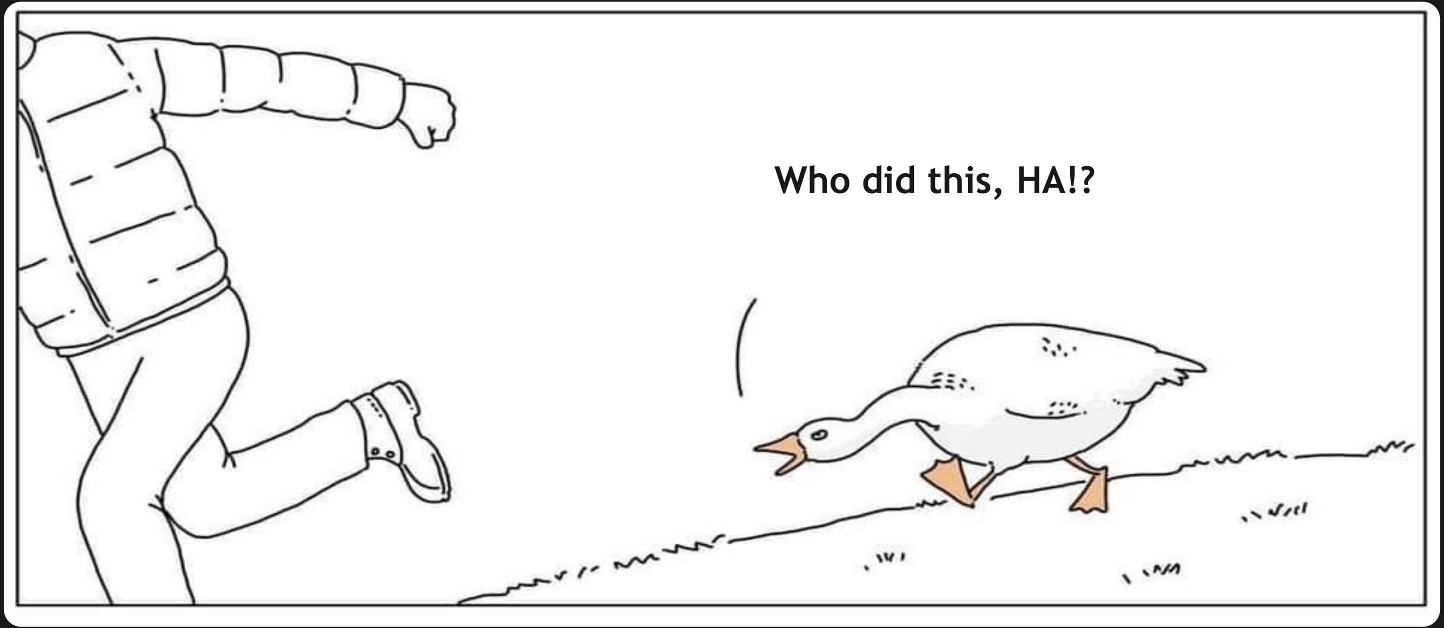
Vadim Tsaregorodtsev • ETG

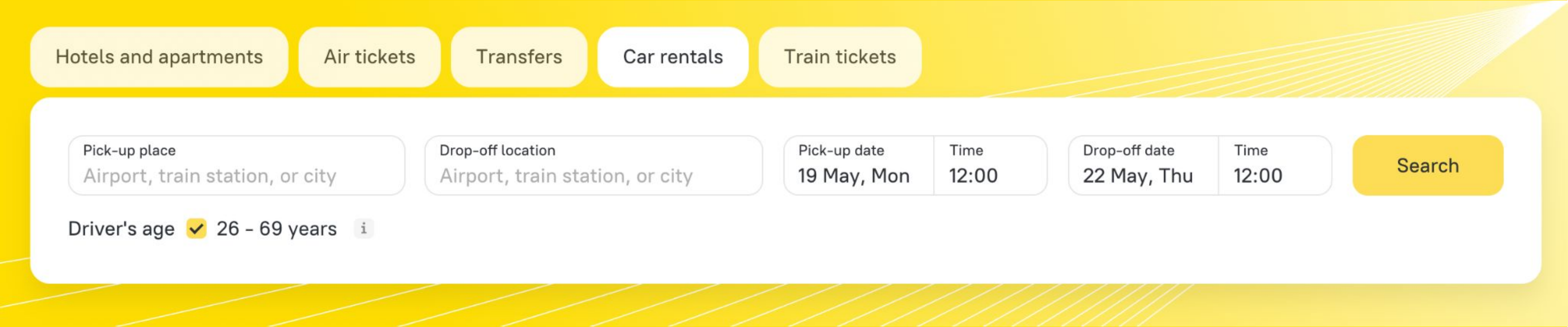
Observability Passport: Navigating the What, Where, and When of Your Frontend App

Vadim Tsaregorodtsev

Frontend Guild Lead
Transportation Products
FrontOps, DevEx

Something broke





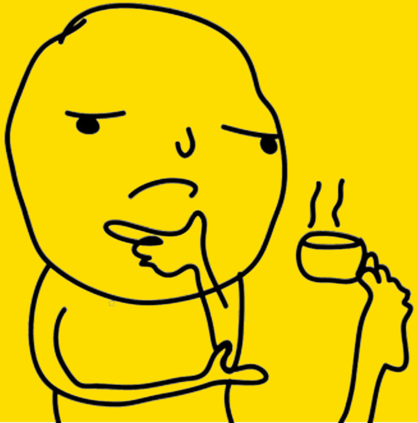
Pick-up place: Airport, train station, or city

Drop-off location: Airport, train station, or city

Pick-up date: 19 May, Mon — Time: 12:00

Drop-off date: 22 May, Thu — Time: 12:00

Driver's age ☑ 26 - 69 years



How can we really know that a product works and works properly?

