```
.##....##.########.##.......##........#######.
.##....##.##.......##.......##.......##.....##
.##....##.##.......##.......##.......##.....##
.#########.######...##.......##.......##.....##
.##....##.##.......##.......##.......##.....##
.##....##.##.......##.......##.......##.....##
.##....##.########.########.########..#######.
```

[info] ──

[info] Mattia Chiesa

[info] Full Stack Developer

[info] tia@eki.studio

[info] +41 79 380 99 35

[info] Contrada Mondrigo 6

[info] CH - 6616 Losone

[info] https://github.com/tia-lab

[info] ──

```
~/CV
bun init

Where should we create your project?
> mattia_chiesa

cd ./mattia_chiesa
```

> about

[info]

I began coding in 1999 at the age of 12, and programming has remained a central part of my life ever since. What started as a passion eventually became my profession in 2015, supported by a background in mathematics and computer systems. This foundation shaped the way I approach both design and problem solving — always with clarity and care.

Over the years, I've grown into a full-stack developer with a focus on building scalable internal tools and frameworks that improve developer experience and reduce complexity. I work primarily with React, Next.js, and modern frontend architectures — always with a strong eye on design and performance.

Among the frameworks I've built is one that integrates GSAP animations at the component level, with transitions orchestrated via a shared config file. It emphasizes separation of concerns, so that animations scale with structure, not chaos.

Much of my recent work has focused on building robust systems using tools like Zustand, Next.js, and GSAP, alongside custom CLI tooling, TypeScript-powered Sass utilities, and scalable design system layers. These solutions are built with long-term maintainability, clarity, and team efficiency in mind.

[debug]

Lately, I've been experimenting with Rust to improve performance in system-critical workflows and sharpen my understanding of lower-level architecture.

I'm currently in the early stages of building an AI orchestration in Node framework that connects a relational database for logic and task flow with a vector database for semantic memory and retrieval.

The system is designed to include basic health checks, automatic model selection based on task type and context, and fallback logic for reliability. The goal is to eventually route requests across multiple providers (like OpenAI, Anthropic, and Gemini) with transparency and control.

Long-term, I hope to integrate custom neural networks and training workflows — forming the foundation of a lightweight, modular framework for building AI-powered applications, with observability and developer ergonomics in mind.

~/CV/mattia_chiesa

ls

about.ts experience.ts stack.ts works.ts lab.ts profile.ts

bun about.ts

> experience --section=spot

[info] ──

Lead Developer — SPOT Werbung
2024 - 2025, St. Moritz (Switzerland)

At SPOT Werbung, I supported the restructuring of the digital department's development workflows to enable a more autonomous and scalable setup across design, development, and content. My focus was on internal tooling, multi-tenant architecture, multi-language support, and improving delivery consistency and speed.

When I joined, most projects were still built with Nuxt. I introduced a custom Nuxt-based starter — used in production for Davos.ch — that adopted a more componentized structure using TSX-style logic. However, its limitations soon became clear, and I helped lead the transition toward a modern, fully Next.js-based architecture.

I then developed a Next.js + TYPO3 starter tailored for multi-tenant platforms. This setup became the foundation for large-scale hospitality projects like Engadin, enabling us to manage multiple frontends, multiple backends, and shared code from a single monorepo. The system also included automated deployment pipelines via Vercel.

One of the biggest shifts I contributed to was moving from a split frontend/backend model to a unified full-stack workflow — where a single developer could take ownership of an entire project. To support this, I evaluated various CMS options, from Sanity to Payload, and chose DatoCMS for its user-friendly interface, robust multi-language support, built-in GraphQL, and excellent revalidation capabilities. Using Dato alongside Turso DB, I began building an internal framework that allowed design, backend, and frontend to integrate seamlessly.

This framework evolved from earlier animation-oriented work and brought GSAP to the component level — with transitions orchestrated via config. Design tokens were written in TypeScript and compiled into Sass utilities and CSS variables. Core components were tightly bound to CMS data, enabling a pattern of "call the component, pass the data, done" — with full TypeScript and GraphQL support via `gql.tada`.

To support this architecture, I also developed:
> A TypeScript-to-Sass variable generator for breakpoints and design tokens
> An environment switcher to support multi-tenancy across environments
> Scripts for linting, formatting, and standardizing team workflows
...

~/CV/mattia_chiesa

bun experience.ts --section=spot

I contributed multiple DatoCMS plugins, including one that integrated Formspree directly into the CMS, allowing editors to create and manage forms and responses without leaving the platform.

