



TAURI

Prérequis

Si vous avez docker :

 <https://sourceforge.net/projects/vcxsrv/>

 Lancer **VcXsrv** avant de démarrer l'app via Docker

Si vous n'avez pas Docker :

 Installez :

- Visual Studio C++ Build Tools
- Rustup

 <https://www.rust-lang.org/tools/install>

Pourquoi Electron a été créé ? ⚙️

- Besoin de créer des apps **desktop multiplateformes** avec des compétences web
- Utilise :
 - **Node.js** pour le backend
 - **Chromium** pour le rendu UI
- Objectif : développement rapide avec JavaScript

Exemples célèbres : VS Code, Discord, Slack

Qu'est-ce que Tauri ?

- Framework pour apps desktop **ultra-légères**
- Frontend : HTML/CSS/JS (React, Vue, etc.)
- Backend : écrit en **Rust**
- Utilise le **WebView natif** du système, pas Chromium

Pourquoi Tauri plutôt qu'Electron ? ⚖️

Critère	Electron	Tauri
Poids app	100–200 Mo	3–10 Mo
Performance	Gourmand	Léger, rapide
Sécurité	Moyenne	Élevée
Backend	Node.js (JS)	Rust
WebView	Chromium	Natif
Maturité	Élevée	Jeune, prometteur

Spécificités de Tauri ✨

- Très **léger**
- Écrit en **Rust**
- Sécurité renforcée (isolation, permissions)
- Modularité (importe que ce qui est utile)
- Intégration native (menu, notifs, raccourcis)
- Communication **Rust** ↔ **JS**

En résumé

Electron

- ✓ Mature et stable
- ✓ Facile pour devs JS

Tauri

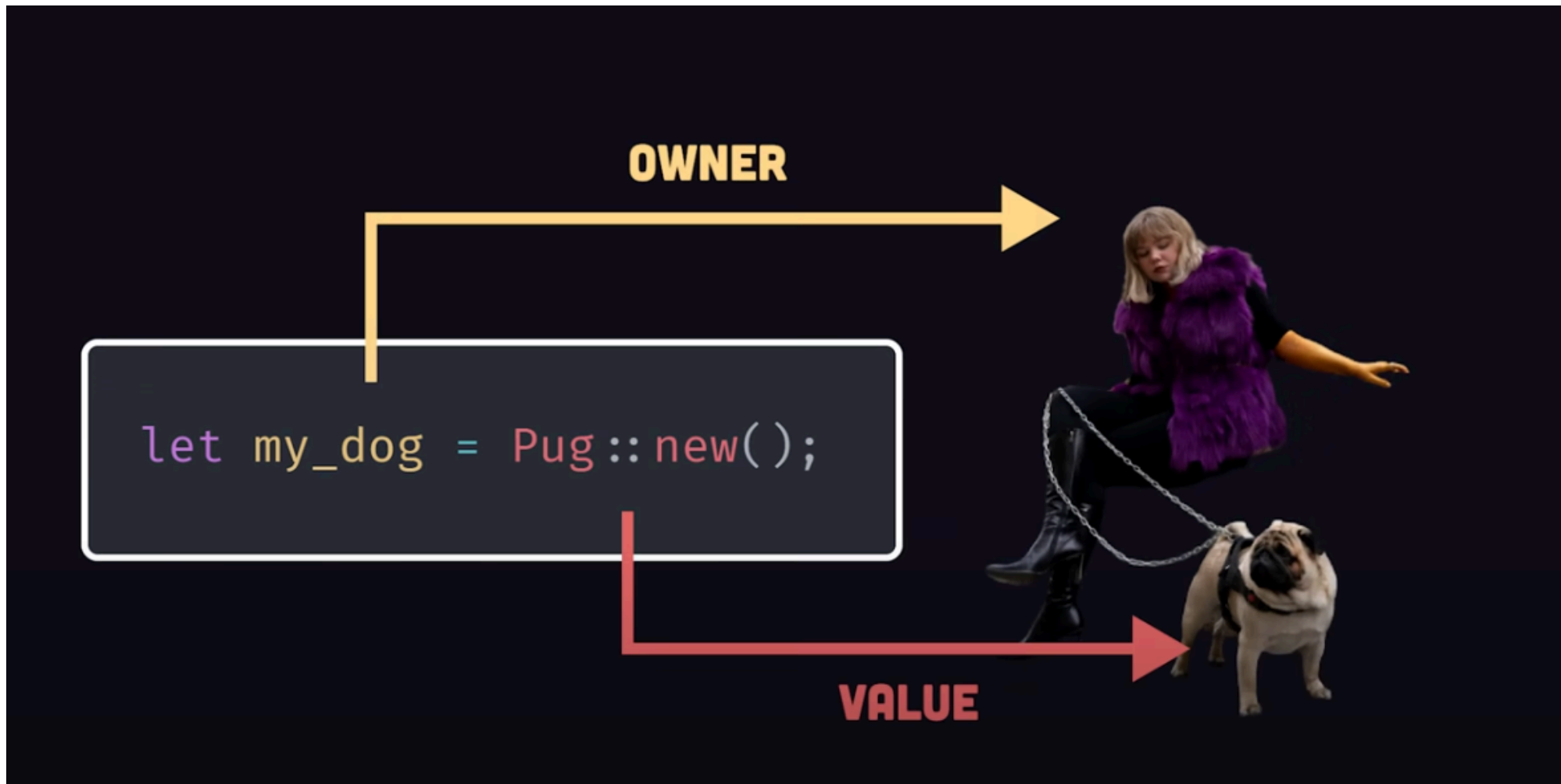
- ✓ Léger et rapide
- ✓ Sécurisé par design
- ✓ Rust = plus complexe, mais puissant

