Evolving with software

Edgars Zvirbulis

I am...

Edgars Zvirbulis

- Software developer at Mintos Marketplace
- Previously 11 years at Tieto
- Graduated from Vidzeme University of Applied Sciences

Mintos Marketplace is...

- Startup company
- Founded in 2014, in Latvia
- Launched in 2015
- Developed from ground up

Mintos Marketplace is...

Mintos in numbers

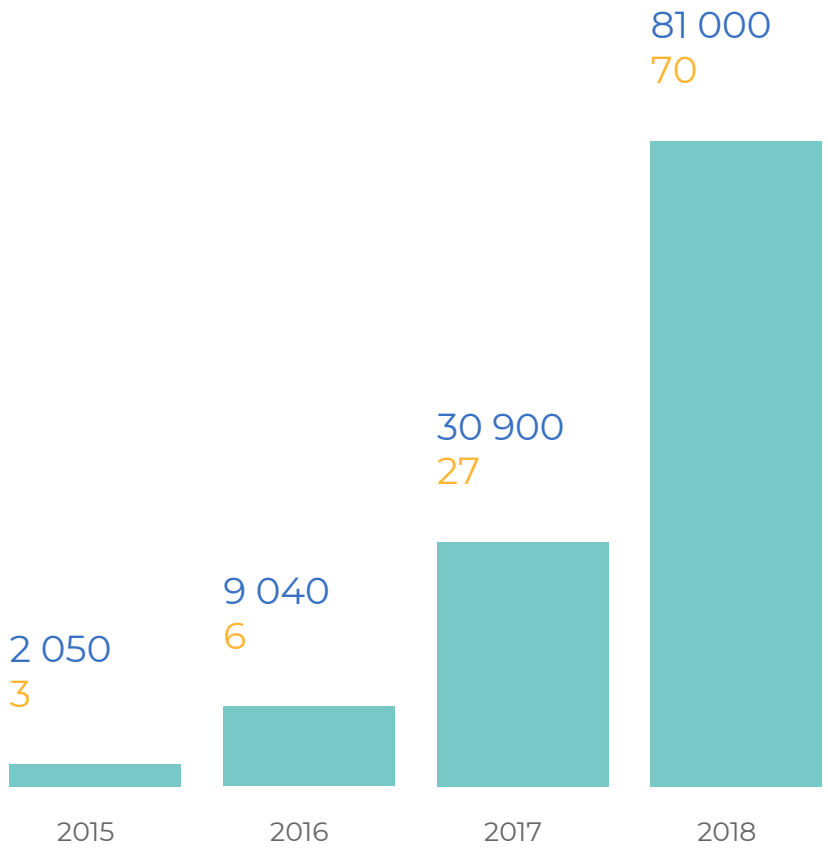
Mintos in numbers

81 000

Investors

70

Countries



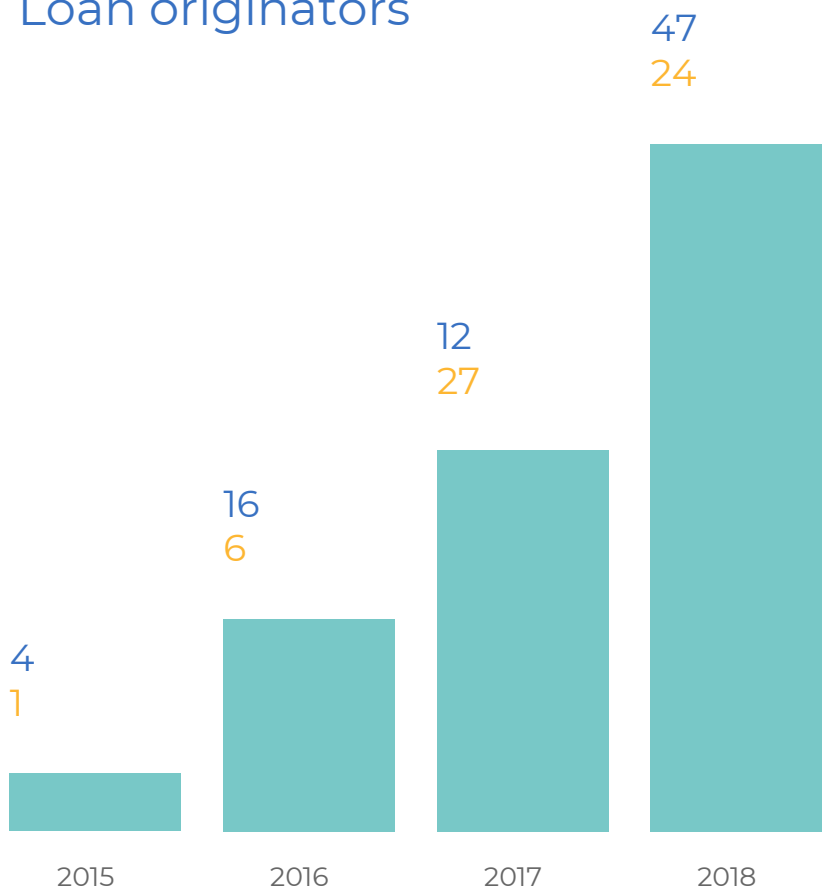
Mintos in numbers

47

Loan originators

24

Countries



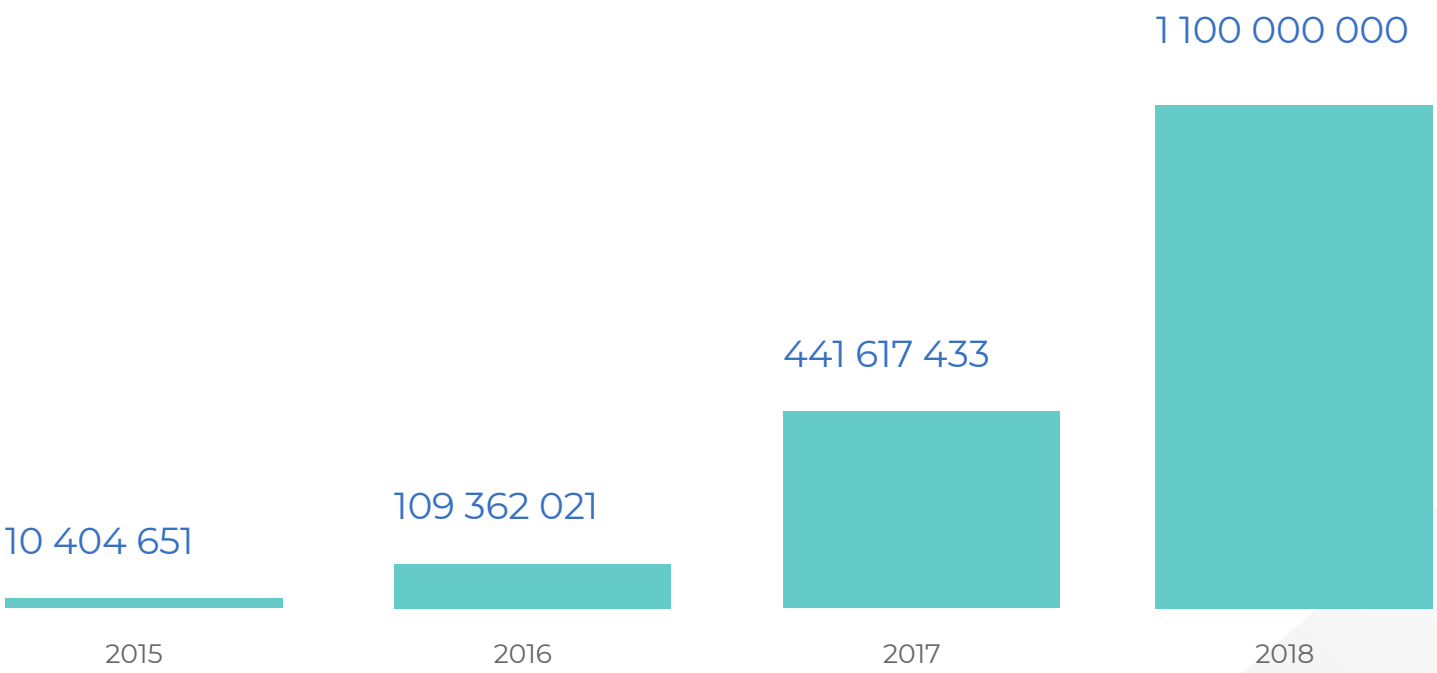
Mintos in numbers

€ 1 100 000 000

Loans funded

1B

End of
August, 2018



Evolution

Evolution

Primarily biological, but the same principles apply to businesses

- constantly ongoing
- somewhat iterative
- changing with every cycle
- responds to changes in the environment
