

Curriculum vitae

Miika Leppänen

Jyväskylä, Finland

+358 50 405 1123

miika.leppanen@hey.com

miikaleppanen.com

linkedin.com/in/milep

github.com/milep

Profile

Senior software developer with more than 25 years of professional experience.

My work combines software development, architecture, and product judgment. Recent professional work has focused on Shopify, payments, and integrations. I have also worked in technical leadership, product management, technical sales, recruiting, and team leadership.

I am comfortable with systems where both the business rules and the technical details are complicated. Integrations are a good example. When they work, nobody notices them. When they fail, somebody usually loses money or has to fix data by hand.

I use coding agents as part of my normal development process. I define requirements and architecture, direct implementation, and take responsibility for review, testing, deployment, and production investigation. Personal projects with AI assistants and narrative systems have also made model limits, context, latency, security boundaries, and behavior testing practical design concerns for me.

Main areas

- Software architecture
- Shopify and eCommerce integrations
- Payment integrations
- ERP, 3PL, PIM, CRM, and POS integrations
- REST APIs, GraphQL, and webhooks
- ETL and data synchronization
- Background jobs and scheduled processing
- Coding agents and agent workflows
- Ruby on Rails
- JavaScript and TypeScript
- SQL and NoSQL databases
- Cloud deployment and CI
- Production debugging
- Product management
- Technical sales
- Team leadership and recruiting

Experience

Woolman

June 2025 to present, 80%

Software developer, payments team

I currently work in Woolman's payments team. We build and maintain integrations between Shopify and payment providers.

The work includes payment gateway APIs, Shopify order and payment flows, capture logic, payment reconciliation, duplicate processing, scheduled jobs, fulfillment events, and production support.

I also use coding agents for investigation, implementation, testing, and maintenance. I have introduced reusable agent skills, production investigation tools, operational scripts, runbooks, and end-to-end tests.

UK luxury department store capture app

This was a separate customer project. I took ownership after the initial discovery.

The application handles payment capture cases where the customer's fulfillment process does not map cleanly to Shopify's standard flow.

- Worked with the customer to define payment and fulfillment cases.
- Added payment tracking and duplicate capture protection.
- Added Klaviyo events driven by webhooks.
- Added deduplication and delayed processing for events.
- Built workflows for fulfillment, cancellation, and Click & Collect.
- Handled architecture, implementation, review, testing, deployment, and production support.
- Used a fully agent-based development process. Coding agents produced the implementation code.

Indoor and outdoor living brands capture app

This was another customer project. I led the architecture and initial implementation.

- Designed the webhook and worker model.
- Added HMAC verification and event deduplication.
- Added capture safeguards, audit trails, retries, and recovery.
- Defined rules for shipping charges, gift cards, short picks, cancellations, capture timing, and authorization limits.
- Directed coding agents during implementation.
- Worked through the customer's security review.
- Handed final testing and deployment to another team member.

Osgenic

August 2022 to present

20% since June 2025

Tech lead / lead web developer

I currently work at Osgenic one day a week.

- Guide technical and architectural decisions for the web portal.
- Take part in development when needed.
- Work on backlog and roadmap priorities.

- Help connect product decisions to customer and business value.
- Introduced regular one-to-one meetings with developers.
- Recruited full-stack developers.
- Worked with company management on development practices and some HR topics.

Before June 2025 this was my full-time role.

Woolman

August 2017 to August 2022

Shopify integrations architect / tech lead

I joined Woolman as a developer and later took on architecture, team leadership, product management, resourcing, and sales support.

- Built Woolman's internal multitenant Shopify integration platform.
- Built integrations between Shopify and ERP, 3PL, PIM, CRM, and POS systems.
- Worked on large data synchronization jobs and Shopify API limits.
- Built the first version of the Nordic Shipping App.
- Helped grow the development team to six developers.
- Led technical work and took part in product management.
- Worked directly with customers on integration requirements.
- Managed developer resourcing between projects, product work, and maintenance.
- Helped sales with technical design, estimates, and project scope.

Shopify integration platform

I designed and built the first version of a multitenant integration system for connecting Shopify to back-office systems.

The system was basically an ETL pipeline with background workers and scheduled jobs. It handled millions of messages per month.

A lot of the work was around practical integration problems. Shopify API limits, large product catalogs, caching, retries, failures, and customer-specific rules all had to be handled without making the system too hard for a small team to maintain.

Role: Initial implementer, architect, team lead

Main stack: Ruby on Rails, Sidekiq, PostgreSQL, Redis, Heroku

Nordic Shipping App

I designed and built the first version of an application for showing Finnish shipping methods in Shopify checkout.

It queried several carriers for pickup points and let merchants print shipping labels. Each carrier had its own API, configuration, and contract process.

The application handled customer Black Friday traffic and had about 200 installations by the time I left Woolman.

Role: Initial implementer, architect, team lead

Eebu Finland Oy

January 2010 to April 2017

Co-founder / software developer

We started Eebu to build SaaS products for small and medium-sized businesses.

Our main product was an ERP with double-entry bookkeeping, inventory, sales orders, and CRM. I also built an invoicing service and a booking application. In several products I was the only developer.

We financed product development with consulting work, including eCommerce projects.

Being responsible for both the business and the software taught me to be careful with abstractions and premature optimization. We spent time on both before we really understood what customers needed. I still prefer code that is easy to change while the problem is still changing.

