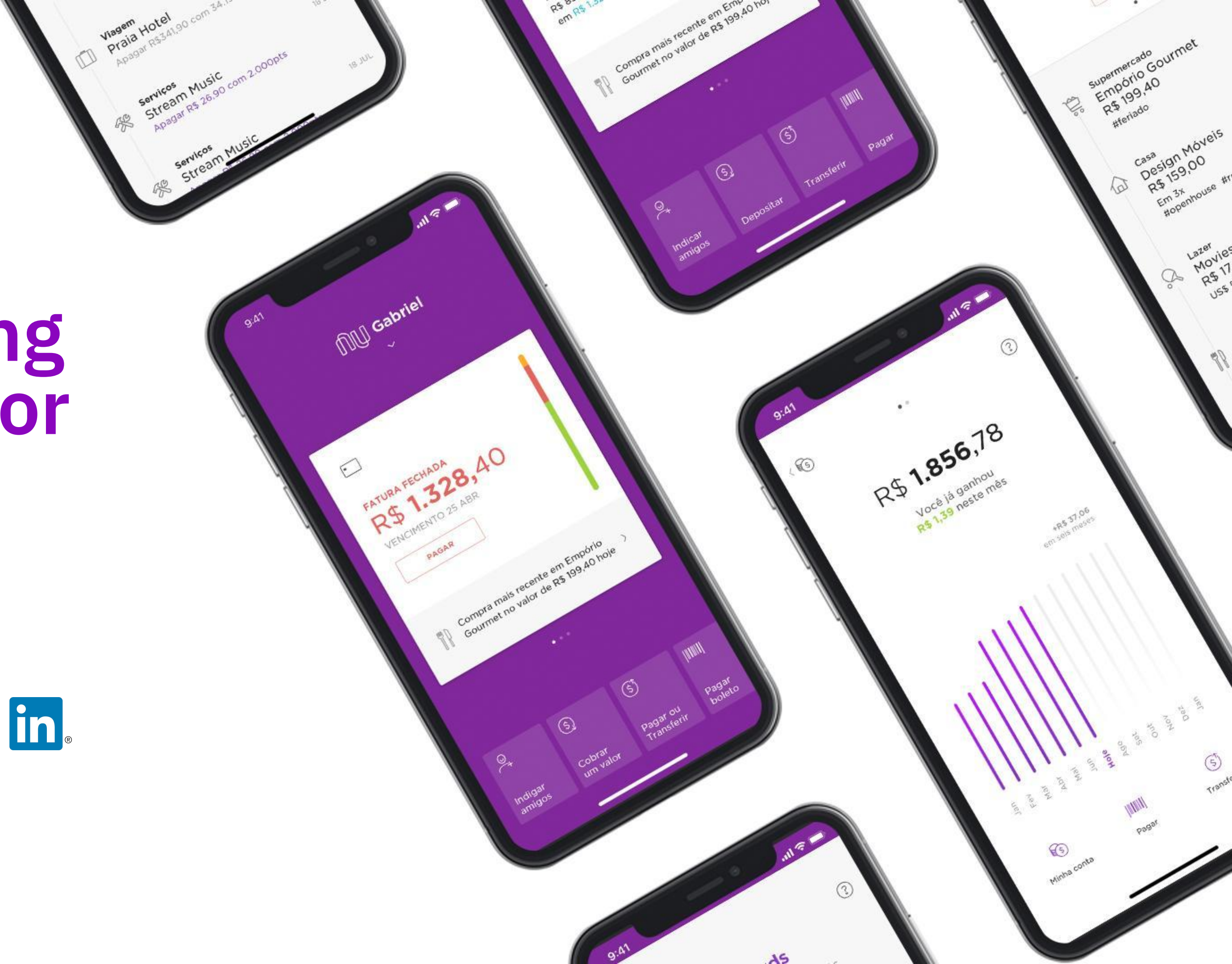
Architecting Software for Leverage

Lucas Cavalcanti

@lucascs

@lucascavalcantisantos



Leverage

noun

FINANCE

the ratio of a company's loan capital (**debt**) to the **value** of its common stock (equity).

verb

use borrowed capital for (an **investment**), expecting the profits made to be greater than the interest payable.

