Moving closer to minimum with Clojure

Robert Pofuk

/about-me

- Co-Founder of AlloraIT

- https://github.com/alpha-prosoft/edd-core
- https://github.com/alpha-prosoft/edd-core-web

- https://github.com/rpofuk
- https://github.com/raiffeisenbankinternational

Agenda

- How we got here?
- From Java to Clojure
- Architecture matters
- Simplicity in production (edd-core)
- Conclusion

How did we get here?

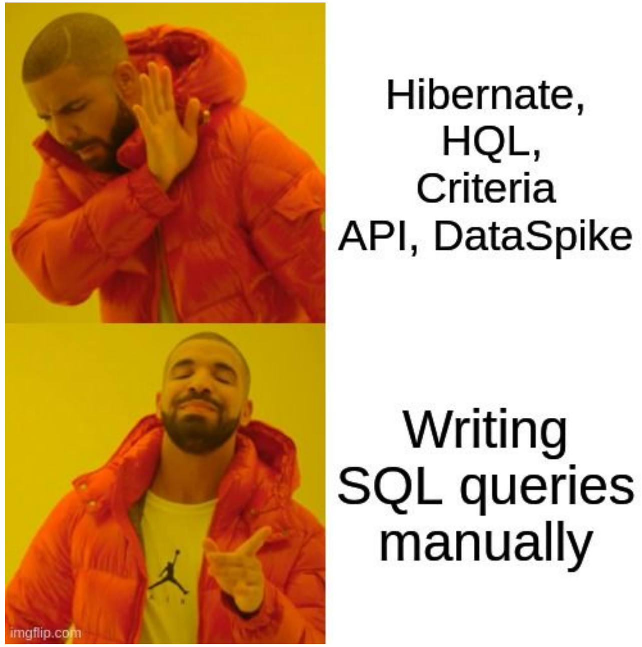
- Confusing “simple” and “easy”
  - Why is everything so complex?
- Frameworks, libraries
  - Horror :)
- Security
  - The US government wants developers to stop using C and C++

From Java to Clojure

- I was always Open Source and standardization enthusiast
  - Using JavaEE, Spring
- How to test
  - Mock all the things
  - Started designing services to be pure (CQRS, Harc
- Clojure
  - Only data and basic things
    - get, assoc, map, reduce, conj, filter, remove…
    - Maps, vectors, lists
- I figured architecture is important

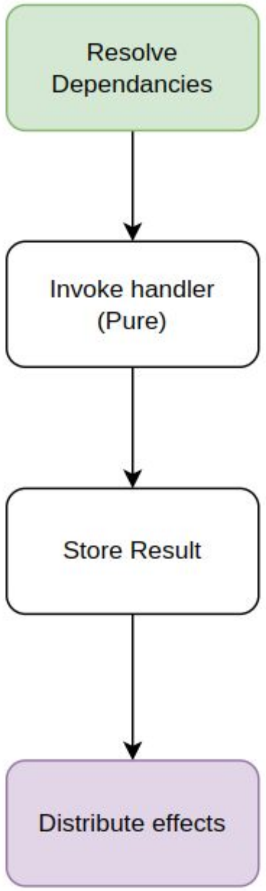
Architecture matters

- Think very very deeply about what you actually need
  - Postgres is now capable of being modest NoSql database
  - Don’t use fancy query libraries
  - Graph Databases, Time Series databases
  - Document generation? HTML
- Microservices
  - Scalability
  - Fancy libraries (i.e. PdfBox)
  - Team
- Design your system more on state transitions then mutation
  - Keep your code pure and testable and it will make persistence layer simpler