Tauri et Rust 🦀

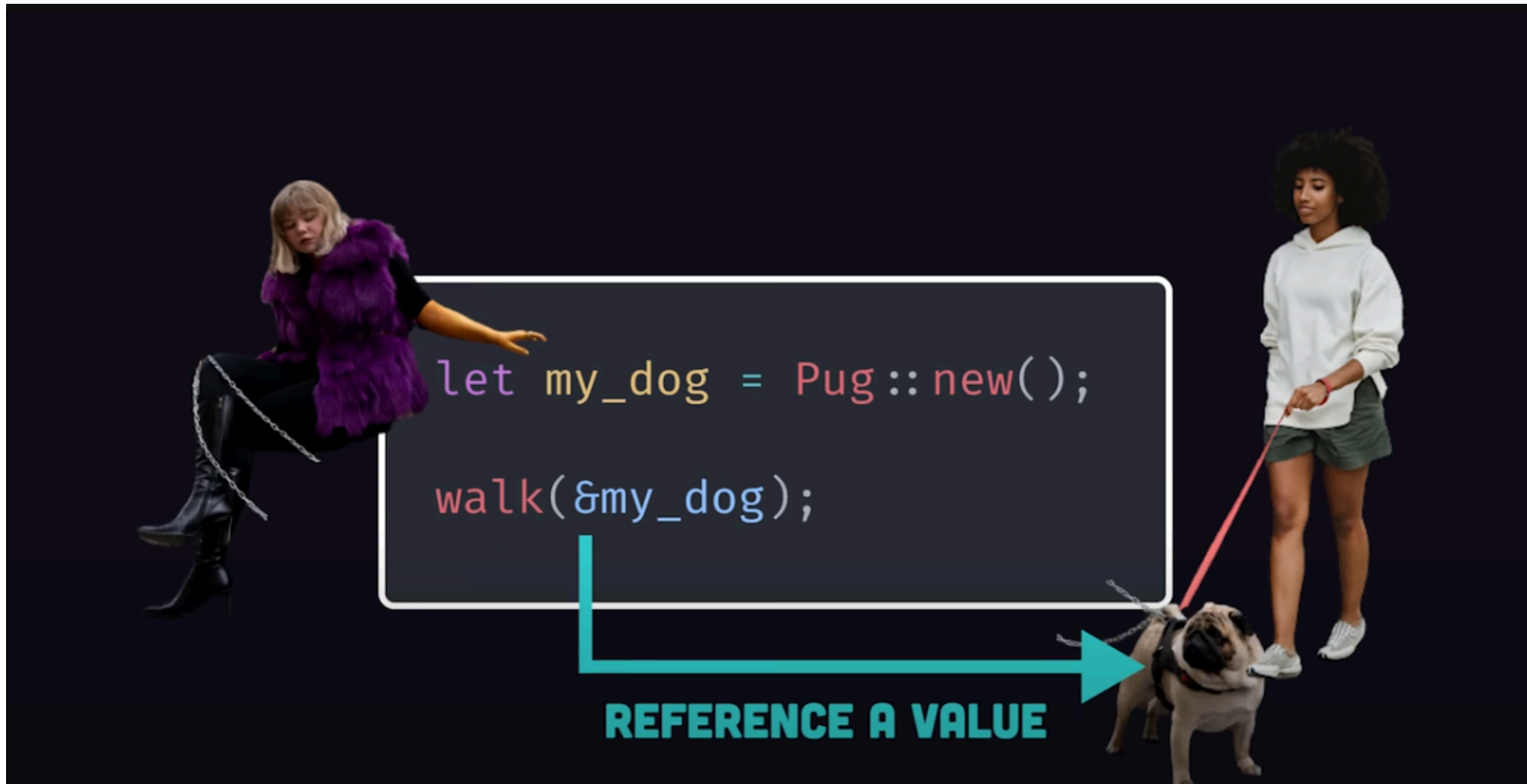
- Le **backend** de Tauri est écrit en **Rust**
- Rust = langage système moderne :
 - 🗝️ **Sécurité mémoire** avec un concept d'ownership et de borrowing
 - ⚡ **Haute performance**
 - 🧩 **Conception modulaire**