What to take away from this talk

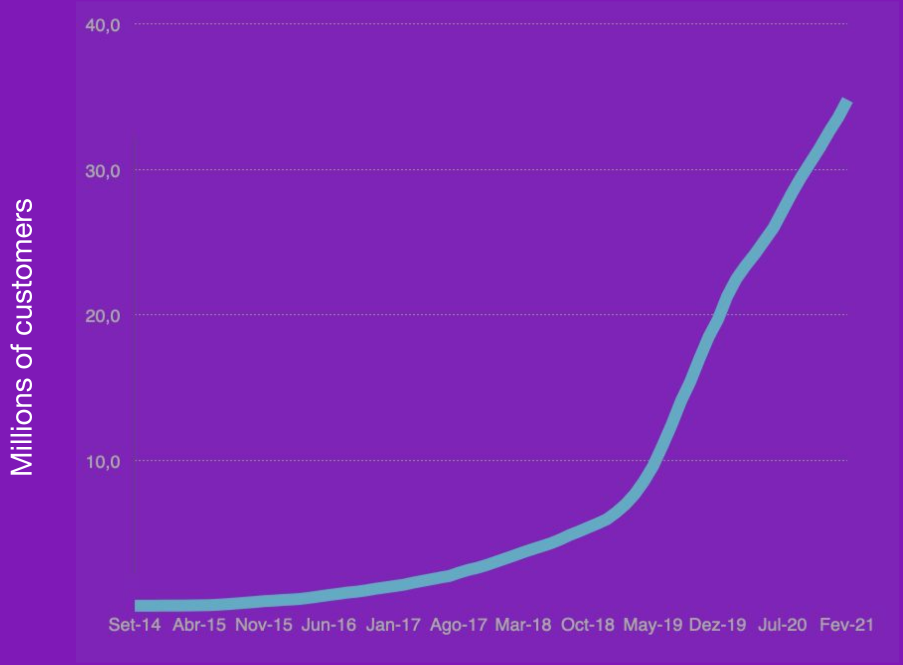
Example of architectural decisions taken during different stages of Nubank's trajectory, aiming at the highest leverage aspect at the time
You may be in a similar position on your current company

Who am I?

Lucas Cavalcanti
Principal Software Engineer
@ Nubank since 2013
Based in São Paulo - Brazil

Nubank is the **leading fintech in Latin America,** born to eliminate complexity and empower customers to take control of their money.

Growing rapidly in a complex domain



35M Customers

1B HTTP Requests/day

1B Kafka Messages/day

10s Deploys/day

700 Microservices

16 Shards

700 Engineers

130 teams

Agenda

Startup time: Time to market and Feedback

Growth time: Resilience and adaptability

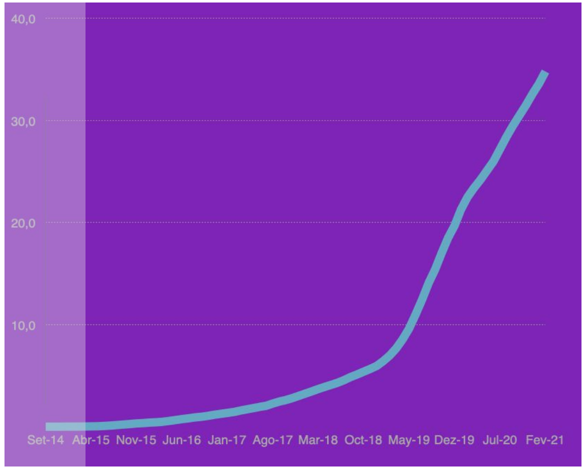
Consolidation time: Reliability and observability