Hibernate, HQL, Criteria API, DataSpike
Writing SQL queries manually
imgflip.com

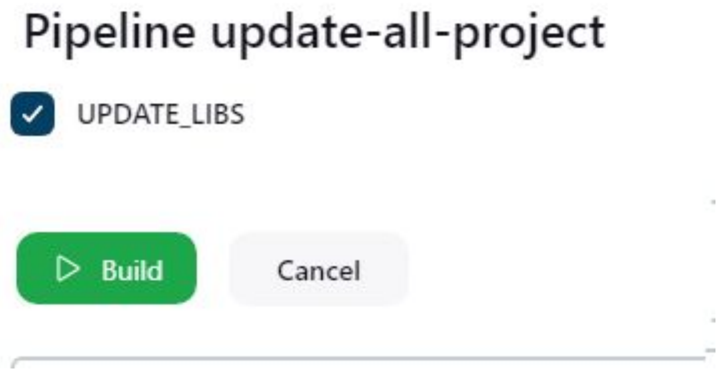
Simplicity in production (edd-core)

- Declarative dependency resolution
  - With combination of CQRS API clients are simple
- Flow:
  - Resolve dependencies
  - Send result to command handler (98% Pure functions)
  - Store output of handler to DB
- All async communication is done via outbox pattern
- Entire system is using same flow
  - Workflow, calculation, document rendering,
- Testing form outside
