nilenso

A QUICK SHOWCASE

May 2020

About us

nilenso is a boutique technology consultancy based in Bangalore. We are structured as an LLP where all members are co-owners of the company.

Over the past 7 years, we have worked on projects in e-commerce, logistics, machine learning, data science, healthcare, payments and education, which we delivered in a variety of languages and platforms: Clojure/ClojureScript, Ruby/Rails, Haskell/PureScript, Go and Javascript/node.js, among others.

Our expertise is in delivering highly available, scalable systems in high throughput environments. Some of the major services we've built include a multivariate testing/experimentation platform, a ride-share driver-allocation service, multiple machine learning runtimes, and a variety of ETL and analytics systems.

You can read about some our recent projects here: https://nilenso.com/recent-tech.html.

Our developers speak regularly at conferences, the talks that were recorded over the years can be found here: https://nilenso.com/talks.html.

Official Address:

Nilenso Software LLP

#105, 10th Cross, Indiranagar I Stage, Bangalore 560038, India

Date of incorporation: 31st July 2013

Clientele

A select set of recent clients, and the work involved

Gojek is the uber + swiggy + dunzo + grofers of south-east Asia. It is operational in 4 countries, and impacts 10s of millions of customers, millions of drivers, and half a million merchants/restaurants.

We have been working with Gojek since 2015, and on multiple products:

- Allocation / match-making, and pooling
- Food, and transport CMS, and OMS
- Pricing
- Experimentation, notifications, and bundles

Talks:

- EuroClojure | Moving people with Clojure
- ClojureD | Reducing Accretion in Monoliths

- Rewrote, and architected their central driver allocation system in a few months, with a generative testing suite.
- Brought Clojure expertise to Gojek, trained multiple teams in Clojure, and continue to advise on language semantics.
- **Led, and supervised teams and product initiatives from the technical front, while working with product, business, ops, design, research, and data-science teams.**
- Established processes and practices around designing architectures, project management, and tech debt management.
- Engineered large scale, low-latency, stateful systems, and data pipelines for ML runtimes.

Staples

Staples Inc. is one of the largest American office retail companies. We consulted with StaplesLabs/SparX, which built recommendations through email and search, providing shipping estimates, and an experimentation platform amongst other products.

We worked with Staples from 2013 to 2017 and on multiple products:

- Experimentation Platform
- **Email recommendation system**
- Web crawler/scraper
- ETL, data warehousing, and analytics

Talks:

- RootConf | Building a postgres cluster
- FnConf | Building an experimentation platform

- Engineered an experimentation platform, incorporating research in white-papers from Microsoft, Netflix, and Google. Built a highly available system with automatic fail-overs.

- **Built and maintained machine learning runtimes for recommendation engines (email marketing for existing customers).**

- Orchestrated a web crawler, and scraper that would deal with unstructured datasets on the web, to provide insights into categorisation, and pricing on a daily basis.

- Built a simulation testing harness and framework, using Causatum, Simulant, and Datomic for the experimentation platform.

Summary: Gojek, Staples

With both these clients, the focus has been around building large scale systems. The long-running services would need to respond within 50ms (sometimes <1ms), often with request load of ~100K RPM (sometimes 200k RPM). Many of these services had SLAs to keep these numbers at 99.9p.

Data-engineering, machine-learning runtimes (not algorithms though), analytics, and stateful systems have been primary aspects of our work here.

Tech Stack

- **Languages:** Clojure, Ruby, Go, Java
- **Datastores:** Postgres, Redis, Redshift, HBase, BigQuery, etcd
- **Messaging:** Kafka, RabbitMQ
- **Monitoring:** ELK Stack , TICK Stack, Prometheus, NewRelic and Grafana

JUSPAY

Juspay develops mobile payment solutions: a mobile payments browser for 1-click payments, and a high-performance payment-gateway booster.

We're working with Juspay from 2018 on a few varied fronts:

- A DSL to build apps for banking clients
- **Scheduled recurring bill payment for BHIM**
- Real-time analytics dashboards for merchants
- Scaling DB to handle large volumes of traffic

Establishing processes in project management, and coordinating delivery over different time-zones was essential part of this engagement.