Three Pillars of Observability

Metrics

Traces

Logs

In this presentation you learn...

- Monitoring and Observability
- Metrics and their types
- Pure technical metrics
- SLA, SLO, SLI
- Observability Passport

Metrics

Technical

- CPU usage
- RAM usage
- API response time

Business

- Conversion
- Retention
- Revenue

Hybrid metrics

Web Vitals

- FCP
- LCP
- TTFB

Error Rate

- 500
- 499

SLA
SLO
SLI

~~Schni Schna Schnappi...~~

Service **L**evel **A**greement

SLO

SLI

~~Schni Schna Schnappi...~~

Service **L**evel **A**greement

Service **L**evel **O**bjective

SLI

~~Schni Schna Schnappi...~~

Service **L**evel **A**greement

Service **L**evel **O**bjective

Service **L**evel **I**ndicator

~~Schni Schna Schnappi…~~

How do these metrics help to solve problems?

- Evaluate the current service level
- Help prioritize tasks
- Help communicate with business

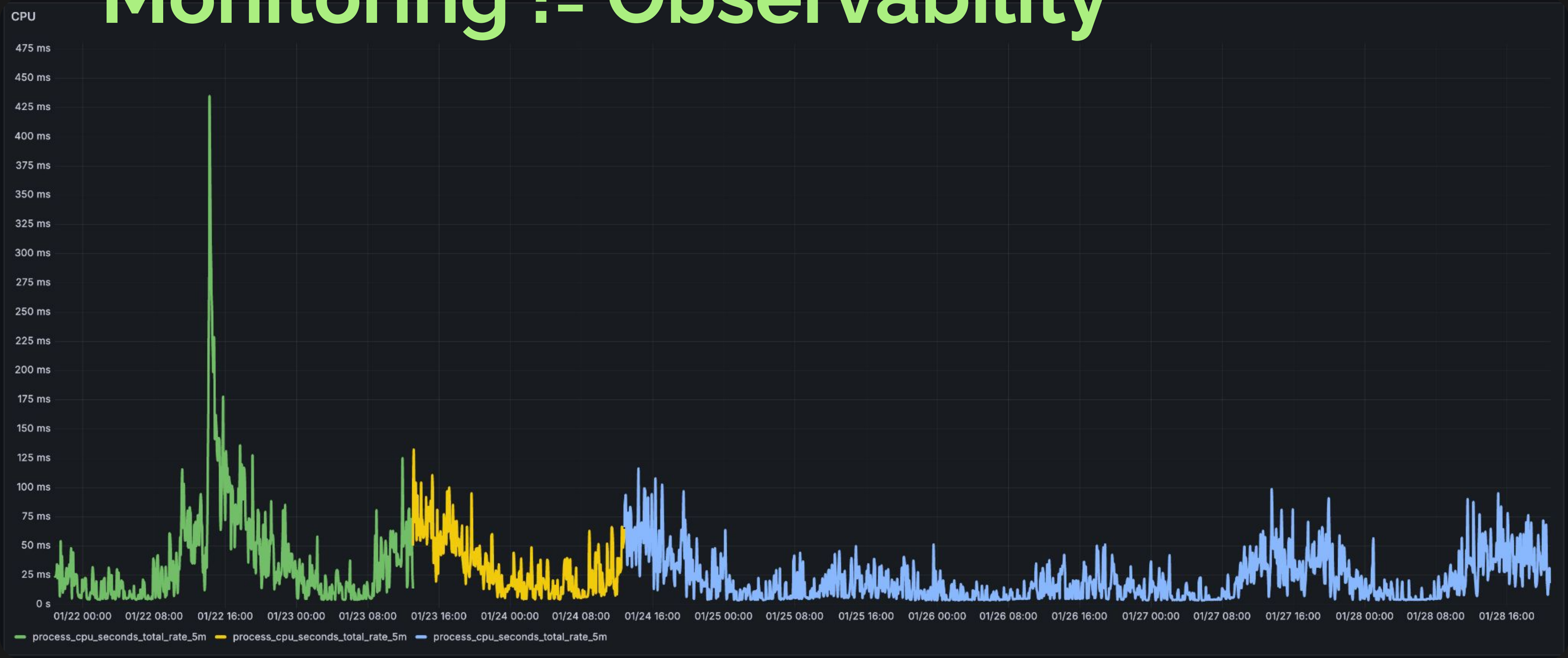
Error Budget

$$EB = 100\% - 99.(9)\%$$

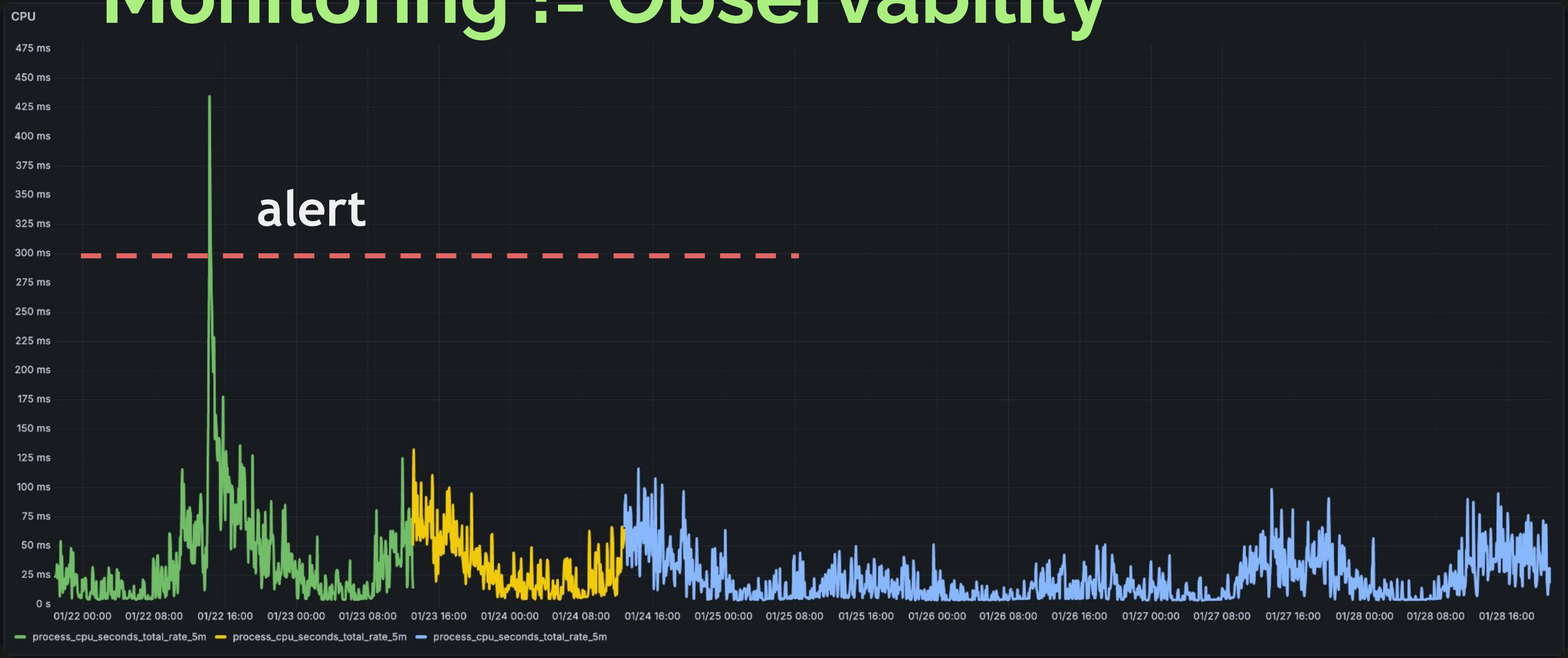
What does Error Budget affect?