- some “species” become extinct, some new ones appear

At Mintos Evolution is driven by Growth

Growth

code

people

demand

---

data

Code evolution

what we're making

Code in numbers

3.8k source code files, 32.5k commits

- PHP - 72.6%
- HTML - 10.9%
- CSS-9.7%
- Javascript - 4.5%

Code evolution

good code

⇧

no code / spaghetti code

Code evolution

Good code

- readable
- understandable
- maintainable
- easily navigable

Code evolution

Disclaimer: Language choice

- language doesn't matter, it's the code smell that people despise
- bad code can be written in any language, as well as good code

Code evolution

Naming

Naming

Why?

- part of language
- you name it once, live with it forever
- bad naming catches on
- newcomer onboarding

Code evolution

Naming

How?

- speak the business language
- grammar
  ~~mattress~~ -> matters
- be consistent
- decide collectively

Code organization

Code organization

Why?

- same as a human body, building, town, country…
- newcomer onboarding

Code organization

How?

- consistency above all
- business domain separation
- functional responsibility separation
- 3rd party integration separation through abstraction

Refactoring

Refactoring

Why?

- most natural way of evolution
- might help to fix naming problems

Code evolution

Refactoring

How?

- boy scout rule
- pattern recognition
- 1, 2, refactor (WET then DRY)
- needs next slide

Code evolution

Testing

Code evolution

Testing

Why?

- allows refactoring
- safety net
- part of culture
- deals with legacy code

Code evolution

Testing

How?

- automated tests
  - unit
  - functional
  - integration (UI)
- manual (acceptance) tests

Code evolution

Legacy code

Code evolution

Legacy code

- any code that’s already written
- not so bad if naming and code organization done right

