Stéphane ERARD

Pragmatic Full-Stack Web Architect

Nice, France | stephane.erard@gmail.com | +33 643 617 931 | GitHub | FrenchExDev | CV Website (stephane-erard-cv.vercel.app)

Full-stack developer with **20+ years of experience** in B2B and B2C product development. Specialized in **.NET/C#**, **Roslyn source generators**, and **DevOps tooling**. Builder of **FrenchExDev**, a 20-project monorepo spanning type-safe CLI wrappers, DDD frameworks, and meta-metamodeling DSLs. Seeking a **Full-Time Remote** position in an agile team.

PROFESSIONAL EXPERIENCE

Schneider Electric (via STEP UP)

EcoStruxure Automation Platform Developer / Tooling Engineer

Aug 2025 – Dec 2025 | Full-Time Remote — SAFe

- Contributed to EcoStruxure IEC 61499-based automation IDE features
- Designed git-managed Windows developer workstation bootstrap (PowerShell + winget)
- Optimized build toolchains, caching, and incremental compilation for faster developer feedback
- Participated in SAFe sprint/PI planning as part of a multidisciplinary ART

PowerShell winget TypeScript C# IEC 61499 SAFe

BIM&CO;

Senior Software Developer

2020 – 2025 | Full-Time Remote

- Refactored core feature V4 (XML) and re-developed V5 (JSON) for BIM data SAAS platform
- Revived non-working unit test suite to full operational state
- Developed MVP of new product; built PfDev developer tooling
- Extracted core feature into distributable task architecture for scalability

C# .NET 6-9 ASP.NET Web API Workers OpenAPI Azure AWS

Qwant

Senior Software Developer

2015 – 2017 | Full-Time On-Site

- Built scalable Software Factory automating Dev, CI, QA, CDel, CDep pipelines
- **Work audited for CDC investment, valorizing the company at 75M€**
- Coordinated QA developers; built developer tooling and Jenkins monitoring

Bash GitLab Jenkins Packer Vagrant Docker Selenium Node.js

CrossKnowledge

Full-Stack Web Developer

2013 – 2015 | Full-Time On-Site

- Developed E-Learning SAAS authoring tool (Flex), runtimes (Flash, HTML5), server (C#)
- **Reduced delivery pipeline from 5 days to 1 hour**
- Company sold for **$175M to Wiley & Son**

C# ASP.NET TypeScript Flex Flash jQuery

AUSY — Clients: Doctissimo, INERIS, Confidential

Full-Stack Web Developer

2010 – 2013 | Full-Time On-Site

- Doctissimo: refactored brand website (part of a lot **sold for ~18M€**); high availability, high volume
- INERIS: designed G2B application with custom state-machine framework on Doctrine
- E-Tourism website with ticket purchasing and historical pathways

PHP Symfony Doctrine MySQL Bootstrap jQuery Selenium

Freelance

Software Developer

2011 – 2012 | Freelance / Remote

- Appointment management web app with retailer/dentist scheduling
- Asbestos diagnosis web app with building annotation and photo positioning

PHP Symfony Doctrine Diem CMF MySQL

Diem Project (Open Source)

Software Developer Volunteer — Successor to founder Thibault Duplessis

2010 – 2011 | Open Source

- Took succession of founder to continue the 5.1 branch of this Symfony/Doctrine CMF
- Multiple additions, bugfixes on plugins, core components, Doctrine Nested Tree plugin

PHP Symfony Doctrine Diem CMF

Tequila Rapido

Software Developer

2009 – 2010 | Full-Time On-Site

- POC of next-generation CMS for CAC40 player using Symfony + Apostrophe
- Plugin installation system; finished implementation and release of RenaultRent.com

PHP Symfony Doctrine Apostrophe CMS jQuery

AB Croisière

Software Developer

2007 – 2008 | Full-Time On-Site

- Redesigned tariff/availability systems consuming SOAP services; XHR for better UX
- Company **sold to Promovacances-Karavel**

LAMP HTML JS SOAP XHR

TECHNICAL SKILLS

.NET Ecosystem

Core Platform: C# 13, .NET 10, ASP.NET Core, Web API, Workers, SignalR, Blazor (Server + WASM), Razor Pages, OpenAPI, EF Core, LINQ, .NET Aspire

Roslyn & Source Generation: Incremental source generators (`ForAttributeWithMetadataName`), semantic analysis (syntax trees, symbols, compilation inspection). 2-step pattern: **Extract** (Roslyn) then **Emit** (plain C# string building). Code generation for: builders, DDD patterns, CLI wrappers, requirements traceability, quality metrics.

