to read file: {}", e))?;

    Ok(DesignFile { path, content })
}

#[command]
pub fn write_design_file(path: String, content: String) ->
    Result<(), String> {
    fs::write(&path, &content)
        .map_err(|e| format!("Failed to write file: {}", e))?;

    Ok(())
}

#[command]
pub async fn export_png(
    path: String,
    data: Vec<u8>,
    width: u32,
    height: u32,
) -> Result<(), String> {
    // PNG export implementation
    fs::write(&path, &data)
        .map_err(|e| format!("Failed to export PNG: {}", e))?;

    Ok(())
}

#[command]
```

```rust
pub async fn export_svg(path: String, content: String) -> Result<(), String> {
    fs::write(&path, &content)
        .map_err(|e| format!("Failed to export SVG: {}", e))?;

    Ok(())
}

#[command]
pub fn get_system_fonts() -> Result<Vec<String>, String> {
    // Platform-specific font enumeration
    let mut fonts = Vec::new();

    #[cfg(target_os = "linux")]
    {
        // Use fontconfig on Linux
        if let Ok(entries) = fs::read_dir("/usr/share/fonts") {
            for entry in entries.flatten() {
                if let Some(name) = entry.file_name().to_str() {
                    fonts.push(name.to_string());
                }
            }
        }
    }

    #[cfg(target_os = "macos")]
    {
        // macOS font directories
        let font_dirs = [
            "/System/Library/Fonts",
            "/Library/Fonts",
        ];
        for dir in font_dirs {
            if let Ok(entries) = fs::read_dir(dir) {
                for entry in entries.flatten() {
                    if let Some(name) = entry.file_name().to_str() {
                        fonts.push(name.to_string());
                    }
                }
            }
        }
    }

    #[cfg(target_os = "windows")]
    {
        // Windows fonts directory
        if let Ok(entries) = fs::read_dir("C:\\Windows\\Fonts") {
            for entry in entries.flatten() {
                if let Some(name) = entry.file_name().to_str() {
                    fonts.push(name.to_string());
                }
            }
        }
    }

    Ok(fonts)
}
```

Listing 7: src-tauri/src/commands.rs

5.4 Cargo Configuration

```
[package]
name = "designlibre"
version = "1.0.0"
description = "A 2D vector design application"
authors = ["DesignLibre Team"]
license = "MIT"
repository = "https://github.com/designlibre/designlibre"
edition = "2021"

[build-dependencies]
tauri-build = { version = "2.0", features = [] }

[dependencies]
tauri = { version = "2.0", features = ["macos-private-api"] }
tauri-plugin-fs = "2.0"
tauri-plugin-dialog = "2.0"
tauri-plugin-clipboard-manager = "2.0"
tauri-plugin-shell = "2.0"
serde = { version = "1.0", features = ["derive"] }
serde_json = "1.0"

[features]
default = ["custom-protocol"]
custom-protocol = ["tauri/custom-protocol"]

[profile.release]
panic = "abort"
codegen-units = 1
lto = true
opt-level = "z"
strip = true
```

Listing 8: src-tauri/Cargo.toml

6 TypeScript Integration

6.1 Tauri API Usage

```
import { invoke } from '@tauri-apps/api/core';
import { open, save } from '@tauri-apps/plugin-dialog';
import { readTextFile, writeTextFile } from '@tauri-apps/plugin-fs';
import { writeText, readText } from
    '@tauri-apps/plugin-clipboard-manager';

// File types for dialogs
const DESIGN_FILE_FILTERS = [
  { name: 'DesignLibre Files', extensions: ['dlb', 'dlbx'] },
  { name: 'SVG Files', extensions: ['svg'] },
```

```
  { name: 'All Files', extensions: ['*'] },
];

// Open file dialog
export async function openDesignFile(): Promise<string | null> {
  const selected = await open({
    multiple: false,
    filters: DESIGN_FILE_FILTERS,
  });

  if (selected && typeof selected === 'string') {
    return selected;
  }
  return null;
}

// Save file dialog
export async function saveDesignFile(defaultName?: string):
    Promise<string | null> {
  const selected = await save({
    defaultPath: defaultName,
    filters: DESIGN_FILE_FILTERS,
  });

  return selected;
}

// Read design file via Rust command
export async function readDesignFile(path: string):
    Promise<DesignFile> {
  return invoke<DesignFile>('read_design_file', { path });
}