Documentation

Code growth

Documentation

- code IS your documentation
- external API is an exception
- guidelines

Human evolution

how we’re making it

Humans in numbers

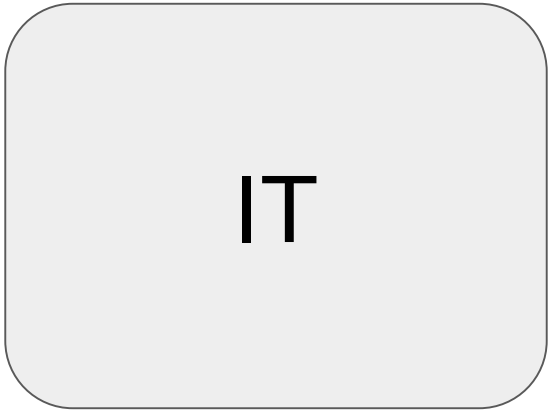
2014 — 4

2018 — **60**

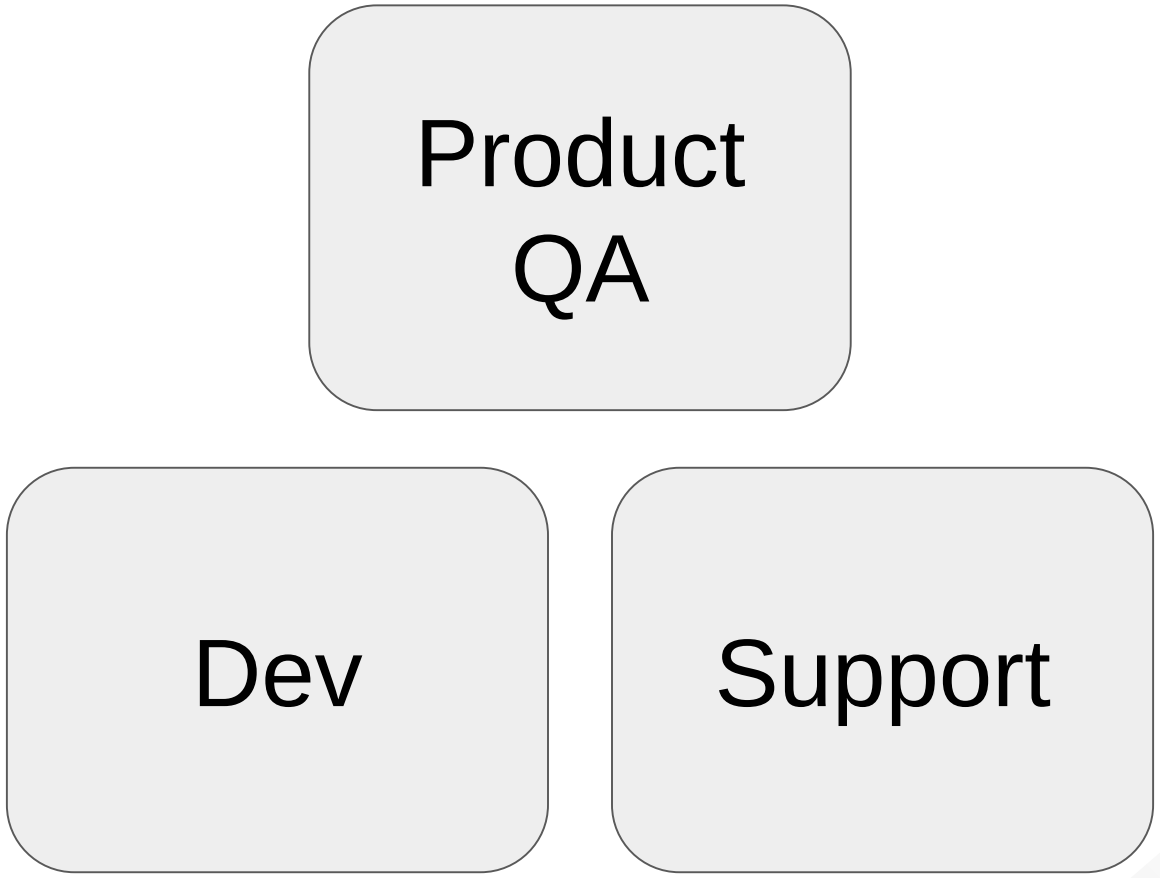
2019 — *2

9 developers + CTO
3 teams
2 QA specialists

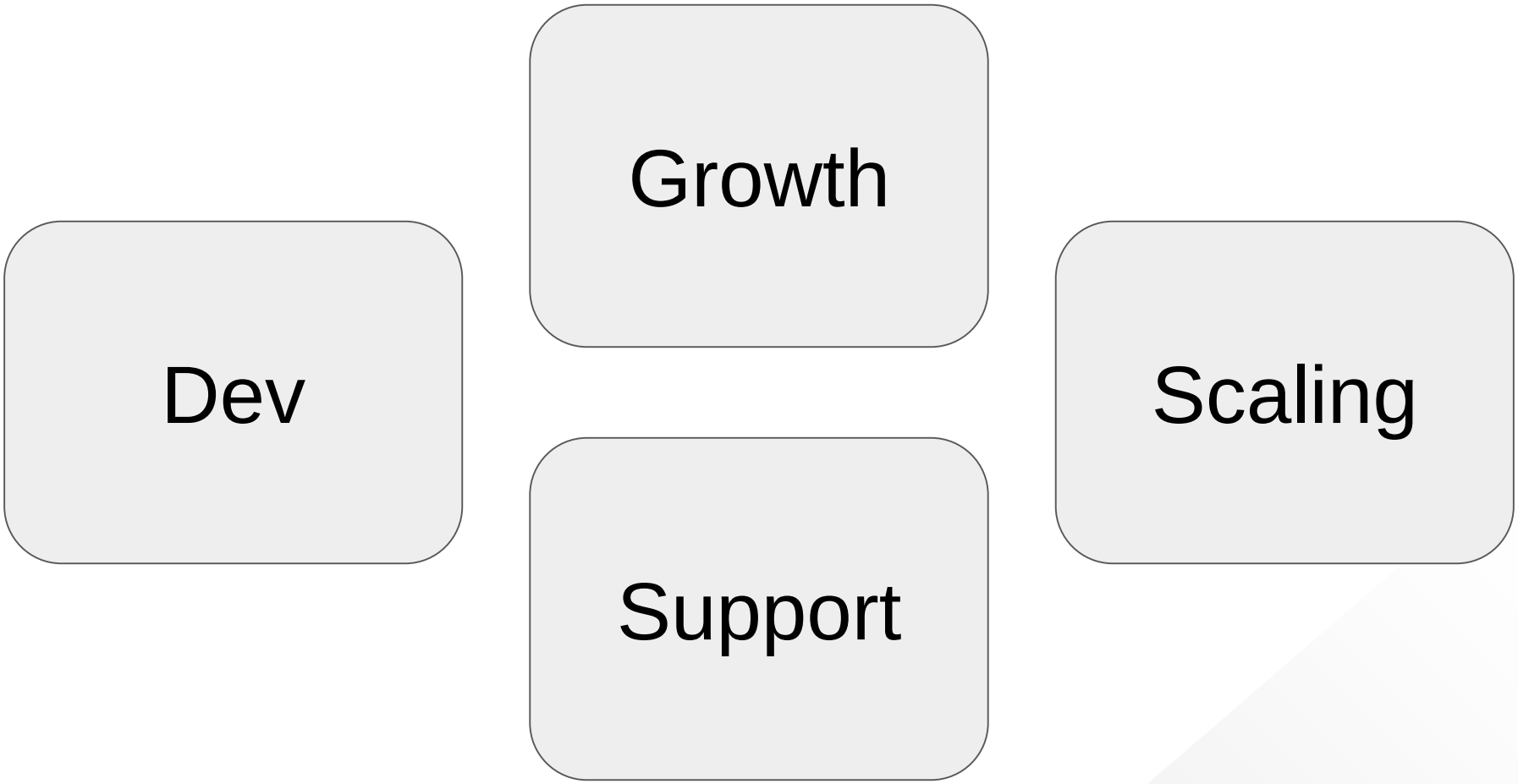
Team evolution



Team evolution



Team evolution



Development process

- Process
  - Scrum + Kanban
- Procedures
  - pull requests
  - mandatory code review by 2 peers
  - automated code style checks
  - automated tests
  - acceptance tests
  - deployment

Developer evolution

Human evolution

Developer evolution

Mindset

- you are making a difference (git diff can prove that)
- everyone is an architect in a way

Human evolution

Developer evolution

Methods

- zoom out - look at the impact
- make mistakes often, learn quickly, tell others what you've learned, supported by:
  - code review
  - automated tests
  - internal communication

Human evolution

Developer evolution

Traits

- patterns over syntax knowledge
- you have to **want** to be better

Company evolution

Human evolution

Company evolution

Caused by more people:

- business-oriented functionality instead of CRUD
- audit
- access right management

Human evolution

Company evolution

Caused by more operations:

