AMBIENT-IT

+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated on 02/05/2026

Register

Tauri Training: Modern Desktop Applications

3 days (21 hours)

Overview

Tauri is a development toolkit for creating cross-platform desktop applications from a web frontend. Based on a Rust core and a WebView, it allows you to build lightweight, fast applications with a secure approach to exposing native capabilities.

Our Tauri training course will enable you to design a modern desktop application by mastering the WebView + Rust backend architecture, IPC communication, permissions management, and integration with your front-end frameworks (React, Vue, Svelte, etc.).

You will learn how to structure a Tauri project, expose reliable Rust commands, secure access to native features, and package the application for Windows, macOS, and Linux, while adopting production-oriented practices.

By the end of the training, you will be able to develop a complete Tauri application, industrialize its delivery cycle (build, bundling, CI/CD), harden its configuration, and optimize the user experience through a practical and operational approach.

Like all our training courses, this one will introduce you to **the latest stable version** of the technology and its new features.

Objectives

- Understand the Tauri architecture and the WebView + Rust backend model.
- Set up a Tauri project and integrate an existing web frontend.
- Develop IPC commands and structure the integration layer.
- Secure the application through permissions, best practices, and hardening.
- Package and distribute the application on multiple operating systems.

- Industrialize with testing, observability, and CI/CD.

Target audience

- Front-end developers
- Full-stack developers
- Developers who want to create cross-platform desktop applications

Prerequisites

- Good foundation in JavaScript/TypeScript and web tooling
- Basic knowledge of command line and dependency management (npm/pnpm/yarn)
- Knowledge of Rust is a plus (but not essential)

Tauri training

[Day 1 - Morning]

Tauri foundations and getting started

- Understanding Tauri: desktop toolkit with web frontend and Rust core
- Installing the environment: Rust, Node.js, Tauri CLI
- Project structure: src-tauri, config, lifecycle
- First screens: window, assets, navigation
- Best practices for getting started (templates, conventions, repo)
- Hands-on workshop: Initializing a Tauri app.

[Day 1 - Afternoon]

Architecture, security, and execution model

- WebView architecture + Rust backend and message exchanges
- Security model: API surface, principle of least privilege
- Permission management and command exposure reduction
- Configuration: files, environments, dev/prod profiles
- Secret and environment variable management
- Hands-on workshop: Locking a command, testing an authorized/denied call.

IPC and JavaScript Tauri API

- IPC concepts: commands, events, and serialization
- Using @tauri-apps/api: window, dialog, fs, shell (as needed)

- Error handling: typing, returns, UI display
- Architecture patterns: front-end services, "bridge" layer
- Debugging: logs, devtools, Rust-side and UI-side traces

[Day 2 - Morning]

Frontend integration (React/Vue/Svelte/Vanilla)

- Connecting a framework: web build + Tauri integration
- Routing, state management, local storage (depending on stack)
- Desktop UI: shortcuts, menus, tray (principles)
- Asset management, i18n, themes, UI packaging
- Hands-on workshop: Integrating an existing frontend and building a "Settings" screen.

[Day 2 - Afternoon]

Rust on the Tauri side: commands, state, and async

- Creating robust Rust commands (typed inputs/outputs)
- Application state management (cache, runtime configuration)
- Async: long tasks, threads, non-blocking for the UI
- Idiomatic error handling (Result, business errors)
- Input security: validation, limits

Plugins and native features

- Understanding the plugin ecosystem
- Adding native capabilities (files, notifications, OS integration)
- Separating "core" and "features" that can be enabled
- Update strategies and compatibility
- Hands-on workshop: Add a plugin + expose a native feature via a command.

[Day 3 - Morning]

Build, bundling, and distribution

- Multi-platform build: constraints and best practices
- Bundling: artifacts, signatures, icons, metadata
- Version management, changelog, release strategy
- Optimization: features, stripping, asset management
- Hands-on workshop: Produce a bundle/installer and validate on two operating systems.

[Day 3 - Afternoon]

Quality, testing, and CI/CD

- Testing: Rust unit tests, UI tests, integration tests
- CI pipeline: caches, OS matrix, artifacts
- CI secret management and signing (depending on context)
- Static analysis: lint, format, dependency audit
- Hands-on workshop: CI build pipeline + release artifacts.

Performance, hardening, and production deployment

- Observability: logs, crash reports, diagnostics
- Hardening: permission reduction, review of exposed APIs
- Update management and maintenance cycle
- Production checklist: security, performance, UX, compliance
- Hands-on workshop: Final audit (API surface + performance) and production plan.

Target companies

This training is intended for both individuals and companies, large or small, wishing to train their teams in new advanced IT technologies or to acquire specific business knowledge or modern methods.

Positioning at the start of training

The positioning at the start of the training course complies with Qualiopi quality criteria. Once they have finalized their registration, learners receive a self-assessment questionnaire that allows us to assess their estimated level of proficiency in different types of technologies, as well as their expectations and personal objectives for the upcoming training course, within the limits imposed by the selected format. This questionnaire also allows us to anticipate certain connection or internal security issues within the company (intra-company or virtual classroom) that could be problematic for the monitoring and smooth running of the training session.

Teaching methods

Practical training: 60% practical, 40% theory. Training materials distributed in digital format to all participants.

Organization

The course alternates between theoretical input from the trainer, supported by examples and reflection sessions, and group work.

Assessment

At the end of the session, a multiple-choice questionnaire is used to verify that the skills have been correctly acquired.

Certification

A certificate will be issued to each trainee who has completed the entire training course.