// Write design file via Rust command
export async function writeDesignFile(path: string, content:
    string): Promise<void> {
  return invoke('write_design_file', { path, content });
}

// Export to PNG
export async function exportPng(
  path: string,
  data: Uint8Array,
  width: number,
  height: number
): Promise<void> {
  return invoke('export_png', {
    path,
    data: Array.from(data),
    width,
    height
  });
}

// Export to SVG
export async function exportSvg(path: string, content: string):
    Promise<void> {
  return invoke('export_svg', { path, content });
```

```
}

// Get system fonts
export async function getSystemFonts(): Promise<string[]> {
  return invoke<string[]>('get_system_fonts');
}

// Clipboard operations
export async function copyToClipboard(text: string): Promise<void> {
  return writeText(text);
}

export async function pasteFromClipboard(): Promise<string> {
  return readText();
}

// Types
interface DesignFile {
  path: string;
  content: string;
}
```

Listing 9: src/lib/tauri.ts

6.2 App Menu Integration

```
import { useEffect } from 'react';
import { getCurrentWindow } from '@tauri-apps/api/window';
import { Menu, MenuItem, Submenu } from '@tauri-apps/api/menu';

export function useAppMenu() {
  useEffect(() => {
    async function setupMenu() {
      const fileSubmenu = await Submenu.new({
        text: 'File',
        items: [
          await MenuItem.new({
            text: 'New',
            accelerator: 'CmdOrCtrl+N',
            action: () => console.log('New file'),
          }),
          await MenuItem.new({
            text: 'Open...',
            accelerator: 'CmdOrCtrl+O',
            action: () => handleOpen(),
          }),
          await MenuItem.new({
            text: 'Save',
            accelerator: 'CmdOrCtrl+S',
            action: () => handleSave(),
          }),
          await MenuItem.new({
            text: 'Save As...',
            accelerator: 'CmdOrCtrl+Shift+S',
            action: () => handleSaveAs(),
          }),
```

```
            { item: 'Separator' },
            await MenuItem.new({
              text: 'Export...',
              accelerator: 'CmdOrCtrl+E',
              action: () => handleExport(),
            }),
            { item: 'Separator' },
            await MenuItem.new({
              text: 'Exit',
              accelerator: 'CmdOrCtrl+Q',
              action: () => getCurrentWindow().close(),
            }),
          ],
        });

        const editSubmenu = await Submenu.new({
          text: 'Edit',
          items: [
            await MenuItem.new({
              text: 'Undo',
              accelerator: 'CmdOrCtrl+Z',
              action: () => handleUndo(),
            }),
            await MenuItem.new({
              text: 'Redo',
              accelerator: 'CmdOrCtrl+Shift+Z',
              action: () => handleRedo(),
            }),
            { item: 'Separator' },
            await MenuItem.new({
              text: 'Cut',
              accelerator: 'CmdOrCtrl+X',
              action: () => handleCut(),
            }),
            await MenuItem.new({
              text: 'Copy',
              accelerator: 'CmdOrCtrl+C',
              action: () => handleCopy(),
            }),
            await MenuItem.new({
              text: 'Paste',
              accelerator: 'CmdOrCtrl+V',
              action: () => handlePaste(),
            }),
          ],
        });

        const menu = await Menu.new({
          items: [fileSubmenu, editSubmenu],
        });

        await menu.setAsAppMenu();
      }

      setupMenu();
    }, []);
}
```

Listing 10: src/components/AppMenu.tsx

Part III
Development Workflow

7 Daily Development

7.1 Development Mode

Hot Module Replacement

During development, you work exactly like a web application. The native shell connects to your Vite dev server and provides hot module replacement.