Earlier software development and consulting roles

1999 to 2010

Before Eebu I worked in software development and consulting roles at Carigna, Integware, Sofia Digital, Yomi Solutions, and Aplicom.

The work included SaaS and enterprise software, web applications, interactive systems, and embedded development. I used several languages and platforms over the years, including Java, PHP, and JavaScript.

Personal projects and writing

I use personal projects to test product ideas and technical approaches without forcing them into a customer project. Actual use, even by a small group, usually changes what I think is important.

Pi Web

Pi Web is a web interface around the Pi coding agent environment. I built it so I can use the same agent workflow from my desktop and phone.

A supervisor keeps requirements and architecture decisions in the main conversation. Subagents handle bounded work in separate sessions and return structured handoffs. This preserves context and makes parallel investigation possible without making parallel work the default.

I kept Pi responsible for models, login, tools, settings, and session files. The web interface stays focused on access and coordination instead of copying the underlying development environment.

[Shaping your own tools with agents](#)

AI assistants and narrative systems

Sleeve was a Discord-based personal assistant with tools, memory, and proactive tasks. Its Eidolon game experiment became technically elaborate without becoming clear or engaging for players. Helper Bot tested a smaller assistant through private WhatsApp chats. The familiar interface reduced setup, but it did not create a recurring reason to use the assistant.

Lore is a separate narrative game engine built from those lessons. The first design simulated nearby characters through several model calls and took up to two minutes per turn. The current design uses one call to interpret player intent and another to write the scene, reducing an average turn to about 20 seconds.

Ruby owns Lore's rules and canonical state. Models receive selected context, return structured output, and cannot declare outcomes or commit facts directly. This makes retries and rejection possible when output is invalid. Scene quality and response time are still active tradeoffs.

Coding agents wrote the implementation for these experiments. I directed their architecture and behavior. For Lore, I reviewed most code and used reusable skills and review agents. For the lower-risk Helper Bot experiment, I tested behavior and security boundaries without reading the implementation.

[Learning by building my own AI assistant](#)

[Building an AI game engine by simulating less](#)

Show Manager

Show Manager controls an always-on Raspberry Pi display and a shared YouTube queue for parties. Guests scan a QR code, search from their phones, and add music without joining my home Wi-Fi.

A web application on a VPS sends commands through a Raspberry Pi to the native YouTube app on Android TV. Search combines YouTube, YouTube Music, and a local collection of videos that people have already selected.

I built the first version over a few days with coding agents and kept changing it as actual use exposed friction. The project stays specific to my own setup rather than becoming a general platform.

[Building a shared YouTube queue for parties](#)

[Show Manager on GitHub](#)

Writing

I write at miikaleppanen.com to document current decisions and lessons from software, coding agents, personal projects, and product development. The articles are working notes rather than a complete account of every project.

[Structuring teams and product development](#)

Technical skills

Software development

- More than 25 years of professional software development
- Long-term Ruby and Ruby on Rails experience
- JavaScript and TypeScript
- Frontend development with several JavaScript frameworks
- Previous professional experience with Java and PHP
- PostgreSQL, MySQL, MongoDB, Redis, and SQLite
- Background jobs and scheduled processing
- API-based application design
- Automated and end-to-end testing

Integrations

- Shopify APIs
- REST and GraphQL
- Webhooks
- ETL and data synchronization
- Payment integrations
- ERP, 3PL, PIM, CRM, and POS integrations
- Asynchronous processing
- Idempotency and event deduplication
- Webhook authentication
- Retries and recovery
- Audit trails
- API rate limits
- Production debugging

Coding agents and language-model applications

- Agent-based software development

- Requirements, architecture, and implementation plans for coding agents
- Supervisor and subagent workflows with structured handoffs
- Reusable agent skills and review agents
- Agent-assisted testing and production investigation
- Developer tools built around agent workflows
- Tool use, memory, selected context, and structured model output
- Application-owned rules, permissions, and canonical state
- Behavior testing and model-output validation
- Response-time, quality, and cost tradeoffs

Infrastructure

- Linux
- Docker
- Heroku
- AWS and GCP
- GitHub and GitLab CI
- Self-hosted systems
- Automated deployment
- Monitoring and operational tooling
- Production debugging

Leadership and product work

- Software architecture
- Technical leadership
- Team leadership and mentoring
- One-to-one meetings
- Recruiting and interviewing
- Product management
- Backlog and roadmap work
- Customer discovery
- Requirements analysis
- Technical sales
- Project estimation
- Customer and stakeholder communication
- Balancing new development, maintenance, and technical debt

How I work

I want to understand the user and business problem before making the technical solution complicated. A familiar interface or a capable model does not create a real need by itself.

Early in a product I prefer simple code that can be changed quickly. If the product is wrong, a nice architecture does not help much. Once a product has real users and needs to keep working for years, reliability, maintainability, monitoring, and technical debt matter more.

I prefer short feedback loops. Developers should be able to talk to customers, domain experts, product people, and other stakeholders without information passing through too many layers.

Teams also need enough ownership to solve problems instead of only implementing tickets. I like useful processes, but I do not think every team needs the same one.

I apply the same risk-based approach to coding agents and model-based features. The application should own permissions, rules, and important state. I test the behavior that users see, and I increase code review and operational safeguards when failures would have serious consequences.

Education

Äänekosken kauppaoppilaitos

Merkonomi, Laskentatoimi
1995 to 1997