- Worked on writing the Purescript Framework used to create mobile apps declaratively.
- Built generic, re-usable components for the merchant dashboard, and built the dashboard as well.
- **Engineered the data pipelines for the dashboard, working with the existing teams to do so, on backend services written in Haskell.**
- Consulted with them to debug and establish best practices in maintaining a large Postgres installation, and setting up high-availability, and failovers for their DBs.

Omnyway aims to be a payments platform, providing solutions that bridge the gap between retail sales and e-commerce. It's in the B2B space, providing a platform to cross-sell, and up-sell via promos, and payment-methods.

We worked with Omnyway from 2016 to 2019 on some of their products:

- **HaiWai: WeChat integrated payments**
- ZapBuy: One Click Buy from digital ads
- API Simulator for the API offering
- Mobile App prototype for ZapBuy

Talks:

- Fragments | REPL driven mobile development
- DutchClojure | Generative testing patterns

- Built a rule-based chatbot which used WeChat APIs, and MasterCard APIs and integrated payments to enable international travellers to pay at local stores.

- Created generic e-commerce abstractions for basket, payments, and wallets to work with any 3rd party payment vendor, and the adapter architecture to enable that.

- **Shipped a cross-platform prototype app using React Native for ZapBuy within a month, with complete API integration for a demo.**

- Engineered a visual simulator, and a generative testing framework for black-box testing the 100+ APIs that were consumed by the mobile SDKs.

Summary: Juspay, Omnyway

Working on these **fintech products**, both in entirely different settings, has allowed us to learn about the internals of payment infrastructures, and the level of commitment to **data integrity, security**, and ruthless testing that is necessary in dealing with monetary transactions.

We've acquired the skills to design a good plugin-architecture that allows easily adding vendors for wallets, payments, etc, and experience in integrating with relatively unknown platforms, that don't come with full API documentation.

Tech Stack

- **Languages:** Haskell, Purescript, Clojure, Clojurescript
- **Datastores:** MySQL, Postgres
- **Messaging:** Kafka, Orpheus (home-grown)
- **Monitoring:** Prometheus

PayTM Insider is India's largest online ticketing and events company. Insider approached nilenso with a concept for an entirely new business model: to bring their live events online.

We worked with Insider from 2018 to 2019 to build their live platform that hosts digital events, with the objective of **"Streaming interactive video to one million concurrent users"**.

Even to begin with, this involved building a real-time quiz coupled with the event where hundreds of thousands of participants needed to see the questions at **the exact same time**.

Talks:

- FnConf | After the crash
- RootConf | Scalable distributed systems in Elixir

- Designed and implemented a fault-tolerant distributed system in Elixir, and Erlang's OTP tooling. And built video transcoding and streaming into the platform, using mimoLive and Wowza Cloud.

- Built a Mob Simulator, that simulated concurrent users ordering in the hundreds of thousands, to load test the system.

- Synchronised the quiz and video by stamping metadata onto the stream, self-identifying sync time, and compensated for server-side transcoding delays on mobile clients.

- Built a dashboard to create the quiz, publish questions and answers to the massive live audience, and coordinate transitions from one question to the next.

- **Transitioned to an in-house team over months, while coaching them on Elixir, and OTP as well.**

QUINTYPE TECHNOLOGIES

Quintype provides a complete suite of digital publishing solutions, aimed at bringing the Indian paper based media to the digital age.

We worked with Quintype from their inception in 2014 to 2016 to build:

- **A CMS for editors to write and publish stories**
- Tools for social media outreach and email marketing
- Caching and CDN integration for performance
- Ad network integrations for publishers
- SEO for dynamic content

In addition to building software, we also helped them build their internal tech team by providing referrals, and interviewing candidates for them.

- **Built a state of the art CMS from scratch, with interactive drag-and-drop editorial support, and launched multiple media outlets onto the platform in a short time.**

- Engineered a fully customisable real-time analytics platform for administrators to segment and view data on ads, impressions, and click streams. Also wrote a query generator to automatically transform data from OLTP to OLAP databases.