```
# Start development (opens native window with HMR)
npm run tauri:dev
```

This runs two processes:

1. **Vite dev server** on `localhost:5173` serving TypeScript
2. **Native window** loading from the dev server

Edit TypeScript/CSS and see changes instantly. The native shell does NOT rebuild.

Figure 2: Development Mode Architecture

7.2 When Does the Native Shell Rebuild?

Change	Rebuild?	Action
Edit TypeScript/CSS	No	HMR instant reload
Add new React component	No	HMR instant reload
Change Vite config	No	Restart Vite
Change `tauri.conf.json`	Yes	Restart `tauri dev`
Edit Rust code (`*.rs`)	Yes	Auto-rebuilds
Add new Rust command	Yes	Auto-rebuilds
Change app icon	Yes	Run `tauri build`

Table 3: Rebuild Triggers

7.3 Debugging

```
# Open DevTools in development
# Press F12 or right-click -> Inspect

# Enable Rust debug logging
RUST_LOG=debug npm run tauri:dev

# Build with debug symbols
npm run tauri:build:debug

# Run debug build directly
./src-tauri/target/debug/designlibre
```

Listing 11: Debugging Commands

7.4 Testing Native Features

```
// In browser console or React component:
import { open } from '@tauri-apps/plugin-dialog';

const testDialog = async () => {
  const file = await open({
    multiple: false,
    filters: [{ name: 'Images', extensions: ['png', 'jpg'] }],
  });
  console.log('Selected:', file);
};

testDialog();
```

Listing 12: Testing File Dialog

8 Build Process

8.1 Build Pipeline

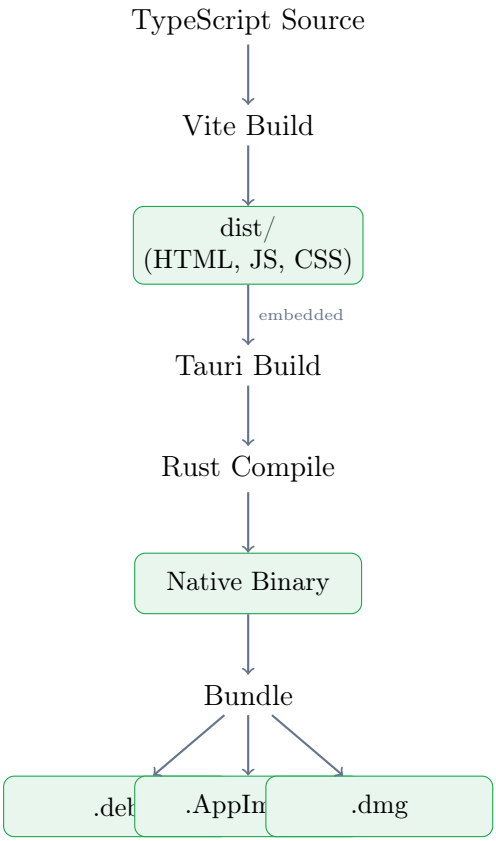


Figure 3: Production Build Pipeline

8.2 Build Commands

```
# Development build (faster, larger, with debug info)
npm run tauri:build:debug

# Production build (optimized, smaller)
npm run tauri:build

# Platform-specific builds
npm run release:linux       # x86_64 Linux
npm run release:mac-intel   # x86_64 macOS
npm run release:mac-arm     # aarch64 macOS (Apple Silicon)
npm run release:windows     # x86_64 Windows

# Build specific targets only
npx tauri build --target x86_64-unknown-linux-gnu
npx tauri build --bundles deb,appimage   # Linux only
npx tauri build --bundles dmg,app        # macOS only
npx tauri build --bundles msi,nsis       # Windows only
```

Listing 13: Build Commands

8.3 Build Output

```
src-tauri/target/release/
          designlibre                        # Linux/macOS binary
          designlibre.exe                    # Windows binary
          bundle/
              deb/
                    designlibre_1.0.0_amd64.deb
              appimage/
                    designlibre_1.0.0_amd64.AppImage
              rpm/
                    designlibre-1.0.0-1.x86_64.rpm
              dmg/
                    DesignLibre_1.0.0_x64.dmg
              macos/
                    DesignLibre.app/
              msi/
                    DesignLibre_1.0.0_x64_en-US.msi
              nsis/
                  DesignLibre_1.0.0_x64-setup.exe
```

Listing 14: Build Output Structure

Part IV
Platform-Specific Configuration

9 Linux

9.1 Package Formats

Format	Target	Notes
`.deb`	Debian, Ubuntu, Mint	Uses `dpkg`, integrates with apt
`.rpm`	Fedora, RHEL, openSUSE	Uses `rpm`, integrates with dnf/yum
`.AppImage`	Universal	Self-contained, no installation needed

Table 4: Linux Package Formats

Linux:

.deb Configuration