- Determines priorities
- Simplifies decision making
- Helps find balance

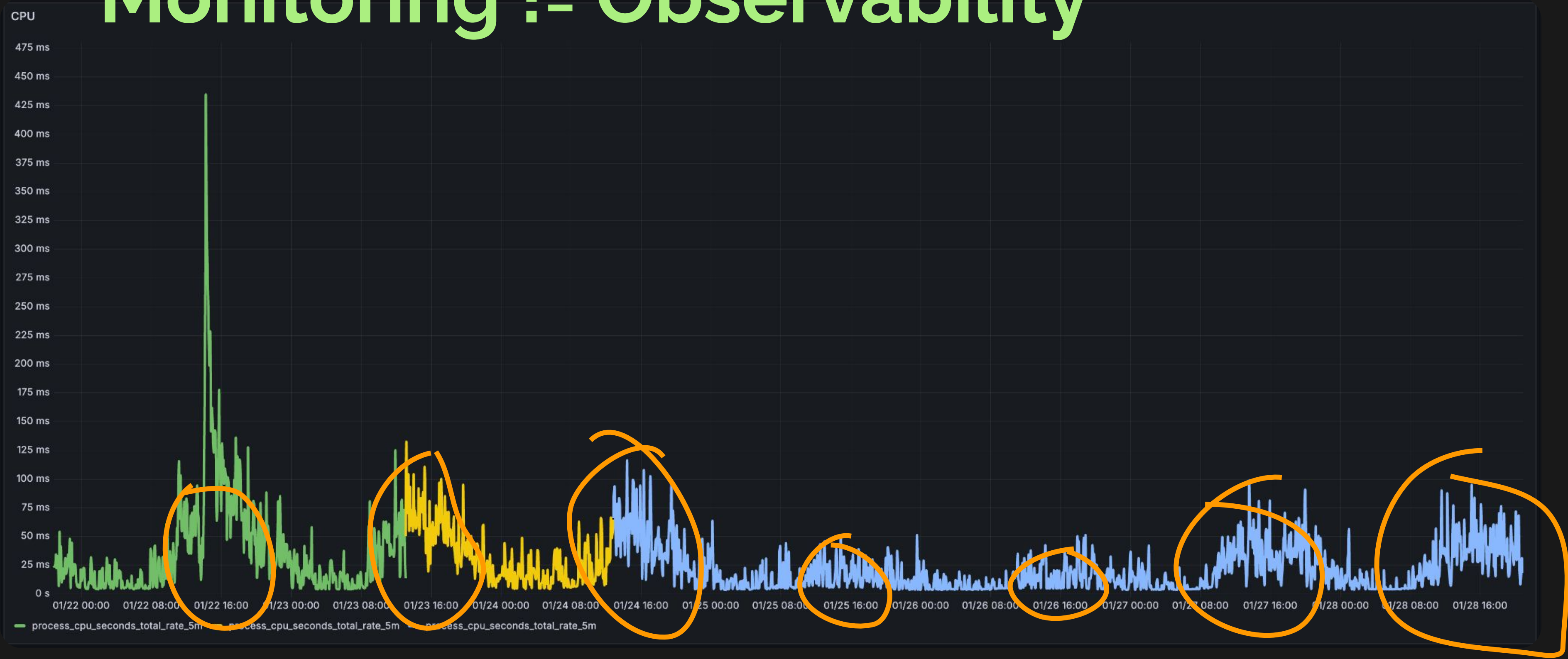
Monitoring != Observability



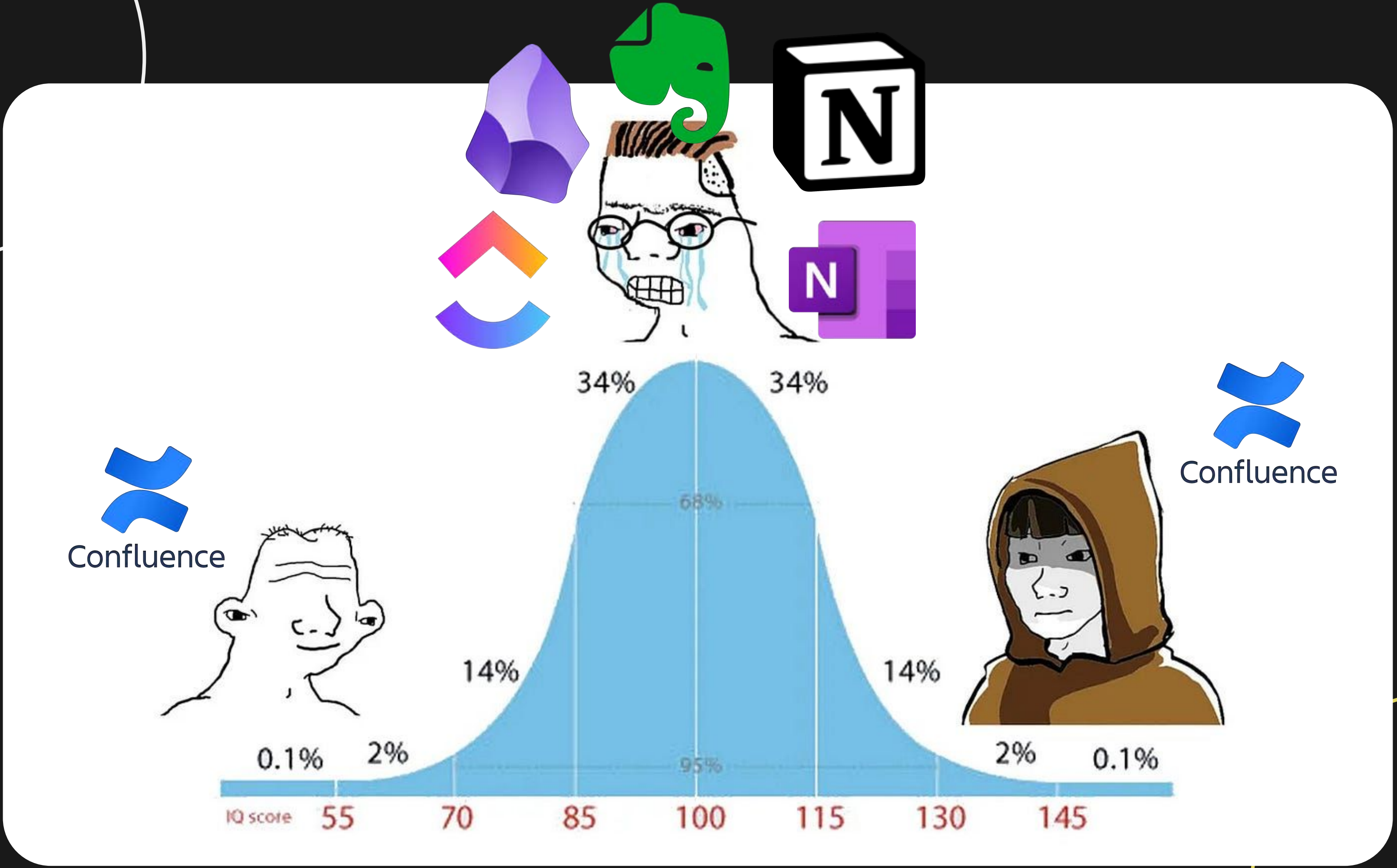