Expansion time: Flexibility and extensibility

Startup time

STARTUP TIME

Late 2013 to early 2015



GREENFIELD PROJECT

That magical moment when you get to choose any technology

FIRST PRODUCT

A digital first credit card with no fees and real time experience on the app

LIMITED RESOURCES

Just few people and a limited cash source to burn

LICENSE DEADLINE

If not operating by May 2014, we'd have to apply for a license with would take up to 2 years to be granted

SMALL "OFFICE"

A little house in a quiet neighborhood in São Paulo

UNKNOWNS

No previous domain knowledge. Uncertain if product would get traction

STARTUP TIME LEVERS

Technology Choices

VALUE: Time to market

LEVERAGE TYPE: Maximizing work not to be done; Containing complexity

✓ DATABASE: DATOMIC

Immutable ledger database of facts
Auditing for free
Updates preserve history
Querying database at any point in time

✓ LANGUAGE: CLOJURE

Runs on the JVM: leverage the whole Java ecosystem
Immutability by default
Simple made easy
Functional Programming close to finance
Hexagonal architecture

✓ MESSAGING: KAFKA

Persistent log of messages (with a TTL)
Ability for resetting offsets so we can reprocess old messages
Strong durability guarantees
Topics partitioned by default

DEBT: Niche and unconsolidated technologies.
Hard to find people with previous experience

Vendors

VALUE: Time to market

LEVERAGE TYPE: Maximizing work not to be done; Build vs Buy

✓ CLOUD: AWS

CloudFormation for deploy automation
DynamoDB for scalable storage for Datomic
Easy to scale services

✓ CREDIT CARD PROCESSOR

Off the shelf solution to run a custom Credit Card
2 months to integrate
MasterCard setup and licensing already done

DEBT: Limited by the processor's ability to scale and respond to problems

STARTUP TIME LEVERS

Practices

VALUE: Fast and early feedback

LEVERAGE TYPE: foundation to build on top of it faster

✓ CI/CD

Github + Pull Requests
Unit and integration tests per service
E2E tests with all services
Baked images (pre-docker)
Test, Staging and Production envs

✓ FAULT TOLERANCE

Rudimentary transient monitoring
Service's health
Immutable infrastructure

✓ MICROSERVICES

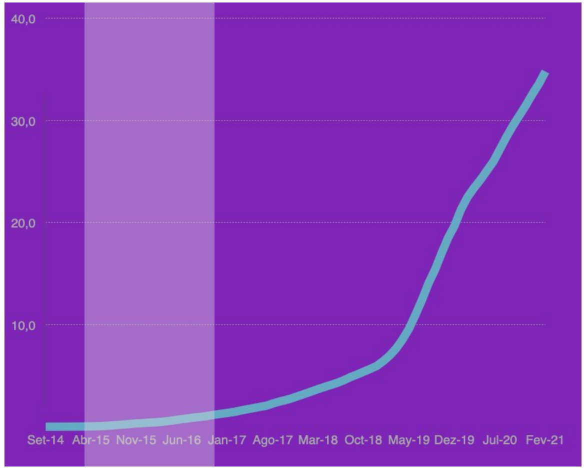
Domain is intrinsically complex
Breaking complexity in smaller services would help contain it

INVESTMENT: Time to build the foundation, which is not always available in startup phases

Growth time

GROWTH TIME

2015 and 2016



FAST GROWTH

Customer base grew way faster than any projection

"OFFICE" NOT SCALING

Moved to a small 3-story office building near Ibirapuera Park in São Paulo which would fit 150 people

VENDOR NOT SCALING

Credit card processor not responding fast enough to the increase in number of customers