  - We deploy 10x per day to production in working hours



Dependencies & Security

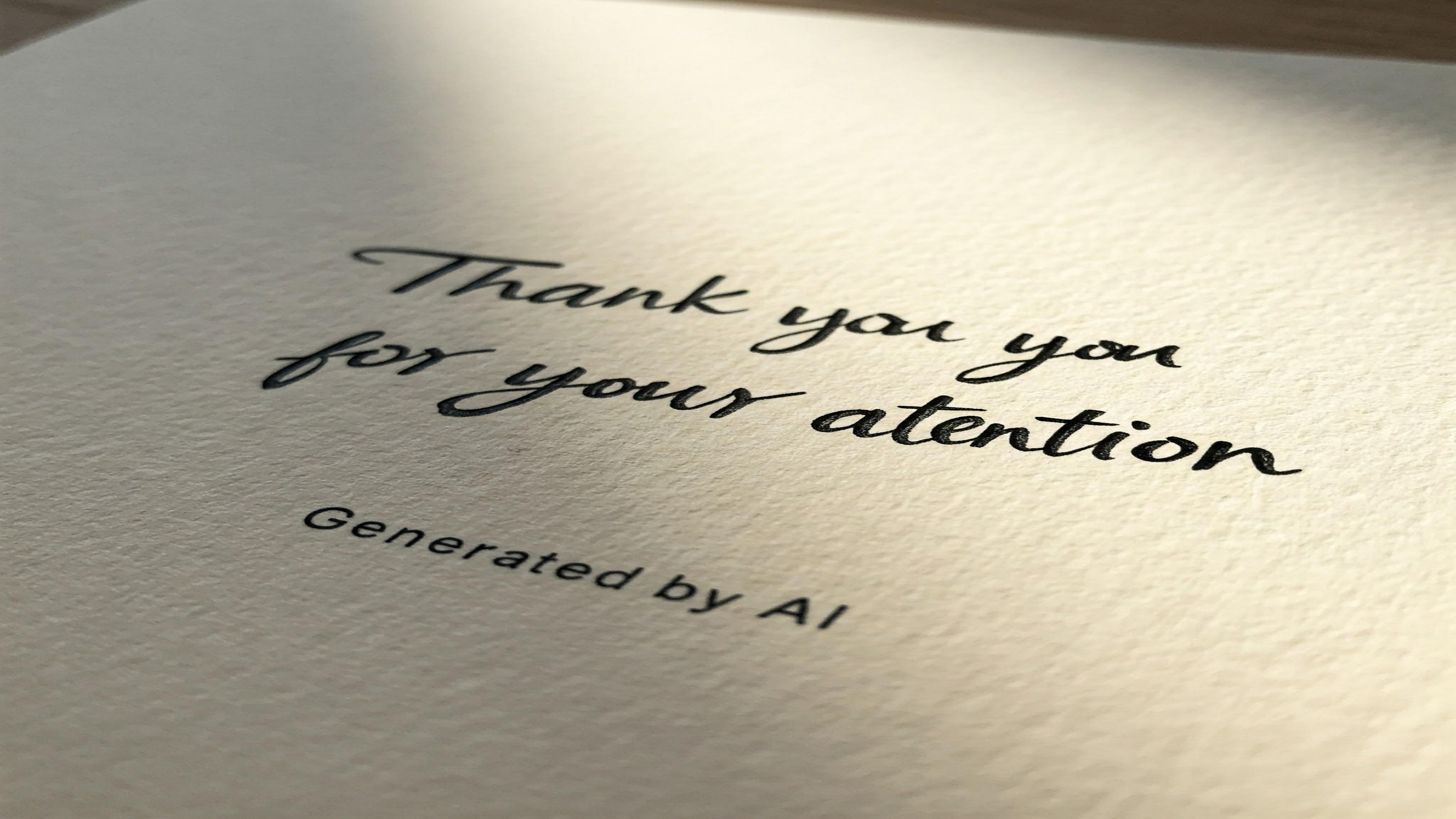
- We have handful dependencies
  - And most of them we forked already and make our own build (HikariCP, Jsonista)
- Most of dependencies are Clojure wrapper around Java
  - They have no dependencies, just JVM
- We scrutinize every single addition (Whitelisting)
  - It is incredible how people take lightly adding new dependencies
  - Used to believe whitelisting is impossible
- Jobs that update entire system (Testing)