Rust évite les crashes et bugs classiques liés aux pointeurs et à la gestion manuelle de mémoire

Ownership



Borrowing



Qu'est-ce que let mut en Rust ?

Par défaut, les variables en Rust sont immuables (non modifiables).

Quand on écrit simplement :


```
let x = 5;
```

On ne peut pas changer x ensuite :

```
x = 6; // ✗ ERREUR : cannot assign twice to immutable variable
```

```
✓ Solution : rendre une variable mutable
```

On utilise alors mut :

```
let mut x = 5;  
x = 6; //  autorisé
```

Le mot-clé mut signifie donc "je veux pouvoir modifier cette variable plus tard".

Architecture interne de Tauri

Tauri repose sur deux composants internes :

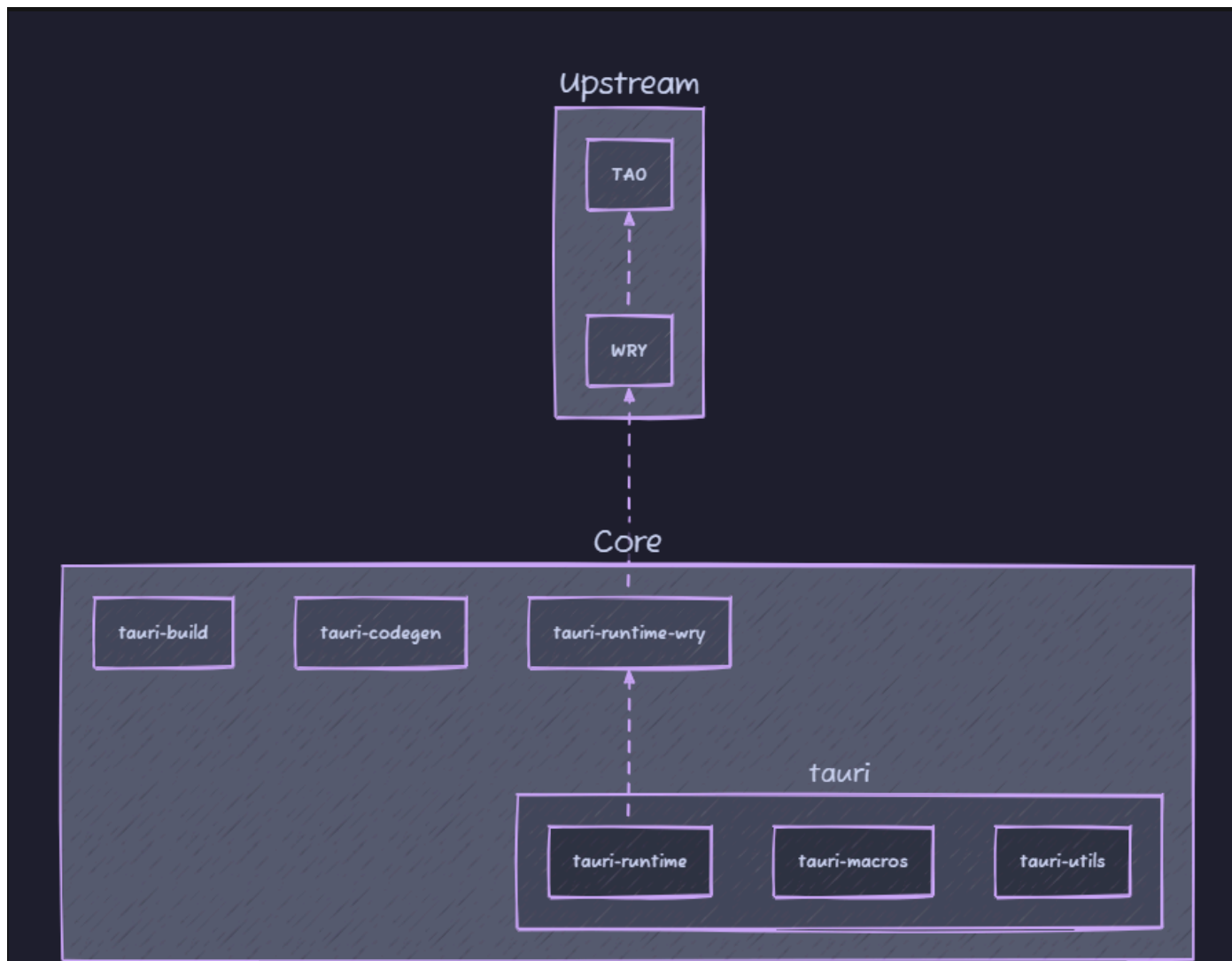
Tao

- Abstraction native multiplateforme pour :
 - Fenêtres
 - Événements systèmes
 - Menu / raccourcis / dialogues
- Alternative légère à Electron



- Intègre un **WebView natif** :
 - **WebKit** sur macOS/Linux
 - **WebView2** (Edge) sur Windows
- Gère la communication JS ↔ Rust

Architecture