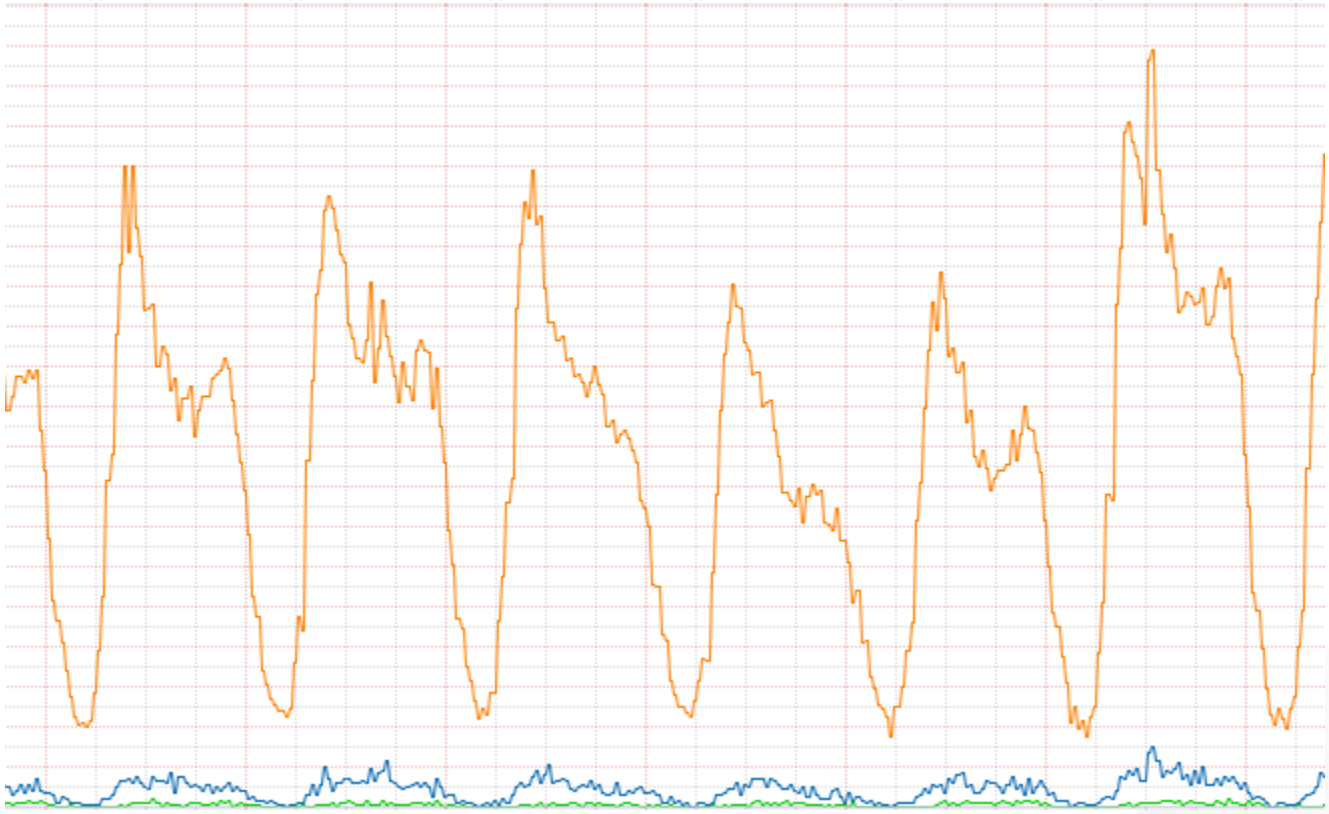
- process automation
- batch processing
- maintenance operations

