

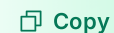
Tauri Minimal Project Structure

Basic Tauri + Vite setup. Simple frontend, minimal Rust backend. Perfect for getting started.

#tauri #rust #desktop #vite #minimal

 PNG

 PDF

 Copy

 Prompt

Project Directory

my-tauri-app/

- package.json Frontend depend...
- vite.config.js
- index.html Entry HTML
- src/ Frontend code
 - main.js App entry point
 - App.vue Root component
 - style.css
 - components/
 - Greet.vue
 - assets/
 - logo.svg
- src-tauri/ Rust backend
 - Cargo.toml Rust dependenci...
 - tauri.conf.json Tauri config
 - build.rs
 - src/
 - main.rs App bootstrap
 - lib.rs Exports and set...
 - icons/ App icons
 - icon.png
 - icon.ico
 - capabilities/ Permission sets
 - default.json
 - public/ Static assets

Why This Structure?

Tauri v2 splits your app into two parts: a web frontend (any framework) and a Rust backend in `src-tauri/`. This minimal structure uses Vite for fast HMR. The Rust side handles native APIs while staying lightweight—Tauri apps are typically 600KB-3MB.

Key Directories

- `src/` - Frontend—Vue, React, Svelte, or vanilla JS
- `src-tauri/` - Rust backend and Tauri configuration
- `src-tauri/tauri.conf.json` - Window settings, permissions, build config
- `src-tauri/capabilities/` - Permission sets for API access
- `src-tauri/icons/` - Platform-specific app icons

Getting Started

- `npm create tauri-app@latest`
- Choose your frontend framework
- `cd my-tauri-app && npm install`
- `npm run tauri dev`

When To Use This

- Building your first Tauri app
- Simple desktop utilities
- Web app wrappers with native features
- Learning Tauri fundamentals

When To Upgrade

- Need complex Rust business logic
- Multiple windows with different contexts
- Heavy IPC between frontend and backend
- Custom system tray or menu bar apps

Naming Conventions

Commands - snake_case in Rust: `greet_user`, camelCase in JS: `greetUser`

Events - kebab-case: `file-dropped`, `window-resized`

Capabilities - Descriptive names: `default`, `main-window`