```
// In tauri.conf.json -> bundle -> linux -> deb
{
  "depends": [
    "libwebkit2gtk-4.1-0 (>= 2.38)",
    "libgtk-3-0 (>= 3.24)",
    "libayatana-appindicator3-1"
  ],
  "section": "graphics",
  "priority": "optional",
  "files": {
    "/usr/share/applications/designlibre.desktop":
       "assets/designlibre.desktop"
```

```
    }
  }
```

Linux:

Desktop Entry File

```
[Desktop Entry]
Name=DesignLibre
Comment=2D Vector Design Application
Exec=designlibre %F
Icon=designlibre
Terminal=false
Type=Application
Categories=Graphics;VectorGraphics;GTK;
MimeType=application/x-designlibre;image/svg+xml;
StartupNotify=true
StartupWMClass=designlibre
```

Listing 15: assets/designlibre.desktop

9.2 AppImage Distribution

```
# Make executable and run
chmod +x DesignLibre_1.0.0_amd64.AppImage
./DesignLibre_1.0.0_amd64.AppImage

# Optional: Integrate with desktop
# Uses AppImageLauncher or manual .desktop file
```

Listing 16: AppImage Usage

10 macOS

10.1 Code Signing

macOS:

Signing Requirements For distribution outside the Mac App Store, you need:

- Apple Developer account ($99/year)
- Developer ID Application certificate
- Notarization with Apple's servers

Without signing, users see "App is damaged" or Gatekeeper warnings.

```
# List available signing identities
security find-identity -v -p codesigning
```

```
3
4  # Set in environment
5  export APPLE_SIGNING_IDENTITY="Developer ID Application: Your Name
       (TEAMID)"
6  export APPLE_ID="your@email.com"
7  export APPLE_PASSWORD="app-specific-password"
8  export APPLE_TEAM_ID="YOURTEAMID"
9
10 # Build with signing
11 npm run tauri:build
12
13 # Or configure in tauri.conf.json:
14 # bundle.macOS.signingIdentity = "Developer ID Application: ..."
```

Listing 17: macOS Signing Setup

10.2 Notarization

```
1  # Notarization happens automatically during build if credentials set
2  # Manual notarization:
3  xcrun notarytool submit DesignLibre.dmg \
4    --apple-id "$APPLE_ID" \
5    --password "$APPLE_PASSWORD" \
6    --team-id "$APPLE_TEAM_ID" \
7    --wait
8
9  # Staple the notarization ticket
10 xcrun stapler staple DesignLibre.dmg
```

Listing 18: Notarization Process

10.3 Universal Binary (Intel + Apple Silicon)

```
1  # Install both targets
2  rustup target add x86_64-apple-darwin
3  rustup target add aarch64-apple-darwin
4
5  # Build universal binary
6  npm run tauri:build -- --target universal-apple-darwin
7
8  # Or build separately and combine:
9  npm run release:mac-intel
10 npm run release:mac-arm
11
12 # Combine with lipo
13 lipo -create \
14   target/x86_64-apple-darwin/release/designlibre \
15   target/aarch64-apple-darwin/release/designlibre \
16   -output target/universal/designlibre
```

Listing 19: Universal Binary Build

10.4 macOS Configuration

```
// In tauri.conf.json -> bundle -> macOS
{
  "minimumSystemVersion": "10.15",
  "frameworks": [],
  "entitlements": "./entitlements.plist",
  "exceptionDomain": null,
  "signingIdentity": null,
  "providerShortName": null
}
```

Listing 20: macOS Bundle Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
  "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>com.apple.security.app-sandbox</key>
  <false/>
  <key>com.apple.security.files.user-selected.read-write</key>
  <true/>
  <key>com.apple.security.network.client</key>
  <true/>
</dict>
</plist>
```

Listing 21: entitlements.plist

11 Windows

11.1 Installer Formats

Format	Builder	Notes
`.msi`	WiX	Enterprise-friendly, Group Policy support
`.exe`	NSIS	User-friendly installer wizard
`.exe` (portable)	None	Standalone executable