Pipeline update-all-project

✓ UPDATE_LIBS

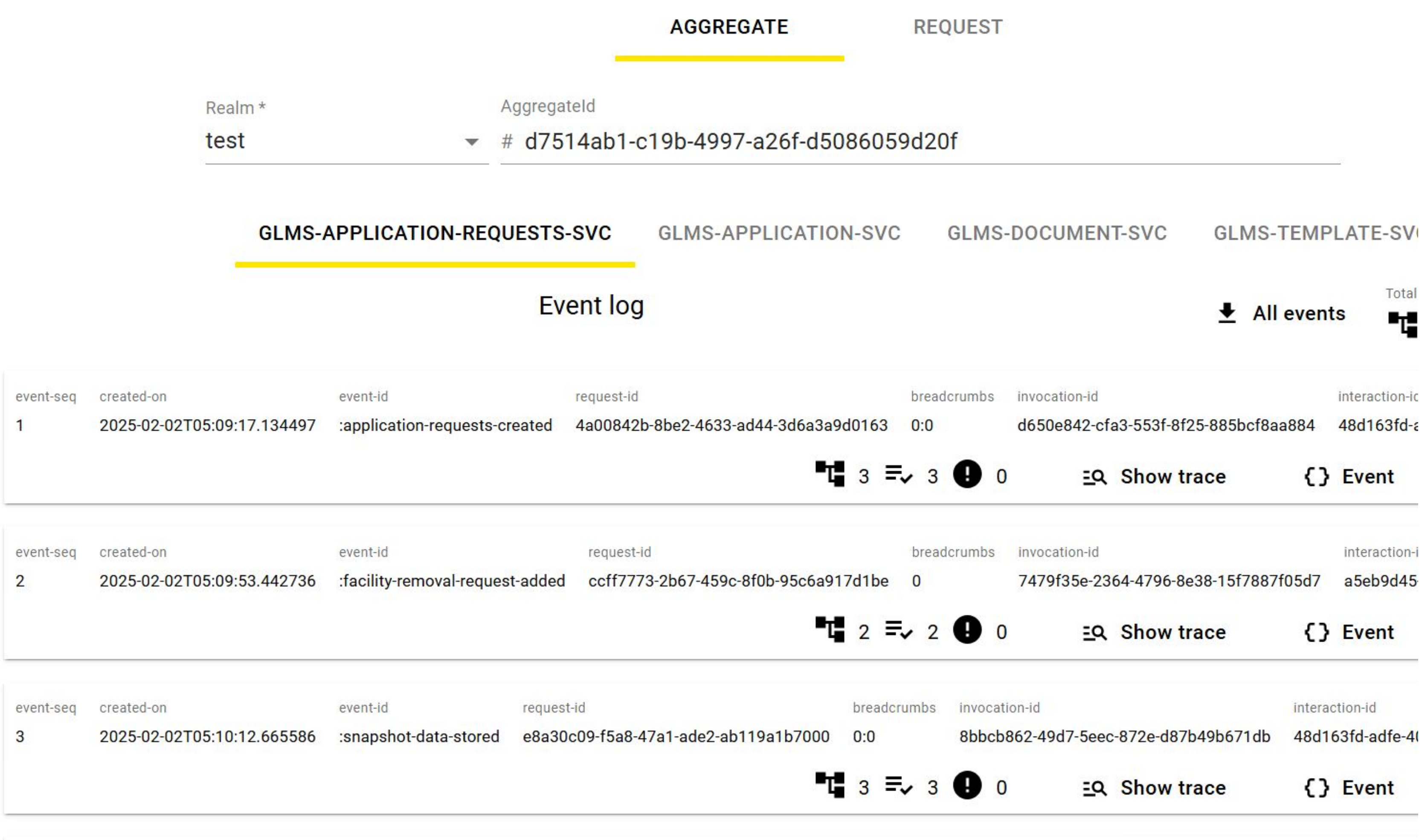
Conclusion

- Use what language offers
  - Java http client vs Apache Http client
- Stick to basic things
  - Don't abstract and hide complexity behind frameworks (i.e. Spring batch vs Pure Java)
- Design architecture to support simplicity
  - Denormalized data instead of complex query magic and DSLs
  - I.e. Store JSON instead of using hibernate
- Microservices
  - Isolate things that need special dependencies and have tools to update things automatically (Pipelines “update all projects”)
- Testing
  - Even if you do not need to release daily make sure you can
  - Only way you can keep system updated and secure



Thank you you
for your atention

Generated by AI



AGGREGATE   REQUEST

Realm *
test

AggregateId
# d7514ab1-c19b-4997-a26f-d5086059d20f

GLMS-APPLICATION-REQUESTS-SVC   GLMS-APPLICATION-SVC   GLMS-DOCUMENT-SVC   GLMS-TEMPLATE-SV

Event log

All events   Total

event-seq	created-on	event-id	request-id	breadcrumbs	invocation-id	interaction-ic
1	2025-02-02T05:09:17.134497	:application-requests-created	4a00842b-8be2-4633-ad44-3d6a3a9d0163	0:0	d650e842-cfa3-553f-8f25-885bcf8aa884	48d163fd-a