Monitoring != Observability



Monitoring != Observability

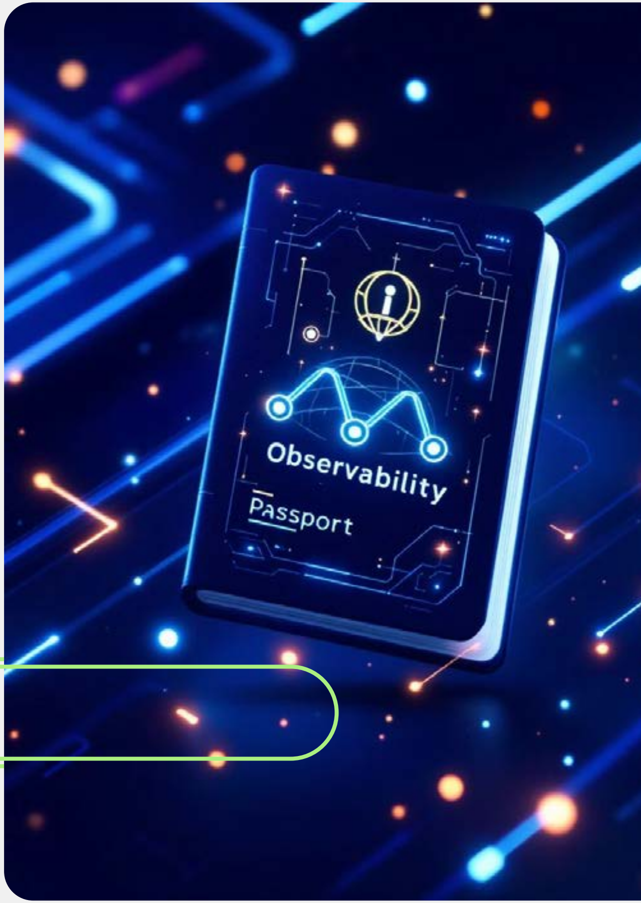


How can we really know that a product works and works properly?