Growth in Demand

in what circumstances

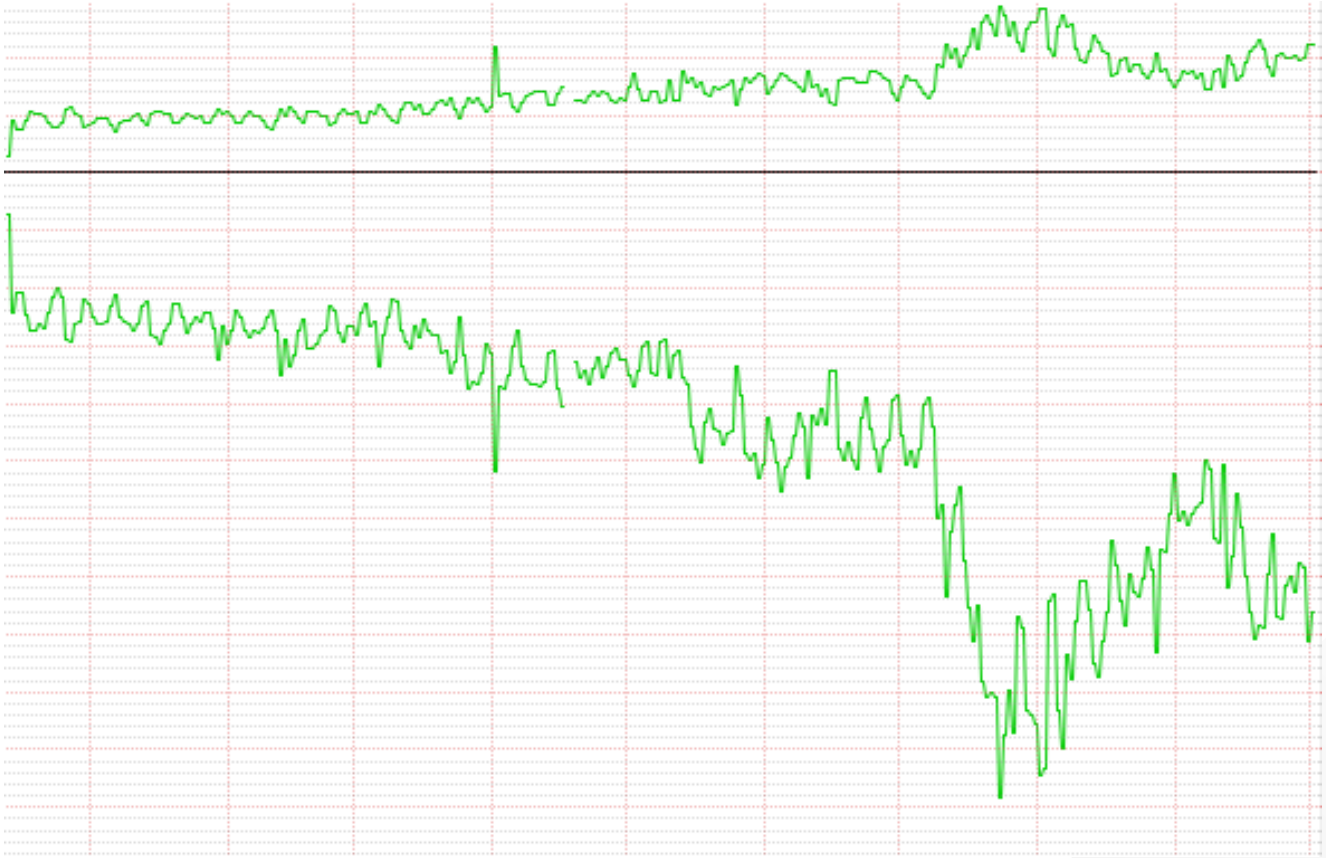
Demand in numbers

Demand on daily basis (1:12 low to high):



Demand in numbers

Traffic over ~1 year (in / out):



Growth in demand

Goal:

System can handle increase in demand without necessity of being rewritten

Growth in demand

Challenge

- solution

Growth in demand

Heavy calculations, latency

- asynchronous processes
- process separation
- queues

Growth in demand

High transaction count

- parallel processing

Growth in demand

Heavy cumulative load

- infrastructure division
- eventually domain separation

Growth in demand

DOS

- firewall
- CDN
- rate limiting

Data growth

what’s the result of it all

Data in numbers

>1.5 TB of data

>4 billion rows in biggest table

Data growth

Today's thousands are tomorrow's billions

- any table can grow
- queries become exponentially slower

"Billions and billions and billions"
/Donald Trump/

Data growth

Blindness

Monitoring

Data growth

Stale data

Archiving

Data growth

Slow calculations

Aggregates

Data growth

Serving UI

Denormalized cache tables

Data growth

High load

Read-only replicas

Data growth

Reporting

Offloading to data warehouse

Data growth

ORM queries *

- good for prototyping
- sub-optimal queries on complex structures
- eventually you grow out of it

* framework specific

To sum up…

Key points

- Growth is unpredictable
- Growth pains are real
- You can’t be proactive enough
- You have to **want** to be better to evolve

Thank you!

www.mintos.com/en/about-us/careers