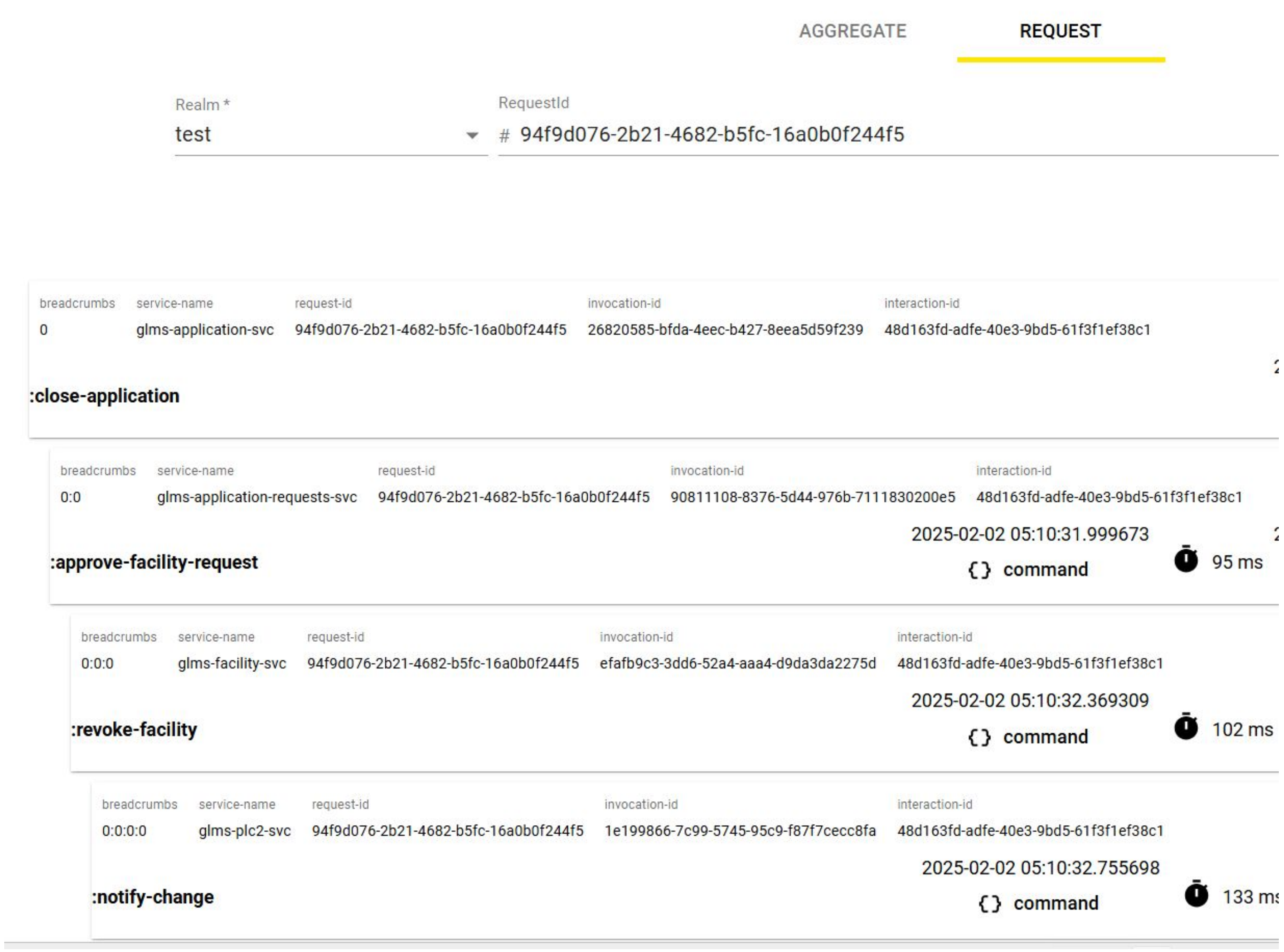
3   3   0   Show trace   { } Event

event-seq	created-on	event-id	request-id	breadcrumbs	invocation-id	interaction-i
2	2025-02-02T05:09:53.442736	:facility-removal-request-added	ccff7773-2b67-459c-8f0b-95c6a917d1be	0	7479f35e-2364-4796-8e38-15f7887f05d7	a5eb9d45

2   2   0   Show trace   { } Event

event-seq	created-on	event-id	request-id	breadcrumbs	invocation-id	interaction-id
3	2025-02-02T05:10:12.665586	:snapshot-data-stored	e8a30c09-f5a8-47a1-ade2-ab119a1b7000	0:0	8bbcb862-49d7-5eec-872e-d87b49b671db	48d163fd-adfe-4

3   3   0   Show trace   { } Event


AGGREGATE REQUEST

Realm *
test

RequestId
94f9d076-2b21-4682-b5fc-16a0b0f244f5

breadcrumbs	service-name	request-id	invocation-id	interaction-id
0	glms-application-svc	94f9d076-2b21-4682-b5fc-16a0b0f244f5	26820585-bfda-4eec-b427-8eea5d59f239	48d163fd-adfe-40e3-9bd5-61f3f1ef38c1