- Worked on the customer, and admin facing mobile apps, designed significant parts of the editor interface.

- Built progressive web apps with Clojurescript, React, and Rum.

Summary: Insider, Quintype

With both Insider and Quintype, we were **involved in the product since the inception**, and carried the product to a reasonable amount of success, and maturity in production. We were also able to help in grooming, and **building their internal teams prior to rolling off** the project.

Tech Stack

- **Languages:** Elixir, Clojure, Ruby, Javascript, Clojurescript
- **Datastores:** Postgres, Redshift
- **Messaging:** Kafka

Simple

Simple is an open-source, award-winning app (an initiative of Resolve To Save Lives) that helps improve hypertension control and prevent heart attacks and strokes by making it easy for healthcare workers to manage blood pressure measurements and medications.

We have been working on Simple.org since 2018 across multiple verticals:

- Building and managing the backend web stack
- The multi-country deployment infrastructure
- **Operational and product decision making**

Talks:

- Fragments | Engineering to embrace change

- Worked on building a fast, production-ready app prototype in ClojureScript + ReactNative along with the actual native Android app to enhance feedback cycles.

- **Built fully offline-first sync APIs to allow for the app to be used in rural areas with poor internet connectivity.**

- Designed an offline-first phone-masking system using DTMF tones in low network areas, to protect the privacy of nurses.

- Established processes and communication channels with health-officials from different states, the ICMR and CVHOs from various health clinics.

- **Built complex dashboards around health data via the power of materialised views and Russian Doll caching.**

How we work

On XP, project management, and other processes

eXtreme Programming

At nilenso, we follow agile software development practices, which are iterative and collaborative in nature.

Our default delivery style is based on the classic eXtreme Programming methodologies which emphasise tight feedback cycles by means of **1-week iterations, daily standups,** pair programming, and heavy communication.

Inception:

We typically kick off engagements with an inception meeting or scoping exercise.

The majority of the inception attempts to capture features in "user stories", as these frame each feature in terms of how it benefits the end user, estimate them if necessary, and prioritise them.

User Stories:

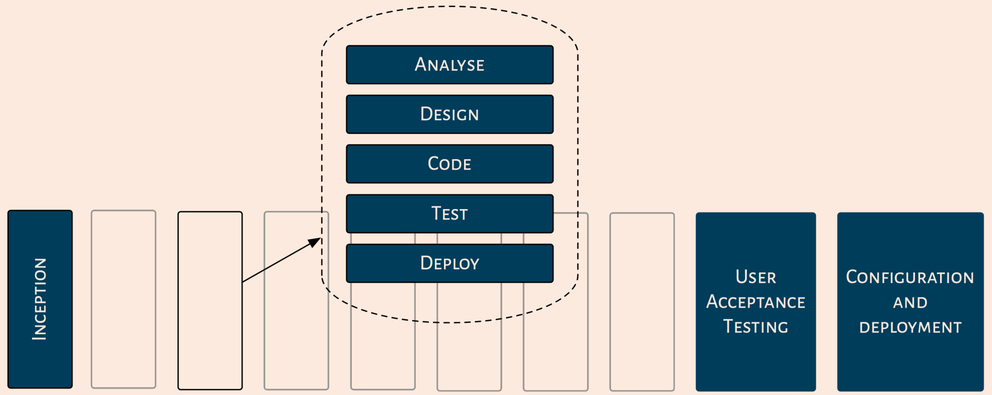
- Whether stories are individually estimated or not, they are then packed into a few sample weeks of the project to rough out the expected pace of development.
- Stories can be reprioritised as the project progresses and the initial projection will be updated every week with the team's actual rate of progress. The project's scope is adjusted every week accordingly.
- All stories incorporate verifiable acceptance criteria.
- Stories are captured on a project management tool such as Pivotal Tracker, JIRA, Trello or Github.
- **Progress on individual features/stories/cards would be updated in real time (daily) so stakeholders can watch work in-flight.**

Tests, CI, CD

We write fully automated tests, and believe in **continuous integration, and continuous deployment**.