Design Patterns & Architecture

- **Domain-Driven Design** — attribute-based, no base classes: `[AggregateRoot]`, `[Entity]`, `[ValueObject]`, `[Command]`, `[DomainEvent]`, `[Invariant]`
- **CQRS** — lightweight (interface-level separation, no event sourcing), direct injection (no mediator)
- **Result Pattern** — composable `Result<T, TError>` types replacing exceptions: Map, Bind, Recover, Tap, Ensure
- **Builder Pattern** — source-generated async construction with fluent API, validation, circular reference detection
- **Finite State Machine** — generic FSM with enum-typed states and transitions

- **Meta-metamodeling (M3)** — 5 self-describing primitives forming the foundation for all DSLs. 4-layer metamodel: M0 (runtime) → M1 (domain) → M2 (DSL) → M3 (meta-metamodel). 5 DSLs built on M3: DDD, Content, Admin, Pages, Workflow

SOLID Principles (Applied, Not Theoretical)

- **Interface Segregation** as primary driver — 1-method interfaces as standard
- **Dependency Inversion** via optional constructor parameters with defaults
- **Open/Closed** through extension points and middleware pipelines
- **No mocking frameworks** — hand-written Fakes only, testing through real interfaces
- 100% code coverage as **design pressure**, not just a metric

Testing & Quality

- **TDD** — tests drive design, not just verify it
- **QualityGate** — custom Roslyn-based static analysis: cyclomatic complexity, cognitive complexity, coupling (Ca/Ce), cohesion (LCOM4), maintainability index. Quality gates in YAML, enforced in CI
- **Mutation testing** (Stryker) — test quality score >= 0.95
- No mocking frameworks — FakeHttpClient, hand-written fakes, deterministic tests

Design-Time Engineering (distinctive skill)

Building **design-time tooling** that runs before code generation:

- **Design pipelines** — composable middleware for scraping, transforming, and serializing versioned data
- **Container-based scraping** — parallel image builds + container execution via channels
- **Version diffing** — merge multiple version trees, compute `[SinceVersion]`/`[UntilVersion]` annotations
- **Help parsers** — GNU, Cobra, argparse, custom formats (5 parser implementations)
- **JSON schema processing** — download and transform versioned schemas from GitHub/GitLab

Requirements Engineering

- **Requirements as code** — abstract classes with acceptance criteria as abstract methods
- 4-layer traceability: Requirement → Specification → Implementation → Test
- Compile-time enforcement: the build fails if any AC is unlinked
- `typeof()` and `nameof()` references (never strings)

DevOps & Infrastructure

CI/CD: GitLab CI/CD (pipelines, runners, package registry, container registry), Jenkins, Job DSL. Dev / CI / QA / CDel / CDep problem-space separation.

Containers: Docker, Docker Compose (V2), Podman, Podman Compose — all CLI tools wrapped via BinaryWrapper with full IntelliSense. Packer (VM image building), Vagrant (VM lifecycle with typed event hierarchies).

Scripting: PowerShell 7+ (modules, cmdlets, tab completers, structured logging, DevPoSh), Bash. **Cloud:** Azure, AWS, self-hosted GitLab CE, Traefik, Alpine Linux provisioning.

Frontend	TypeScript, Node.js, vanilla ES6+, HTML5, CSS3, responsive design, marked.js, Mermaid, highlight.js
PHP (legacy)	Symfony, Doctrine, Diem CMF (successor to founder), LAMP
AI-Assisted	Claude Code (daily driver), Claude Code Skills, multi-LLM integration (Claude, OpenAI, Ollama)

EDUCATION & LANGUAGES

2007 — BTS Informatique de Gestion (2-year apprenticeship)

2004 — BAC STI (Sciences et Technologies Industrielles)

French (native) | **English** (fluent, written and spoken)

SOFT SKILLS & PHILOSOPHY

Strong team spirit and communication | Self-taught, continuously learning | Agile methodology practitioner | Remote work experience across multiple positions

Mojos: **"anything-as-code"** & **"you better make it a module"**

FRENCHEXDEV ECOSYSTEM

Personal monorepo — **20+ top-level .NET & PowerShell projects**, 57 .csproj files. Philosophy: **type safety over conventions** — if the compiler can enforce it, it should. Source generators replace runtime reflection. Sealed records replace mutable classes. Explicit Result types replace thrown exceptions.