2

:close-application

breadcrumbs	service-name	request-id	invocation-id	interaction-id
0:0	glms-application-requests-svc	94f9d076-2b21-4682-b5fc-16a0b0f244f5	90811108-8376-5d44-976b-7111830200e5	48d163fd-adfe-40e3-9bd5-61f3f1ef38c1

2025-02-02 05:10:31.999673 2

:approve-facility-request

{ } command 95 ms

breadcrumbs	service-name	request-id	invocation-id	interaction-id
0:0:0	glms-facility-svc	94f9d076-2b21-4682-b5fc-16a0b0f244f5	efafb9c3-3dd6-52a4-aaa4-d9da3da2275d	48d163fd-adfe-40e3-9bd5-61f3f1ef38c1

2025-02-02 05:10:32.369309

:revoke-facility

{ } command 102 ms

breadcrumbs	service-name	request-id	invocation-id	interaction-id
0:0:0:0	glms-plc2-svc	94f9d076-2b21-4682-b5fc-16a0b0f244f5	1e199866-7c99-5745-95c9-f87f7cecc8fa	48d163fd-adfe-40e3-9bd5-61f3f1ef38c1

2025-02-02 05:10:32.755698

:notify-change

{ } command 133 m

What is still wrong

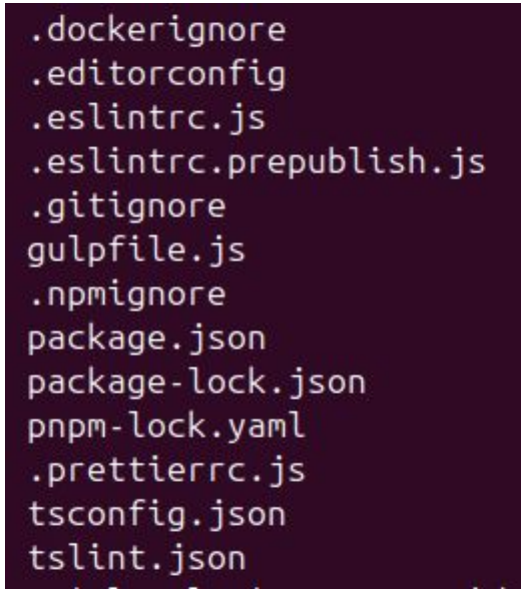
- People want to use different technologies
  - Seems like just for sake of using them
  - Finding edge case that something else will be better suited for problem does not justify introducing new technology
- It is hard to find people to support change
  - People understand what I’m talking about but then they fallback to same regular things
- Clojure
  - Some small things missing in core
    - Built in advanced schema validation (i.e. malli like thing)
    - Json