Confluence
34%
34%
68%
Confluence
14%
14%
0.1%
2%
95%
2%
0.1%
IQ score
55
70
85
100
115
130
145

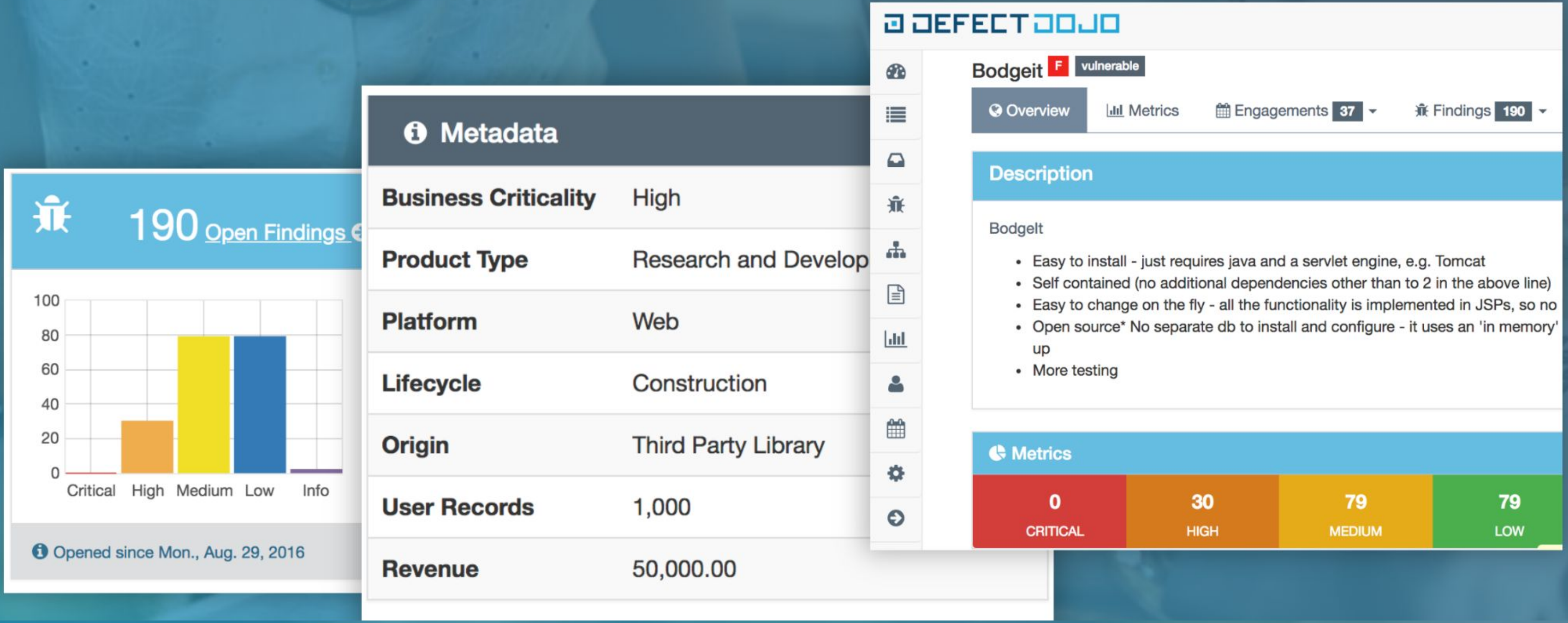
What's in my passport?

- Error tracker (**Sentry**)



How to analyze code?

- Linting
- Testing
- Scanning



190 Open Findings

Opened since Mon., Aug. 29, 2016

Metadata

Business Criticality	High
Product Type	Research and Develop
Platform	Web
Lifecycle	Construction
Origin	Third Party Library
User Records	1,000
Revenue	50,000.00

DEFECTDOJO

Bodgeit F vulnerable

Description

BodgeIt

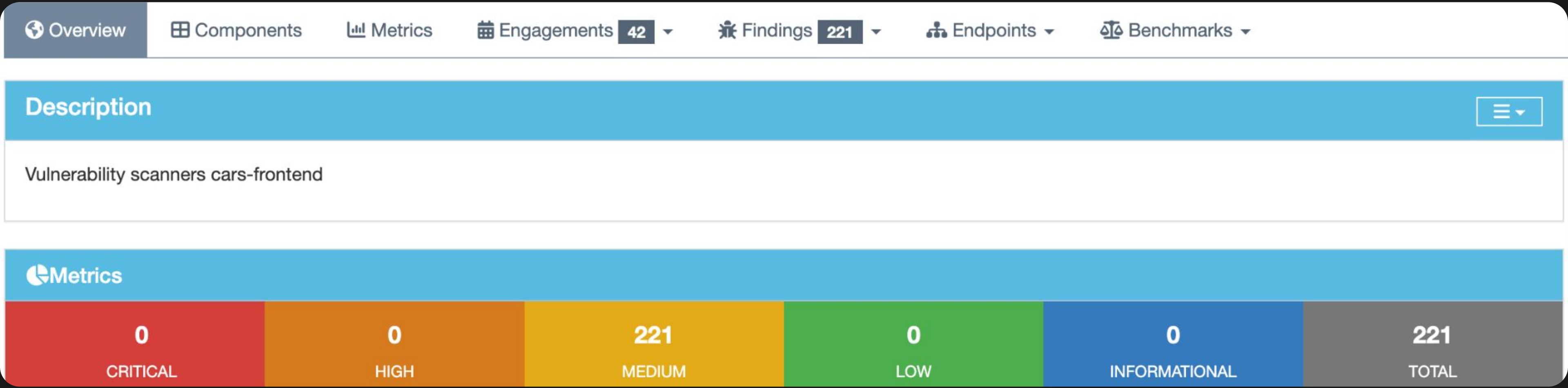
- Easy to install - just requires java and a servlet engine, e.g. Tomcat
- Self contained (no additional dependencies other than to 2 in the above line)
- Easy to change on the fly - all the functionality is implemented in JSPs, so no
- Open source* No separate db to install and configure - it uses an 'in memory' up
- More testing