TECH NOT SCALING

Scaling horizontally would only get us so far. Bottlenecks started appearing

GROWTH TIME LEVERS

Practices

VALUES: Scalability, Fault tolerance

LEVERAGE TYPE: Avoiding/delaying optimization

✓ INFRASTRUCTURE SHARDING

Sharding the whole infrastructure
Scalability units
Limited blast radius on failures
If shards are small enough, no need for optimization

✓ MORE CI/CD

Frequent automatic deploys
E2e taking up to 1h to run
Replaced e2e with consumer-driven contract tests
Docker images rather than EC2 images

INVESTMENT: year long project to implement and rollout sharding.
Design and creation of a whole new tool

DEBT: project took longer than expected and first shard was bigger than expected and was a special shard for a long time
Each shard has a minimal AWS cost, regardless of number of customers

GROWTH TIME LEVERS

In-housing

VALUES: Resilience, Scalability, Flexibility

LEVERAGE TYPE: Owning your own destiny

✓ PROCESSING IN-HOUSING

Processing CC transactions is the core of the business
Vendor didn't scale as expected
Any change would take months of back and forth to implement

✓ CUSTOMER SUPPORT IN-HOUSING

Providing the best customer support in the industry was key to success
Existing solutions didn't have enough flexibility or were too hard to configure

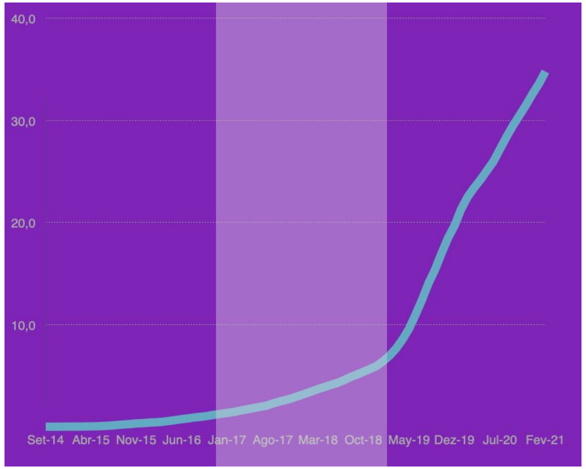
INVESTMENT: 18mo long project to bring each feature in-house and migrate from previous system

DEBT: Long period without any major product changes

Consolidation time

CONSOLIDATION TIME

2017 and 2018



SCALABLE YET UNSTABLE

Although sharding helped a lot on the scaling part, we got to the point where every little corner case would happen, several times

SECOND PRODUCT

With Credit Card consolidating, we moved to a checking account product

OFFICE NOT SCALING, AGAIN

Moved to a 8-story office building near Av. Paulista in São Paulo, which would fit over 1000 people

BIG DATA

Aggregating data from all services and shards became of a huge importance

CONSOLIDATION TIME LEVERS

Technology

VALUES: Scalability, Adaptability, Observability

LEVERAGE TYPE: Ease of infrastructure changes. Cloud tools

✓ KUBERNETES

Ecosystem of infrastructure tools
With high number of services, it scales better than AWS CloudFormation

PROMETHEUS + GRAFANA

Collecting real time metrics
Operational dashboards
Inflow of metrics for other tools, like OpsGenie, Slack, CI/CD canary deploys

INVESTMENT: year long project to setup and migrate shard by shard to k8s

DEBT: constantly hitting AWS limits and spending $$$ with duplicated infrastructure until project was done

CONSOLIDATION TIME LEVERS

Internal Tools

VALUES: Resilience, Observability

LEVERAGE TYPE: Engineering operational productivity

✓ NuCLI

Repository of command line tools evolved by all engineers
Most common operations (like restarting a service or curl'ing with proper credentials) easy at hand

✓ DECLARATIVE INFRA

Repository of declarative resources configuration
Automatic update as soon as change hits the main branch at github

INVESTMENT: dedicated team to curate, maintain and ensure that changes get applied properly