Table 5: Windows Package Formats

11.2 NSIS Installer Configuration

```
// In tauri.conf.json -> bundle -> windows -> nsis
{
  "license": "../LICENSE",
  "installerIcon": "icons/icon.ico",
  "sidebarImage": "icons/sidebar.bmp",
  "headerImage": "icons/header.bmp",
  "installMode": "currentUser",
  "languages": ["English", "German", "French", "Spanish"],
  "displayLanguageSelector": true,
```

```
  "shortcuts": {
    "desktop": true,
    "startMenu": true
  }
}
```

Listing 22: NSIS Configuration

11.3 Code Signing (Windows)

Windows:

Signing Options

- **EV Certificate**: Extended Validation, immediate SmartScreen trust
- **OV Certificate**: Organization Validation, builds trust over time
- **Self-signed**: Development only, triggers SmartScreen warnings

```
# Using SignTool (Windows SDK)
signtool sign /tr http://timestamp.digicert.com /td sha256 \
  /fd sha256 /a DesignLibre_1.0.0_x64-setup.exe

# Configure in tauri.conf.json:
# bundle.windows.certificateThumbprint = "YOUR_CERT_THUMBPRINT"
# bundle.windows.timestampUrl = "http://timestamp.digicert.com"
```

Listing 23: Windows Signing

11.4 WebView2 Runtime

```
// In tauri.conf.json -> bundle -> windows
{
  "webviewInstallMode": {
    "type": "downloadBootstrapper"
  }
}

// Options:
// "skip" - Assume WebView2 is installed
// "downloadBootstrapper" - Download installer if missing
    (recommended)
// "embedBootstrapper" - Embed installer in app (larger size)
// "offlineInstaller" - Embed full runtime (much larger)
```