In addition, in an agile project, the system is constantly being tested as features are released on an ongoing basis.

Once sufficient functionality is ready to be released to production and be used by end users, a formal UAT phase will then conduct an end to end testing of the complete functionality to validate the production readiness of the system. The developer(s) would fix the prioritised issues arising from UAT.



The process involves constantly releasing whatever is built **on a daily basis to a staging environment**, so that the product can be tested, and any feedback from product owners is received early in the development cycle.

In contrast with the waterfall model, this removes the anxiety in not knowing if the product being built is as per expectations.

Documentation

Good code is good documentation, but we acknowledge that it's not enough.

Onboarding new people quickly to the team, and passing on knowledge about the evolution of the software design is as important as understanding the code itself.

We write Architecture Decision Records, READMEs, Swagger documentation and detailed other forms of documentation as applicable.

For other developers, and product owners:

- Architecture Decision Records
- Architecture Diagrams
- Sequence Diagrams

For development workflows:

- README.md on every repository
- CONTRIBUTING.md on every repository
- Services and tools used in the project
- Setting up developer environment
- Deployment architecture

Manuals:

- Operations manual for sys admins
- User manual for data admins, and other users

On Communication

We believe that open lines of communication and planning/projection, which employs strict measurement and reduces guesswork are critical to the success of any project.

Thanks to nilenso's offshore and distributed history, we are very comfortable with most **asynchronous communication** channels including Slack/equivalent, Hangouts/equivalent and email.

The people staffed on the project are responsible for the end-to-end delivery, and will be the primary people to communicate with on timelines, scope changes, staffing requirements, or any other unforeseen issues that come up during the project.

For day-to-day comms on the project, the people staffed will be the PICs.

In case there's a need to talk to people at nilenso outside the project, about other work, or in case any issues arise, we're reachable over email, or a phone call.

Escalation hierarchy:

- Relevant people staffed on the project
- The Tech Lead on the project, if there is one
- Executives at nilenso

Collab. Structure

Modes of operation, and Finances

Modes of Operation

Staff Augmentation:

Many of our clients have an existing in-house engineering team, and work with us to bring on a specific area of expertise like technical leadership, distributed systems on the web, Clojure/Haskell/Elixir, and generally increase their productivity in delivery.

In this mode, our PIC is usually the CTO/Head-of-Engineering/equivalent on the client side, and we essentially operate as an **integral part of the client's team**, participating in meetings with stakeholders, and work on everything that's essential for the project's success.

Examples: Staples, Gojek, Juspay

Product Building:

Sometimes, we're working as the sole tech team (or the sole back-end team if there's mobile work for example), for the client, and take complete ownership of the technical aspects of the project.

There may, or may not be a CTO on the client's side to work with in such cases, and we're comfortable owning the tech in either case. We make large decisions (like the tech-stack) in consultation with the product owners in this case.

Such projects also often involve building an in-house team for the client, and handing off the work to them over a planned period of time, and we're comfortable doing that as well.

Examples: Insider, Simple, Quintype

Billing, and Finances

Standard Billing Terms:

We have worked both on a **time and materials basis and retainer model**. We typically bill on an hourly basis, or the equivalent monthly rate (assuming billing of 120 hours per developer per month on average).

Our standard agreement template can be found here (please request access):
nilenso standard agreement template

We are flexible on contract terms.

Rates:

Our hourly rates are **₹6200/hr per engineer**.

We prefer to staff our projects in pairs, so this works out to a monthly budget of **~₹16L/mo + GST for a pair of developers**. The actual billing is generally ~10-20% lower when you account for time off.

Payment Schedule:

We raise invoices monthly, with payments being due within 15 days of receipt of the invoice.

We have worked with other engagement models as well, and as mentioned earlier, are flexible on contract terms.

Thank You!

If you have any questions, please feel free to drop an email to business@nilenso.com

nilenso