CONSOLIDATION TIME LEVERS

Data

VALUES: Observability, Consistency

LEVERAGE TYPE: Support for pretty much every decision in the company

✓ SCALA + SPARK

Scalable tools to extract data from all service's databases from all the shards, and then compose them into higher level business definitions

✓ ETL

Repository of dataset definitions
Contributions from people of diverse functions in the company
Outputs to data warehouses
Integrated with BI tools
Support for ML

CONSISTENCY CHECKS

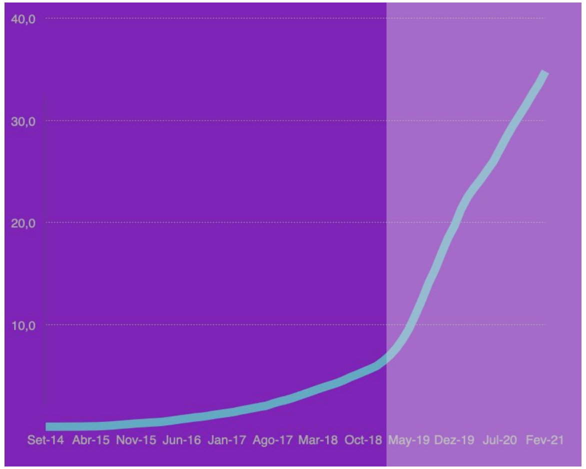
Automated controls that check for inconsistent data and alarm above a threshold
Tool for finding partially completed or failed distributed transactions

INVESTMENT: year long project to create initial versions.
Dedicated team maintain and keep ETL running in a reasonable time
$$$ on AWS to run the ETL process every day

Expansion time

EXPANSION TIME

2019 to present



PRODUCTS FOR EVERYONE

A product offering for everyone that applied

MANY COUNTRIES

Launched products in MX and CO

MANY OFFICES

Offices in BR, DE, MX, CO, AR and US, over 3000 employees

MANY PRODUCTS

Product portfolio growing. Several configurations of existing products

ACQUISITIONS

Acquired Plataformatec, Cognitect and Easyinvest

EXPANSION TIME LEVERS

Horizontal Platforms

VALUES: Extensibility, Productivity

LEVERAGE TYPE: Technology specialist teams building abstracted tools for the others to use

✓ MOBILE + WEB

Flutter + Dart for mobile
ClojureScript + Fulcro for web
NuDS + component libraries
BFFs with GraphQL or REST

✓ INFRASTRUCTURE

Mostly declarative definitions
NuCLI for common operations
All horizontal aspects solved centrally, e.g, fault tolerance, auto scaling, deployment, monitoring

EXPERIMENTATION

Infrastructure to run experiments with feature toggles, A/B testing and monitoring of KPIs

INVESTMENT: dedicated teams creating and maintaining component libraries, tools, documentation and guidelines, with specialists in those technologies so no others need to be specialist to be able to extend and create new things on them

EXPANSION TIME LEVERS

Business Platforms

VALUES: Extensibility, Productivity, Containing complexity

LEVERAGE TYPE: Domain specialist teams building abstracted APIs for the others to use.
Endless possibilities of products built on top of the platforms

✓ BANKING AS A SERVICE

Credit platform to abstract loan issuing, interest generation, reporting and accounting
Future Assets and Payments/Transfers platforms
OpenBanking

✓ INTERNATIONAL CREDIT CARD

Credit Card system broken into platforms for most relevant subdomains: Limits, Billing, Transaction processing, Debts, and others

ACQUISITION

Platform for offering products to customers
Flexible acquisition processes

INVESTMENT: Deep and long design sessions with domain experts to build the long term vision for each business platforms and effort to migrate from legacy systems to the platforms

Recap

Startup time: Delay writing code and build foundation

Growth time: In-housing and sharding

Consolidation time: Infrastructure and data

Expansion time: Horizontal and business platforms

Thanks :)

Lucas Cavalcanti

@lucascs

@lucascavalcantisantos