Explications

Tauri : C'est la grande caisse qui tient tout ensemble. Elle réunit les temps d'exécution, les macros, les utilitaires et l'API en un seul produit final. Il s'occupe de la gestion de la configuration, de l'injection de script, il héberge l'API et gère aussi les mise à jour.

Tauri-runtime-wry : Cette caisse ouvre des interactions directes au niveau du système spécifiquement pour WRY, telles que l'impression, la détection d'écran, et d'autres tâches liées au fenêtrage.

Upstream : TAO pour la création et la gestion des fenêtres d'application, et WRY pour l'interfaçage avec le Webview qui vit dans la fenêtre.

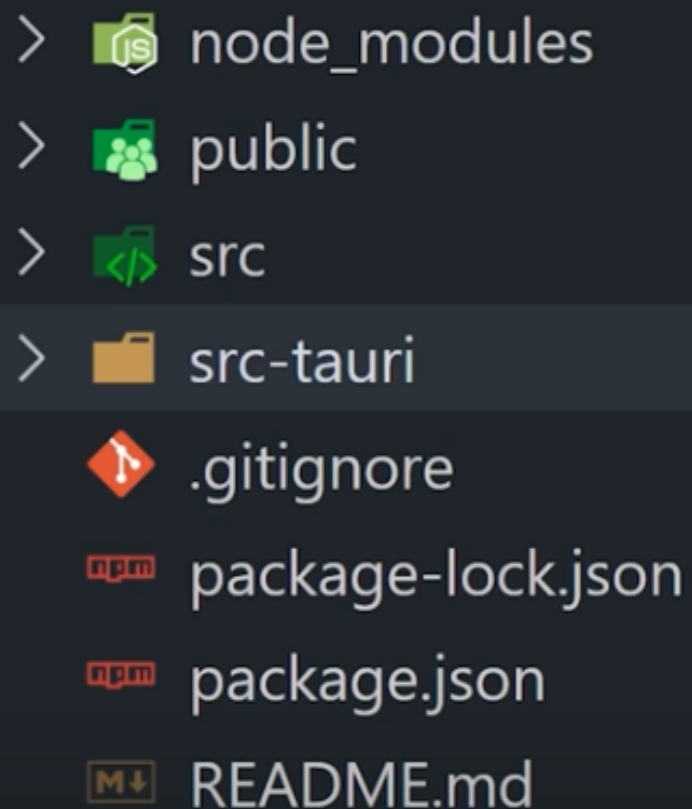
Sécurité dans Tauri

Tauri a été conçu autour de la sécurité :









- ✓ WebView isolé
- ✓ Permissions explicites par API
- ✓ Aucune API système exposée par défaut
- ✓ Communication contrôlée (commandes Rust uniquement déclarées)
- ✓ Code signé et vérifiable
- ✓ Mise à jour sécurisée (chiffrement + signature)

Comment créer une app Tauri ?