Listing 24: WebView2 Configuration

Part V
CI/CD Pipeline

12 GitHub Actions

12.1 Complete Release Workflow

```yaml
name: Release

on:
  push:
    tags:
      - 'v*'

env:
  CARGO_INCREMENTAL: 0
  RUST_BACKTRACE: short

jobs:
  create-release:
    runs-on: ubuntu-latest
    outputs:
      release_id: ${{ steps.create-release.outputs.result }}
    steps:
      - uses: actions/checkout@v4

      - name: Create Release
        id: create-release
        uses: actions/github-script@v7
        with:
          script: |
            const { data } = await github.rest.repos.createRelease({
              owner: context.repo.owner,
              repo: context.repo.repo,
              tag_name: context.ref.replace('refs/tags/', ''),
              name: `DesignLibre ${context.ref.replace('refs/tags/',
                  '')}`,
              body: 'See CHANGELOG.md for release notes.',
              draft: true,
              prerelease: false
            });
            return data.id;

  build-linux:
    needs: create-release
    runs-on: ubuntu-22.04
    steps:
      - uses: actions/checkout@v4

      - name: Install dependencies
        run: |
          sudo apt-get update
          sudo apt-get install -y \
            libwebkit2gtk-4.1-dev \
            libappindicator3-dev \
            librsvg2-dev \
            patchelf \
            libssl-dev

      - name: Setup Node
```

```
      uses: actions/setup-node@v4
      with:
        node-version: 20
        cache: 'npm'

    - name: Setup Rust
      uses: dtolnay/rust-toolchain@stable

    - name: Rust cache
      uses: Swatinem/rust-cache@v2
      with:
        workspaces: src-tauri

    - name: Install npm dependencies
      run: npm ci

    - name: Build
      run: npm run tauri:build

    - name: Upload artifacts
      uses: actions/upload-artifact@v4
      with:
        name: linux-artifacts
        path: |
          src-tauri/target/release/bundle/deb/*.deb
          src-tauri/target/release/bundle/appimage/*.AppImage
          src-tauri/target/release/bundle/rpm/*.rpm

    - name: Upload to release
      uses: actions/github-script@v7
      env:
        RELEASE_ID: ${{ needs.create-release.outputs.release_id }}
      with:
        script: |
          const fs = require('fs');
          const path = require('path');

          const artifactPaths = [
            'src-tauri/target/release/bundle/deb/',
            'src-tauri/target/release/bundle/appimage/',
            'src-tauri/target/release/bundle/rpm/'
          ];

          for (const artifactPath of artifactPaths) {
            if (!fs.existsSync(artifactPath)) continue;

            const files = fs.readdirSync(artifactPath);
            for (const file of files) {
              const filePath = path.join(artifactPath, file);
              const stats = fs.statSync(filePath);

              if (stats.isFile()) {
                console.log(`Uploading ${file}...`);
                await github.rest.repos.uploadReleaseAsset({
                  owner: context.repo.owner,
                  repo: context.repo.repo,
                  release_id: process.env.RELEASE_ID,
                  name: file,
```

```
                    data: fs.readFileSync(filePath)
                  });
                }
              }
            }

  build-macos:
    needs: create-release
    runs-on: macos-latest
    strategy:
      matrix:
        target: [x86_64-apple-darwin, aarch64-apple-darwin]
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node
        uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: 'npm'

      - name: Setup Rust
        uses: dtolnay/rust-toolchain@stable
        with:
          targets: ${{ matrix.target }}

      - name: Rust cache
        uses: Swatinem/rust-cache@v2
        with:
          workspaces: src-tauri

      - name: Install npm dependencies
        run: npm ci

      - name: Build
        env:
          # Add signing credentials for production
          # APPLE_SIGNING_IDENTITY: ${{
              secrets.APPLE_SIGNING_IDENTITY }}
          # APPLE_ID: ${{ secrets.APPLE_ID }}
          # APPLE_PASSWORD: ${{ secrets.APPLE_PASSWORD }}
          # APPLE_TEAM_ID: ${{ secrets.APPLE_TEAM_ID }}
          TAURI_SIGNING_PRIVATE_KEY: ${{
              secrets.TAURI_SIGNING_PRIVATE_KEY }}
        run: npm run tauri:build -- --target ${{ matrix.target }}

      - name: Upload artifacts
        uses: actions/upload-artifact@v4
        with:
          name: macos-${{ matrix.target }}-artifacts
          path: |
            src-tauri/target/${{ matrix.target
                }}/release/bundle/dmg/*.dmg
            src-tauri/target/${{ matrix.target
                }}/release/bundle/macos/*.app

      - name: Upload to release
        uses: actions/github-script@v7
```

```
        env:
          RELEASE_ID: ${{ needs.create-release.outputs.release_id }}
          TARGET: ${{ matrix.target }}
        with:
          script: |
            const fs = require('fs');
            const path = require('path');

            const dmgPath =
               `src-tauri/target/${process.env.TARGET}/release/bundle/dmg/`;

            if (fs.existsSync(dmgPath)) {
              const files = fs.readdirSync(dmgPath);
              for (const file of files) {
                if (file.endsWith('.dmg')) {
                  const filePath = path.join(dmgPath, file);
                  // Add target to filename for clarity
                  const arch =
                     process.env.TARGET.includes('aarch64') ?
                     'arm64' : 'x64';
                  const newName = file.replace('.dmg',
                     `_${arch}.dmg`);

                  console.log(`Uploading ${newName}...`);
                  await github.rest.repos.uploadReleaseAsset({
                    owner: context.repo.owner,
                    repo: context.repo.repo,
                    release_id: process.env.RELEASE_ID,
                    name: newName,
                    data: fs.readFileSync(filePath)
                  });
                }
              }
            }

  build-windows:
    needs: create-release
    runs-on: windows-latest
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node
        uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: 'npm'

      - name: Setup Rust
        uses: dtolnay/rust-toolchain@stable

      - name: Rust cache
        uses: Swatinem/rust-cache@v2
        with:
          workspaces: src-tauri

      - name: Install npm dependencies
        run: npm ci