C# 13 .NET 10 Roslyn Source Generators PowerShell 7+ DDD CQRS TDD

.NET Libraries

- **Result** — Composable `Result<T, TError>` types replacing exceptions — Map, Bind, Recover, Tap, Ensure
- **Builder** — Source-generator-powered async object construction with fluent API, validation, circular reference detection
- **BinaryWrapper** — Flagship project. CLI binary → typed C# API via help scraping, JSON serialization, and Roslyn source generation. 5 pluggable help parsers, multi-version awareness
- **FiniteStateMachine** — Production-grade async FSM — Dynamic/Typed/Rich tiers, guards, hierarchical states, Mermaid export
- **Wrapper.Versioning** — Generic design pipeline for downloading/transforming versioned items from GitHub/GitLab APIs (pagination, channels)
- **JsonSchema** — Design pipeline specialization for versioned JSON schema downloads (used by Docker Compose Bundle)
- **Dsl (M3)** — Meta-metamodeling framework — 5 self-describing primitives forming the foundation for all DSLs
- **Ddd** — Attribute-based DDD DSL — `[AggregateRoot]`, `[Entity]`, `[ValueObject]`, `[Command]`, `[DomainEvent]`, `[Invariant]`
- **Requirements** — Type-safe feature tracking DSL — requirements as abstract classes, ACs as abstract methods, compile-time traceability
- **Diem (CMF)** — 62-project Content Management Framework — 4 sub-DSLs (Content, Admin, Pages, Workflow) generating full-stack Blazor apps
- **HttpClient** — HTTP abstraction + `FakeHttpClient` for deterministic testing — fluent response builders

CLI Binary Wrappers (built on BinaryWrapper)

Binary	Versions	Commands	Parser
Podman	58 (4.1–5.8)	180+	Cobra
Docker Compose	57 (2.20–5.1)	37	Cobra
Vagrant	7 (2.4.3–2.4.9)	30+	Custom
Packer	Multiple	8 groups	Custom
Podman Compose	6 (1.1–1.5)	25	Argparse
GitLab CLI	2+ (1.46–1.47)	50+	Custom
Docker	In progress	—	Cobra

Infrastructure & VM Orchestration

- **Vos** — VM orchestration framework (up/halt/destroy/provision/snapshot) with typed contributors and config merging
- **Vos.Alpine / Vos.Alpine.DockerHost** — Alpine-specific VirtualBox defaults + Docker provisioning layer
- **Packer.Alpine / Packer.Alpine.DockerHost** — Type-safe image building pipeline: Packer CLI → Alpine OS config → Docker layer → golden VM image

Applications

- **QualityGate** — Roslyn-based static analysis: cyclomatic/cognitive complexity, coupling (Ca/Ce), cohesion (LCOM4), mutation scores. YAML quality gates, CI/CD exit codes
- **DockAi** — AI-powered document indexing & search (Lucene.Net, multi-LLM: Claude/OpenAI/Ollama, Blazor viewer)
- **Doc2Pdf** — Document converter (DOCX, XLSX, PPTX, RTF, TXT → PDF) — library + CLI tool
- **Alpine.Version** — Alpine Linux version discovery from CDN with filtering, checksums, architecture support

PowerShell Modules (40 modules — curated selection below)

Developer Environment

- **DevPoSh** — Developer shell profile: UTF-8, structured logging, module auto-loading, VS Code integration

- **Dotnet.PoSh** — .NET SDK/runtime management, project scaffolding, NuGet helpers
- **VsCode.PoSh** — VS Code workspace & extension management from PowerShell
- **Git.PoSh** — Git workflow automation, branch management, commit helpers
- **GitHub.PoSh** — GitHub API interactions: repos, releases, actions

Infrastructure & Containers

- **MyInfra.PoSh** — Infrastructure-as-code utilities for VM management, network config, environment provisioning
- **Docker.PoSh** / **DockerCompose.PoSh** — Container lifecycle & compose stack management
- **Vagrant.PoSh** / **VirtualBox.PoSh** — VM provisioning & hypervisor control
- **Packer.PoSh** / **Packer.Alpine.PoSh** — VM image building pipelines (Alpine, Debian)
- **Traefik.PoSh** / **Traefik.DockerCompose.PoSh** — Reverse proxy configuration & deployment
- **Kubernetes.PoSh** — Cluster management, kubectl wrappers, pod lifecycle

Services & Networking

- **GitLab.DockerCompose.PoSh** — Self-hosted GitLab CE deployment & management
- **Keycloak.PoSh** / **Keycloak.DockerCompose.PoSh** — Identity & access management
- **PiHole.PoSh** — DNS ad-blocking server management
- **MkCert.PoSh** — Local TLS certificate generation for dev environments
- **SshConfig.PoSh** — SSH config management, key rotation, host aliases
- **Claude.PoSh** — Claude Code VM lifecycle management (restart, reset, workflow automation)

C# PROFESSIONAL HOME LAB

Self-hosted development platform: **GitLab CE** forge, CI/CD pipelines, container stack with typed C# wrappers for every tool. NuGet package registry, quality gates, VM provisioning (Vagrant + Packer + Alpine Linux), Traefik reverse proxy.

GitLab CE Podman Docker Compose Vagrant Packer Traefik Alpine