The infi Way

At Infi we like to work the following way.

The Infi Test

A quick check to see if you're roughly following *The Infi Way*. A sort of Joel Test.

1 Are developers seeing real users?

2 Is code being reviewed?

3 Are you running automated builds with tests?

4 Can you automatically deploy to production?

5 Are repositories self-documenting?

6 Does the team autonomously pick fitting technology?

7 Does the team autonomously choose their own process?

8 Is the process evaluated periodically?

9 Is work refined before it's being done?

10 Are outages noticed and handled proactively?

11 Is knowledge actively shared within- and outside of the team?

12 Are secrets managed correctly?

13 Are there (proven) working, encrypted backups?

We aim for maximum score, though sometimes you still have to work towards it.

User-centric

We work for *the client of our client*. They use software to achieve *their* goals, our goal is to help them with that.

How we like to do this:

- *Direct contact* with users
- Prototyping, iterate quickly
- Work user-centric
- Close collaboration with designers
- Work from "why"

Concrete examples:

- User testing
- Google "Design Sprint"

By us:

- Blog (Dutch): Walking Skeletons
- Site: Our clients (and their users!)

Principles

We work in a modern way, aiming for quality. "First time right" is also cheapest in the end.

How we like to do this:

- Everything in version control
- Up-to-date tools, frameworks, and libraries
- Automation where possible
- Strong test coverage
- Code quality controls
- Pair programming
- Code reviews

Concrete examples:

- Git (flow chosen per project)
- Unit tests, integration tests, end-to-end tests
- Sonarqube, ESLint, etc.
- SOLID (of DAMP), YAGNI, KISS
- Domain-Driven Design

By us:

- Blog (Dutch): E2E Testing with Puppeteer
- Blog (Dutch): Mob Programming
- Blog: Auth0 and Cypress

Code is the Source

Source code is just that: the *source*. Self-documenting, so usable for *any* developer, our own or otherwise.

How we like to do this:

- Architecture *in* the source
- Clone & run repositories
- Continuous integration, continuous delivery
- DTAP
- Infra, scripts, and monitoring in code

Concrete examples:

- Architecural Decision Records, C4 Model
- Ansible, Docker, Terraform
- GitLab (SAAS or self-hosted), GitHub
- GitLab CI, GitHub Actions, Azure Devops, CircleCI, Octopus

By us:

- Blog: Elasticsearch + Zabbix part 1 en part 2
- Talk: GitLab CI/CD

Technology

We prefer using "*the best tool for the job*"; we're generalists who can quickly specialize when the need arises.

How we like to do this:

- OS is a personal choice per developer
- We build custom software and aid with buy-over-build
- Hosting in the cloud
- Back-end in a modern stack
- Front-end with mature frameworks
- Mobile cross-platform, unless "native" *really* adds value

Concrete examples:

- Azure (via our Microsoft partnership), AWS, Google Cloud
- .NET Core, PHP (with a modern framework), or Node (with TypeScript)
- Vue, Angular, React
- TailwindCSS, Bootstrap
- ReactNative
- SQL, Document-storage, Service Buses, etc.

By us:

- Blog (Dutch): C# for a Java programmer
- Blog (Dutch): React met TypeScript en WebPack
- Talk: VueJS Deep Dive
- Blog (Dutch): Mobiele Frameworks
- Webinar (Dutch): Mobiele Apps

Process and Team

Teams work *autonomously*, but usually pick an *agile* way of working with a Product Owner from the customer.

How we like to do this:

- Teams determine and improve their processes
- Product Owner from the client
- Scrum master
- Maintain a planning *together*
- Work in small teams
- Call in help from internal and external experts
- Regularly work together with the client at the same location

Concrete examples:

- Scrum or Kanban
- Retrospectives, Lean
- Trello, Jira, Shortcut (formerly Clubhouse), YouTrack

By us:

- Blog (Dutch): Retro Uitkomsten
- Blog (Dutch): Perfecte user stories
- Blog (Dutch): Tips voor remote demos
- Blog (Dutch): Kanban vs Scrum
- Blog (Dutch): teamsamenstelling

Craft

Building custom software is a craft. We are happy (and proud!) to show what's going on behind the scenes.

How we like to do this:

- Studying and knowledge sharing
- Invest in tooling
- Great hardware
- Experiment
- Monitoring and alerting

Concrete examples:

- Internal and external presentations
- Study groups, self-study
- Visit conferences and meetups
- Kibana, Sentry, Zabbix, etc.

By us:

- INFI-CON 2020 and INFI-CON 2019
- Blog (Dutch): Spreken met Impact
- Blog (Dutch): Je eigen Linux kernel bouwen
- Blog (Dutch): "Software engineering" bestaat niet
- Blog (Dutch): Vakmanschap is meesterschap

Foundations

Some things speak for themselves: they are *always* part of the deal. Think Security, Performance, Backups, a critical note, plus the opportunity to be ourselves.

Those foundations include:

- **Backups**, including encrypted offsite backups
- **Security** as a core value in our work
- **Audits** when sensible
- **Performance**, for example via load testing
- Be critical of features, come up with alternatives
- Open Source usage and contributions
- Meetups: host and organize
- Be ourselves: (Dutch manifesto) "*Wij zijn mensen.*"

Concrete examples:

- OWASP
- Password managers, 2FA, GPG, SSH
- Locust, OctoPerf, JMeter
- Auth0, Stripe, SendGrid, Contentful, Lokalise, etc.
- Always lock your laptop!

By us:

- Manifesto (Dutch) Het Infi Manifest
- Blog: SPA Necromancy
- Blog: Launching FOSS