In January 2025, I was asked to support the Ticino Ticket PWA — a multilingual tourism app for the Canton of Ticino — originally to help resolve performance and integration issues. After a brief audit, we decided to rebuild the frontend entirely. Working with another developer, we delivered a full rewrite in just 12 days.

The result was a faster, more scalable application that significantly improved load times and stability. I focused on structuring the data fetch layer, implementing a cache-aware architecture that handled multiple external APIs (Sanity, Cariboo, TYPO3, Laravel) using Next.js, Zustand, and a custom ServerDataProvider. The system included support for server- and client-side rendering, LRU caching, localization, and granular revalidation to optimize both performance and reliability across varied user contexts.

As a side effort, I introduced CLI tooling using Ink, designed for future use with Turborepo, and explored auto-versioning workflows via Git.

Throughout my time at SPOT, I was also involved in team support — onboarding new developers, sharing internal knowledge, and leading in-person training sessions, including a team workshop in Innsbruck.

After the unexpected passing of the founder in July 2024, I did my best to help maintain momentum and ensure stability across our ongoing work.

> experience --section=vovi

[info] ──

Lead Developer — Vovi Studio
2021 - 2023, London

At Vovi, a creative studio focused on Webflow and high-end digital design, I worked on improving development workflows and bridging the gap between visual design and code — particularly at a time when Webflow was far more limited than it is today in terms of extensibility and developer tooling.

One of my main contributions was the creation of an internal framework that extended Webflow's capabilities. I built a local development setup that allowed us to use traditional Git-based workflows while syncing changes back into the Webflow Designer and ship via Vercel and allow the devs to use tools like Alpinejs. This gave developers more freedom without disrupting the visual-first process that Vovi's design team relied on.
...

~/CV/mattia_chiesa

bun experience.ts --section=vovi

This setup enabled the team to build expressive, high-performance websites with faster delivery times, without compromising on design quality or frontend robustness.

Alongside this, I continued developing with React and Next.js. In 2023, I created a fully featured Next.js framework tailored for animated, storytelling-driven websites.

My time at Vovi helped me better understand the design process and how to build tools that support it.

This role gave me a strong appreciation for workflows that balance creativity with engineering structure, and it shaped how I continue to approach cross-disciplinary collaboration.

> experience --section=background

[info] ──

Previous roles

Earlier roles included frontend and full-stack development across freelance and agency work, where I focused on JavaScript, design systems, and adaptable tooling to meet the varying needs of clients.

Alongside React, I regularly worked with platforms like WordPress and Webflow to deliver tailored solutions that balanced design, content, and business constraints. Prior to transitioning into full-time development, I worked as a mathematics teacher until 2016 — a background that continues to inform my structured thinking and problem-solving approach.

[done] ──

~/CV/mattia_chiesa

bun experience.ts --section=background

> stack

[info]

Tech Stack Overview

I've had the opportunity to work across a variety of frontend, backend, and architectural stacks — often with a focus on developer experience, design integration, and long-term maintainability. Below is a general overview of the tools and technologies I've worked with most.

> Frontend

React, Next.js, TypeScript, GSAP, Zustand
I typically work with my own Sass-based starter built from TypeScript design objects, ensuring visual consistency and responsive flexibility from a single source of truth.
I value accessible UI and scalable interfaces when needed — often using tools like Radix, React ARIA, and shadcn/ui depending on project requirements.
Animations are handled with GSAP, layered directly into components and orchestration configs.

> Backend

Node.js, Express, Payload CMS, REST & GraphQL APIs
Also experienced with Laravel (API layer for PWA) and GROQ (used extensively during Sanity projects for custom querying and data shaping).
Often working across multi-service environments, CMS integrations, and backend-connected frontends in monorepo setups.

> Architecture & Tooling

Multi-tenant platforms, monorepo setups (Turborepo), CLI tools built with Ink, custom TS-to-Sass generators, and automated design token pipelines.
Strong experience managing shared infrastructure, component-level animation systems, and internal framework architecture.

> Databases & Dev Infrastructure

Turso (SQLite edge), Neon (PostgreSQL), Pinecone (vector search), LRU caching
Vercel for CI/CD and deployment, GitHub Actions for automation.

...

~/CV/mattia_chiesa

bun stack.ts

> stack

> AI & ML (in porogress)
LangChain, OpenAI, Anthropic, Gemini integration
Building orchestration systems with provider fallback, health checks, typed memory layers, and model selection logic.
Working toward a modular framework for building AI-native apps with semantic memory and custom model support.