```

```yaml
      - name: Build
        env:
          TAURI_SIGNING_PRIVATE_KEY: ${{
              secrets.TAURI_SIGNING_PRIVATE_KEY }}
        run: npm run tauri:build

      - name: Upload artifacts
        uses: actions/upload-artifact@v4
        with:
          name: windows-artifacts
          path: |
            src-tauri/target/release/bundle/msi/*.msi
            src-tauri/target/release/bundle/nsis/*.exe

      - name: Upload to release
        uses: actions/github-script@v7
        env:
          RELEASE_ID: ${{ needs.create-release.outputs.release_id }}
        with:
          script: |
            const fs = require('fs');
            const path = require('path');

            const artifactPaths = [
              'src-tauri/target/release/bundle/msi/',
              'src-tauri/target/release/bundle/nsis/'
            ];

            for (const artifactPath of artifactPaths) {
              if (!fs.existsSync(artifactPath)) continue;

              const files = fs.readdirSync(artifactPath);
              for (const file of files) {
                const filePath = path.join(artifactPath, file);
                const stats = fs.statSync(filePath);

                if (stats.isFile()) {
                  console.log(`Uploading ${file}...`);
                  await github.rest.repos.uploadReleaseAsset({
                    owner: context.repo.owner,
                    repo: context.repo.repo,
                    release_id: process.env.RELEASE_ID,
                    name: file,
                    data: fs.readFileSync(filePath)
                  });
                }
              }
            }

  publish-release:
    needs: [create-release, build-linux, build-macos, build-windows]
    runs-on: ubuntu-latest
    steps:
      - name: Publish release
        uses: actions/github-script@v7
        env:
          RELEASE_ID: ${{ needs.create-release.outputs.release_id }}
        with:
```

```
            script: |
              await github.rest.repos.updateRelease({
                owner: context.repo.owner,
                repo: context.repo.repo,
                release_id: process.env.RELEASE_ID,
                draft: false
              });
```

Listing 25: .github/workflows/release.yml

12.2 Development CI

```yaml
name: CI

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-22.04
    steps:
      - uses: actions/checkout@v4

      - name: Install system dependencies
        run: |
          sudo apt-get update
          sudo apt-get install -y libwebkit2gtk-4.1-dev
              libappindicator3-dev

      - name: Setup Node
        uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: 'npm'

      - name: Setup Rust
        uses: dtolnay/rust-toolchain@stable

      - name: Install dependencies
        run: npm ci

      - name: Lint
        run: npm run lint

      - name: Type check
        run: npm run typecheck

      - name: Test
        run: npm test

      - name: Rust check
        run: cd src-tauri && cargo check

```

```
      - name: Rust clippy
        run: cd src-tauri && cargo clippy -- -D warnings

      - name: Build (debug)
        run: npm run tauri:build:debug
```

Listing 26: .github/workflows/ci.yml

Part VI
Release Process

13 Versioning

13.1 Semantic Versioning

DesignLibre follows Semantic Versioning (SemVer):

- **MAJOR** (1.x.x): Breaking changes, incompatible file format changes
- **MINOR** (x.1.x): New features, backward compatible
- **PATCH** (x.x.1): Bug fixes, backward compatible

13.2 Version Locations

Version must be updated in multiple files:

```
# 1. package.json
"version": "1.2.0"

# 2. src-tauri/tauri.conf.json
"version": "1.2.0"

# 3. src-tauri/Cargo.toml
version = "1.2.0"

# 4. CHANGELOG.md
## [1.2.0] - 2026-01-15
### Added
- New feature X
### Fixed
- Bug Y
```

Listing 27: Version Update Checklist

13.3 Version Bump Script

```
#!/bin/bash
set -e

VERSION=$1

```

```
if [ -z "$VERSION" ]; then
  echo "Usage: ./scripts/bump-version.sh <version>"
  exit 1
fi

echo "Bumping version to $VERSION..."

# Update package.json
npm version $VERSION --no-git-tag-version

# Update tauri.conf.json
sed -i "s/\"version\": \".*\"/\"version\": \"$VERSION\"/"
    src-tauri/tauri.conf.json

# Update Cargo.toml
sed -i "s/^version = \".*\"/version = \"$VERSION\"/"
    src-tauri/Cargo.toml

# Update Cargo.lock
cd src-tauri && cargo update -p designlibre && cd ..

echo "Version bumped to $VERSION"
echo "Don't forget to update CHANGELOG.md!"
```

Listing 28: scripts/bump-version.sh

14 Release Checklist

```
# 1. Ensure main branch is up to date
git checkout main
git pull origin main

# 2. Run full test suite
npm test
npm run lint
npm run typecheck
cd src-tauri && cargo test && cargo clippy && cd ..

# 3. Update version
./scripts/bump-version.sh 1.2.0

# 4. Update CHANGELOG.md
# Add release notes under ## [1.2.0] - YYYY-MM-DD

# 5. Commit version bump
git add -A
git commit -m "chore: bump version to 1.2.0"

# 6. Create and push tag
git tag -a v1.2.0 -m "Release v1.2.0"
git push origin main --tags

# 7. GitHub Actions automatically:
#    - Builds for all platforms
#    - Creates draft release
```

```
#     - Uploads artifacts
#     - Publishes release

# 8. Verify release on GitHub
# 9. Announce release
```