CQRS

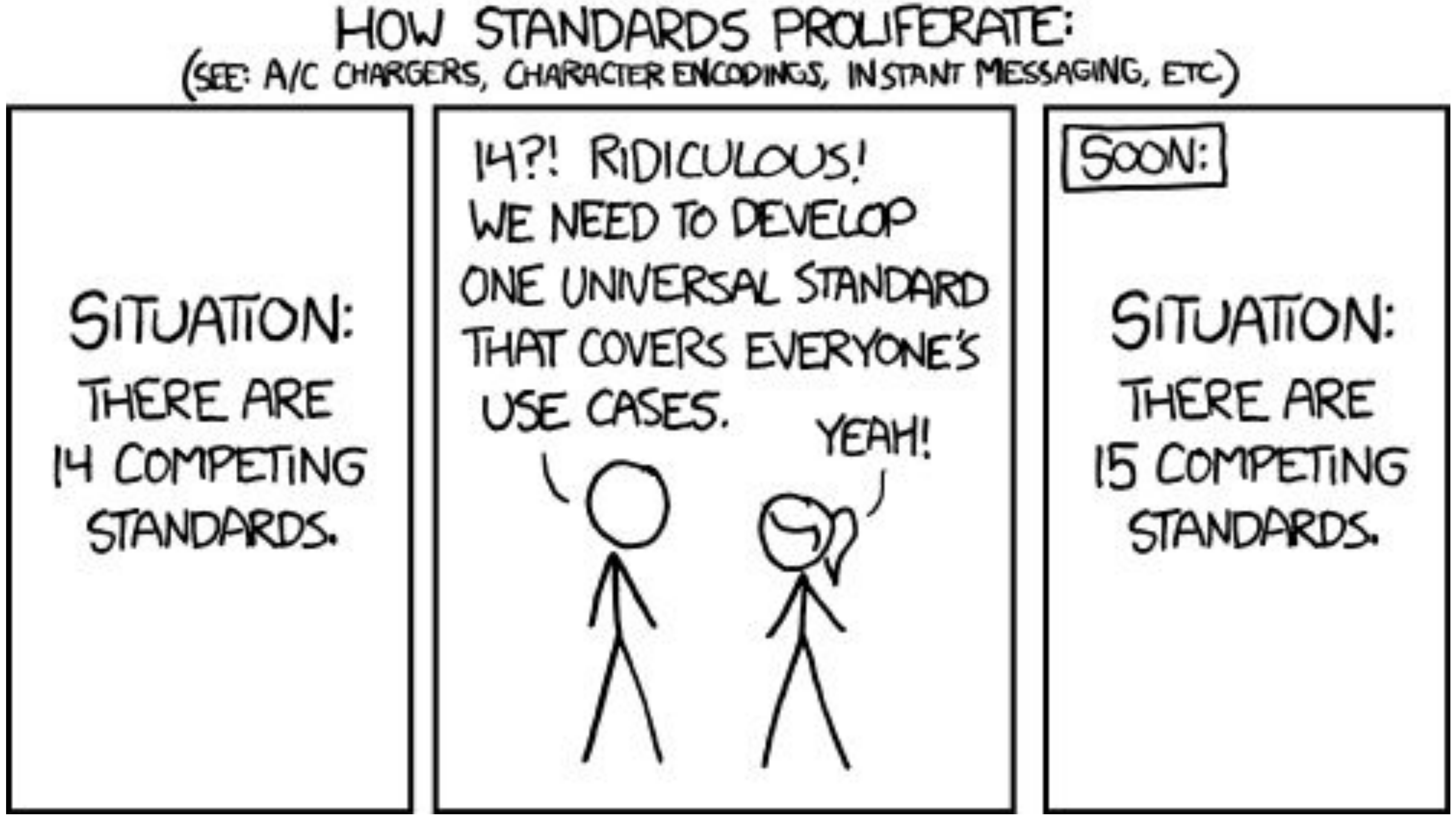
- CQRS stands for Command and Query Responsibility Segregation
  - https://www.youtube.com/watch?v=qDNPQo9UmJA
- Frontend client implementation is simple ~300 lines of code
  - https://github.com/raiffeisenbankinternational/edd-core-web/blob/master/src/edd/client.cljs
- We have 1 API gateway for entire system
  - No fancy annotations, not annotation processor, filters…
  - Just simple routing
- Store requests in db
  - Storing entire requires easy

Frontend?

- It is good and bad
    - npm, yarn, pnpm, corepack, gulp, Grunt
    - Webpack, google compiler
    - React, Angular, Vue, Svelte
    - Selenium, Cypress, Puppeteer
- I have feeling that none of the tools are either abandoned or maintained
- We use MaterialUI/React with re-frame (And couple small libs)
    - Updating is hard (Breaking changes, compatibility, dependencies…)
- Will it event become better?
    - Unify tool on global?



```
.dockerignore
.editorconfig
.eslintrc.js
.eslintrc.prepublish.js
.gitignore
gulpfile.js
.npmignore
package.json
package-lock.json
pnpm-lock.yaml
.prettierrc.js
tsconfig.json
tslint.json
```


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Source: https://xkcd.com/927/