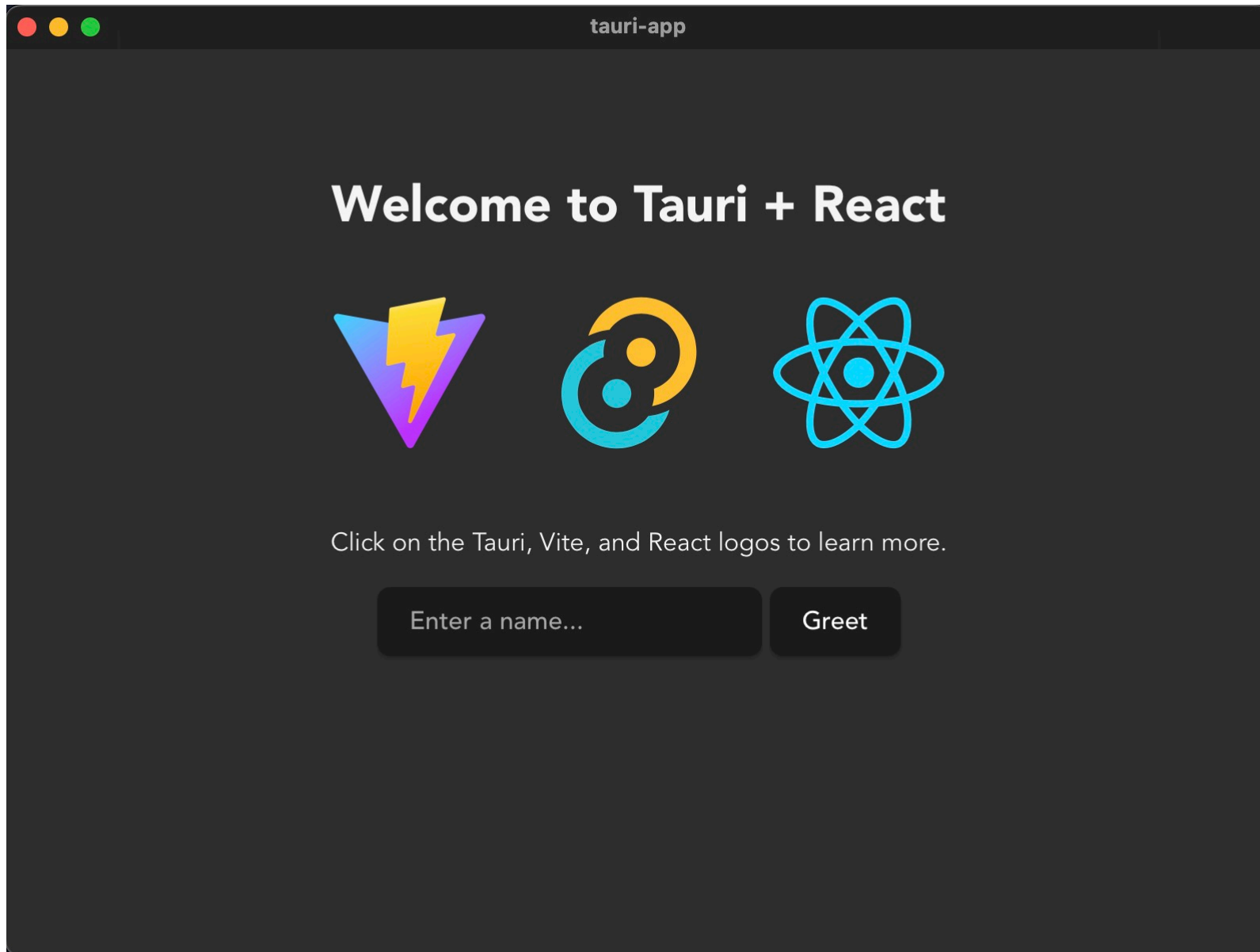
npm create tauri-app@latest



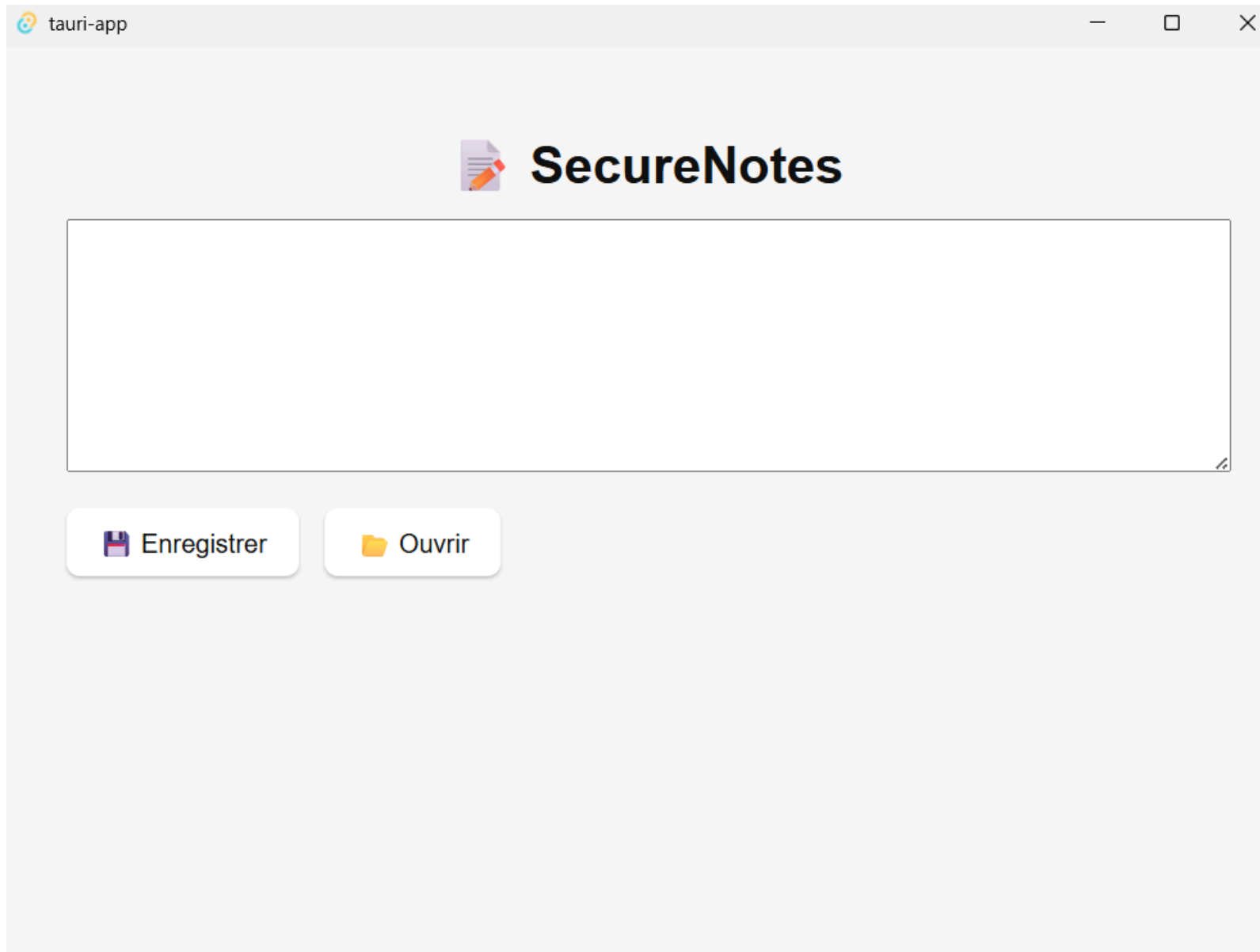
A screenshot of a file explorer window with a dark background. It shows the following items:

- >  node_modules
- >  public
- >  src
- >  src-tauri
-  .gitignore
-  package-lock.json
-  package.json
-  README.md

macOS :



Windows :



Merci 🙌

Des questions ?

Set up TP

Github : <https://github.com/Insu-qg/Tauri>

si vous êtes sur docker :

Build :

```
docker build -t tauri-builder .
```

Run :

```
docker run -it --rm -e DISPLAY=host.docker.internal:0.0 -v "${PWD}:/app" tauri-builder
```

sinon :

```
npm run tauri dev
```

TP

Des aides sont fournies plus bas.

1

Crée une application Tauri appelée SecureNotes :

L'interface (frontend) doit contenir :

- Un champ textarea pour écrire des notes
- Deux boutons : "Enregistrer" et "Ouvrir"

En appuyant sur "Enregistrer" :

- Le contenu du textarea est envoyé à Rust via `invoke()`
- Rust enregistre le fichier localement (nom par défaut : `note.txt`)

En appuyant sur "Ouvrir" :