Listing 29: Release Process

15 Release Artifacts

Platform	Artifact	Description
3*Linux	`designlibre_1.2.0_amd64.deb`	Debian/Ubuntu package
	`designlibre_1.2.0_amd64.AppImage`	Universal Linux binary
	`designlibre-1.2.0-1.x86_64.rpm`	Fedora/RHEL package
2*macOS	`DesignLibre_1.2.0_x64.dmg`	Intel Mac installer
	`DesignLibre_1.2.0_arm64.dmg`	Apple Silicon installer
2*Windows	`DesignLibre_1.2.0_x64-setup.exe`	NSIS installer
	`DesignLibre_1.2.0_x64_en-US.msi`	MSI installer

Table 6: Release Artifacts per Platform

Part VII
Troubleshooting

16 Common Issues

Linux:

WebKitGTK Not Found

```
# Error: package 'webkit2gtk-4.1' not found

# Solution: Install WebKitGTK development package
sudo apt install libwebkit2gtk-4.1-dev   # Debian/Ubuntu
sudo dnf install webkit2gtk4.1-devel      # Fedora
sudo pacman -S webkit2gtk-4.1             # Arch
```

macOS:

Code Signing Failed

```
# Error: The application is damaged

# Solution 1: Clear quarantine attribute
xattr -cr /Applications/DesignLibre.app

# Solution 2: Allow in System Preferences
# Security & Privacy -> General -> Allow anyway
```

```
# Solution 3: Proper signing (for distribution)
# Set up Developer ID certificate and notarization
```

Windows:

WebView2 Missing

```
# Error: WebView2 runtime not found

# Solution 1: Install WebView2 runtime
# Download from:
    https://developer.microsoft.com/microsoft-edge/webview2/

# Solution 2: Configure auto-download in tauri.conf.json
# bundle.windows.webviewInstallMode.type =
    "downloadBootstrapper"
```

Build Fails on CI

```
# Common causes:

# 1. Missing system dependencies
# Add installation step for platform-specific deps

# 2. Rust cache corrupted
# Clear Swatinem/rust-cache or disable temporarily

# 3. Node modules mismatch
# Ensure npm ci (not npm install) for reproducible builds

# 4. Disk space
# GitHub runners have limited space, clean before build:
# - run: rm -rf ~/.cargo/registry/cache
```

17 Performance Optimization

```
[profile.release]
panic = "abort"         # Smaller binary, no unwinding
codegen-units = 1       # Better optimization, slower build
lto = true              # Link-time optimization
opt-level = "z"         # Optimize for size
strip = true            # Strip debug symbols

# For faster builds during development:
[profile.dev]
opt-level = 0
debug = true

```

```
[profile.dev.package."*"]
opt-level = 3          # Optimize dependencies
```

Listing 30: Optimized Cargo.toml Profile

18 Debugging Tips

```
# Enable verbose Tauri logging
RUST_LOG=debug npm run tauri:dev

# Enable WebView console in production
# Add to main.rs:
# .setup(|app| {
#     #[cfg(debug_assertions)]
#     app.get_window("main").unwrap().open_devtools();
#     Ok(())
# })

# Analyze bundle size
cd src-tauri
cargo bloat --release --crates

# Check for unused dependencies
cargo +nightly udeps

# Profile startup time
npm run tauri:build
time ./src-tauri/target/release/designlibre
```

Listing 31: Debugging Commands

19 Summary

Deployment Summary

Phase	Commands
Development	`npm run tauri:dev`
Test Build	`npm run tauri:build:debug`
Release Build	`npm run tauri:build`
Create Release	`git tag v1.2.0 && git push -tags`

Key Points:

- Daily development uses HMR — no native rebuild needed
- Native shell rebuilds only for config/Rust changes or releases
- CI builds all platforms automatically on tag push
- Version must be updated in 3 files before release
- Each platform has specific signing/notarization requirements

---

DesignLibre: Professional design tools, freely available.