Metrics

0	30	79	79
CRITICAL	HIGH	MEDIUM	LOW

SLO = 0

Number of critical vulnerabilities in production. If the number is greater than 0, nothing new is rolled out.



Description

Vulnerability scanners cars-frontend

Metrics

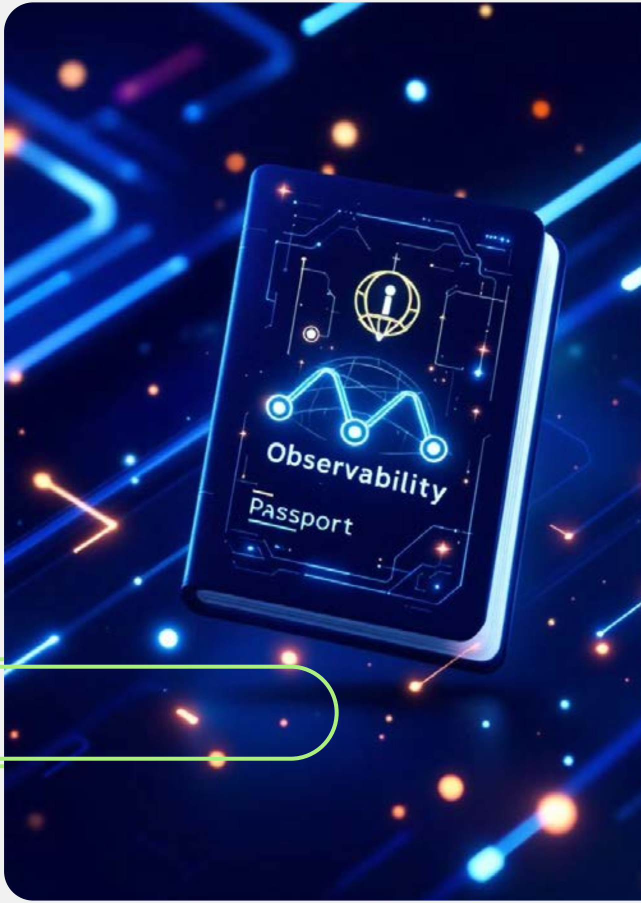
CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL	TOTAL
0	0	221	0	0	221

What’s common in all SLA’s?

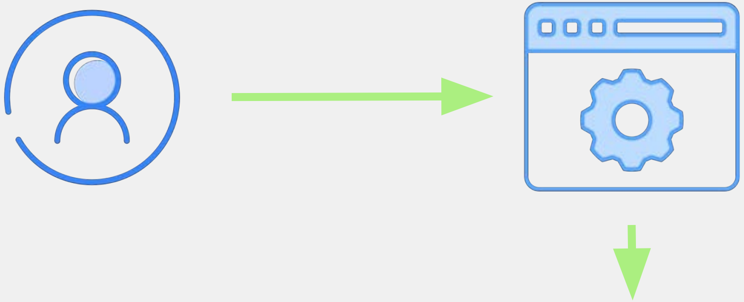
- Identify areas to monitor
- Define - what is an error?
- Set an error budget
- Collect error/metric data
- Visualize data in dashboards
- Set up alerts as needed
- Assign owners and define response protocols for incidents

What's in my passport?

- Error tracker (**Sentry**)
- Code scanning + SLA for vulnerabilities (**DefectDojo**)
- Basic technical metrics (**prom-client**)
- Logs (**Kibana**)