> CMS Platforms
DatoCMS, Sanity, TYP03 (data-only), Payload

> Ancillary Tools
Framer, historical Webflow experience, PostCSS, React animation libraries like Framer Motion and React Spring, Prismic.

> Currently Exploring
Rust for systems programming, CLI tooling, and backend services.
Expanding my skills in applied AI — from neural model training to scalable architectures for AI-powered applications.

[done] ──

~/CV/mattia_chiesa

bun stack.ts

> works --selected

[info] ──

A selection of projects I contributed to during my time at Vovi Studio and SPOT Werbung.

Vovi Studio

> [Plank](https://www.joinplank.com/)

> [Axis Arbor (SOTD, DEV)](https://www.axarb.com/)

> [Black Cube](https://www.blackcube.com/)

> [GetYourGuide - Careers](https://www.getyourguide.careers/)

> [8VC](https://www.8vc.com/)

> [PrimaryBid](https://www.primarybid.com/)

SPOT Werbung

> [Engadin](https://www.engadin.ch) — multi-tenant tourism platforms
  → Shared infrastructure also powering sils.ch, freestylestmoritz2025.ch, and others

> [Beatus](https://www.beatus.ch) & [Ermitage](https://www.ermitage.ch) — multi-tenant hotel platforms

> [Skischule St. Moritz](https://www.skischool.ch)

> [Ticino Ticket](https://my.ticino.ch) — rebuilt PWA

> [CSCS Reports](https://report2023.cscs.ch/)

> [Davos](https://www.davos.ch) — early collaboration around a Nuxt-based starter

~/CV/mattia_chiesa

bun works.ts --selected

> works --selected

Alongside client work, I actively supported the evolution of our internal tooling and development standards.

This included helping restructure the dev workflow, delivering in-person training sessions (including a workshop in Innsbruck), and continuing my own personal research into AI orchestration, developer infrastructure, and Rust-based tooling.

[done] ──

> lab

[debug] ───

AI Analysis Bot

A personal exploration into building a flexible AI orchestration layer capable of routing requests across different LLM providers (like OpenAI, Anthropic, Gemini) based on context, reliability, and task type.

The idea is to combine a relational database (Turso) to manage logic and task state with a vector database (Pinecone) for semantic memory and retrieval. Initial prototypes use TypeScript, with early experiments focusing on typed model responses with Zod, basic health checks, and fallback strategies for provider selection.

Although still in early development, the broader vision includes integrating custom neural networks and lightweight training pipelines to expand control and refine responses over time.

The end goal is to build a modular, developer-oriented AI framework that emphasizes observability, reliability, and provider-agnostic logic — while remaining adaptable to future tooling and use cases.

AI-Assisted CMS (Rust + Node.js + Next.js)

This is an early-stage exploration of a hybrid CMS architecture aimed at combining performance, automation, and modern developer workflows.

I've started by setting up a monorepo with Turborepo, a Neon Postgres database, and an experimental API fetching layer written in Rust — mostly to benchmark response times against common Node.js and PHP implementations.
...

~/CV/mattia_chiesa

bun works.ts --selected

bun lab.ts

> lab

Although I'm still not fully proficient in Rust, this project serves as a way to gradually build up experience, especially around efficient API boundaries and low-latency data access.

The broader vision is to use Rust selectively for performance-critical endpoints, Node.js for background tasks, and a potential admin interface built with Next.js — designed around modular content blocks and multi-tenant support.

Over time, the system could integrate design tokens from Figma and support AI-assisted tooling — enabling features like structure generation from existing websites, auto-syncing content blocks, and streamlining editorial workflows.

The long-term goal is to create a CMS framework that makes it easier to move from design to production — with type-safe APIs, integrated automation, and scalable architecture baked in from the start.

~/CV/mattia_chiesa

bun works.ts --selected

bun lab.ts

> profile

[info] ──

> Languages

> Italian   - Native
> French   - Fluent
> English   - Fluent
> German   - Learning

> GitHub

I'm happy to provide access to my GitHub repository upon request for a deeper look at ongoing and past projects.

https://github.com/tia-lab

> Contact

tia@eki.studio - +41 79 380 99 35

> Closing Notes

Thank you for taking the time to read through my profile.
I appreciate your attention — and I'm always open to new challenges, ideas, and ways to grow.

[info] ──

> SIGINT received
> Application shutting down...
> Thank you for your time.
> Process exited gracefully.

~/CV/mattia_chiesa

bun works.ts --selected

bun lab.ts