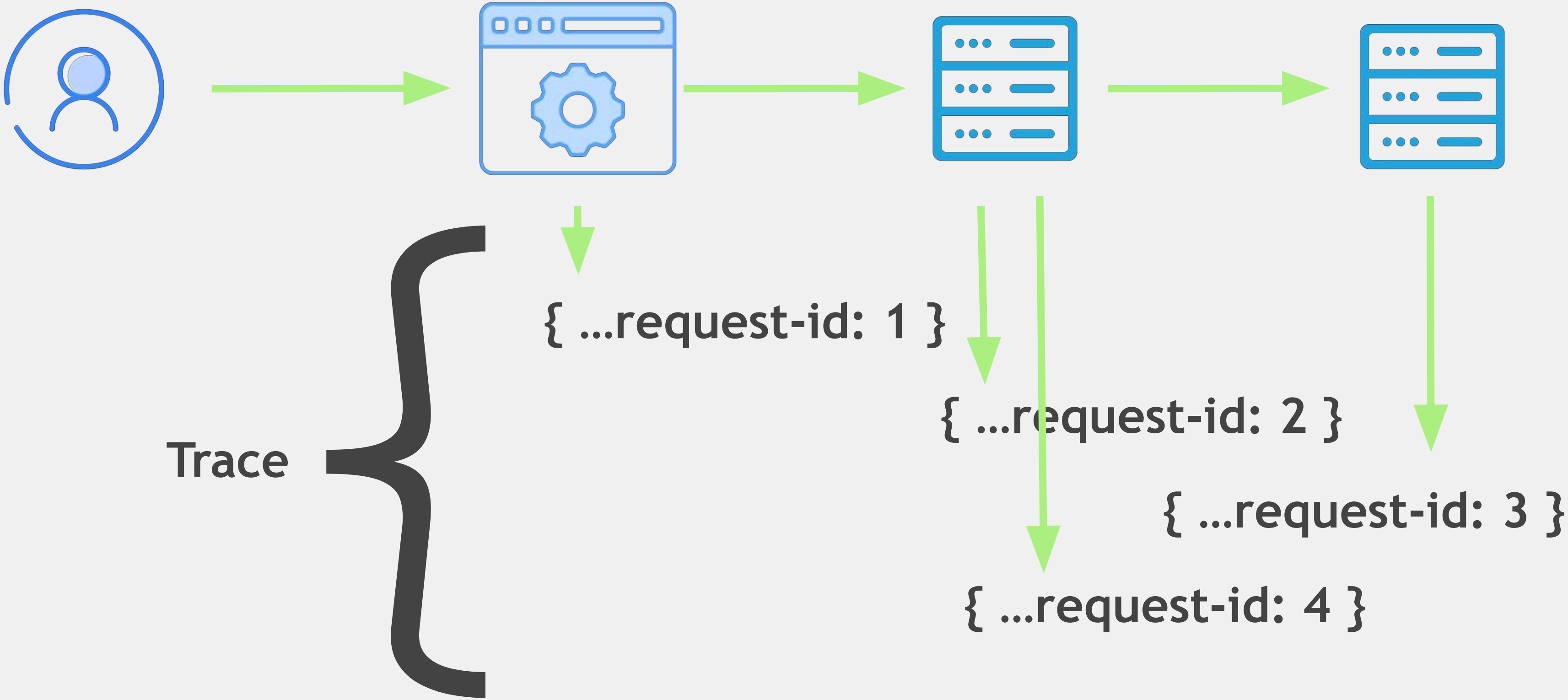


Logs – records of some events



```
{ time: 08:00, x-request-id: 1 }
{ time: 08:01, x-request-id: 2 }
{ time: 08:02, x-request-id: 3 }
```

Traces – records of all events



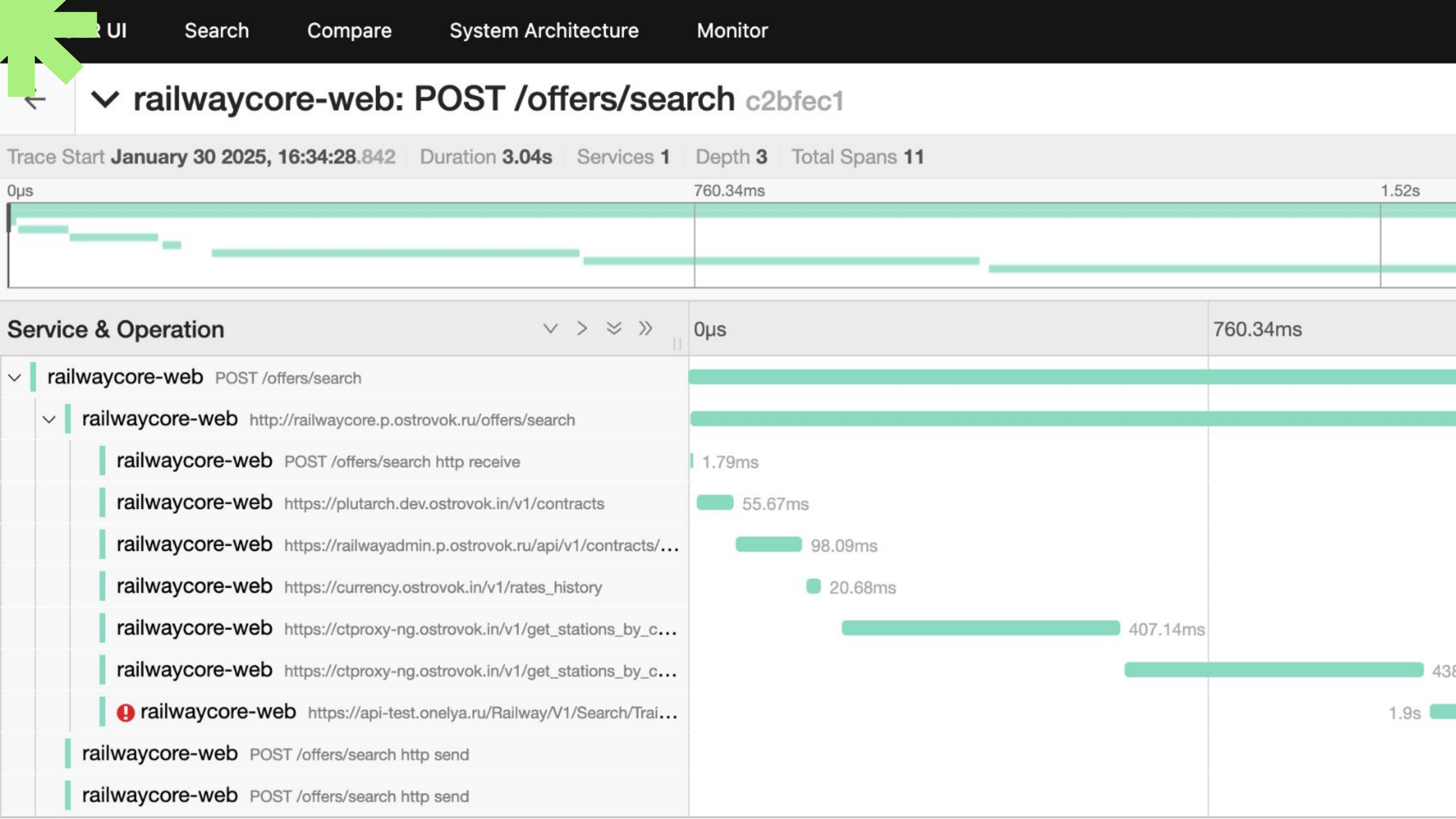
Trace

{ ...request-id: 1 }

{ ...request-id: 2 }

{ ...request-id: 3 }

{ ...request-id: 4 }

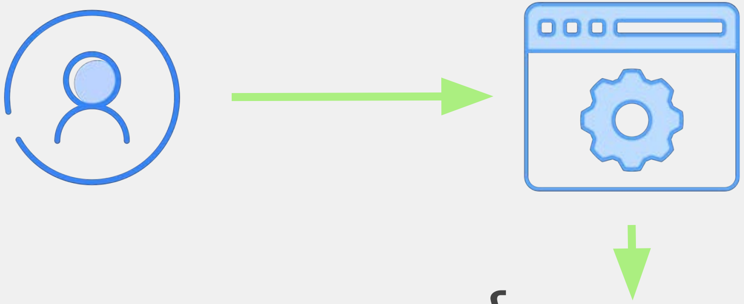


railwaycore-web: POST /offers/search c2bfec1

Trace Start **January 30 2025, 16:34:28**.842 | Duration **3.04s** | Services **1** | Depth **3** | Total Spans **11**

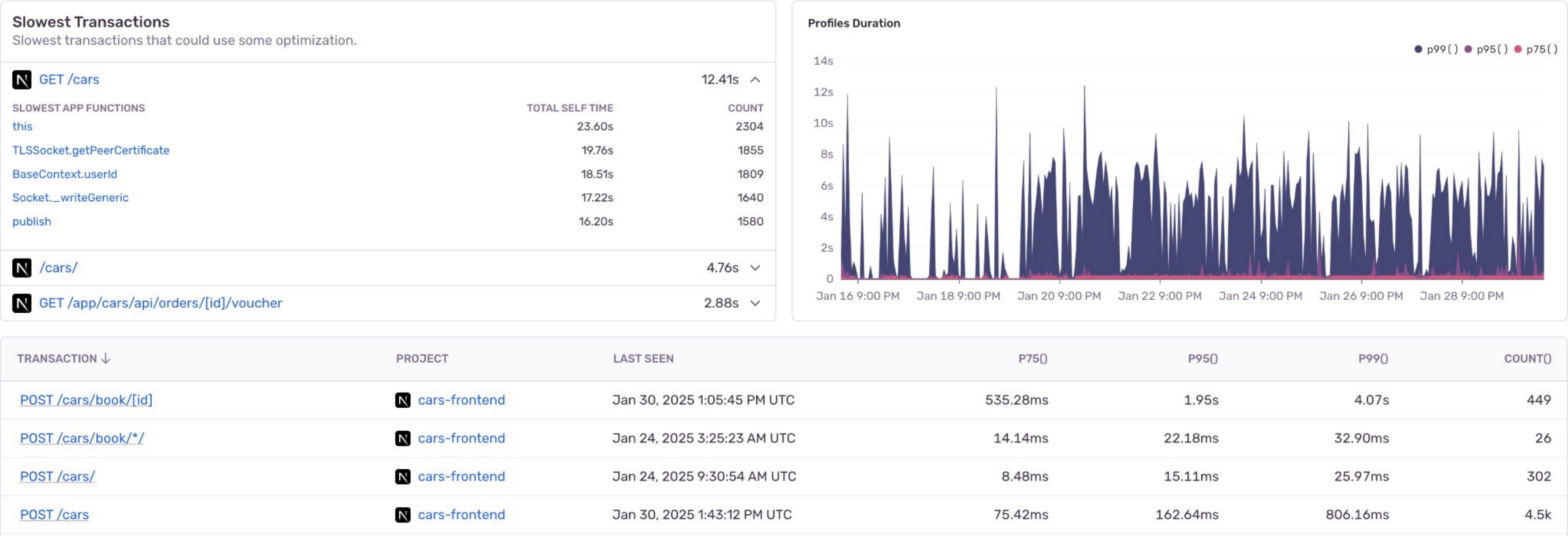
Service & Operation	
railwaycore-web POST /offers/search	
railwaycore-web http://railwaycore.p.ostrovok.ru/offers/search	
railwaycore-web POST /offers/search http receive	1.79ms
railwaycore-web https://plutarch.dev.ostrovok.in/v1/contracts	55.67ms
railwaycore-web https://railwayadmin.p.ostrovok.ru/api/v1/contracts/…	98.09ms
railwaycore-web https://currency.ostrovok.in/v1/rates_history	20.68ms
railwaycore-web https://ctproxy-ng.ostrovok.in/v1/get_stations_by_c…	407.14ms
railwaycore-web https://ctproxy-ng.ostrovok.in/v1/get_stations_by_c…	438
railwaycore-web https://api-test.onelya.ru/Railway/V1/Search/Trai…	1.9s
railwaycore-web POST /offers/search http send	
railwaycore-web POST /offers/search http send	

Profiles – detailed records



Profile

```
{
   step: warm_up, time: 12ms, mem: 12b,
   step: get_data, time: 200ms, mem: 242b,
   step: process, time: 137ms, mem: 44b,
   …
}
```



Slowest Transactions

Slowest transactions that could use some optimization.

N GET /cars 12.41s ^

SLOWEST APP FUNCTIONS	TOTAL SELF TIME	COUNT
this	23.60s	2304
TLSSocket.getPeerCertificate	19.76s	1855
BaseContext.userId	18.51s	1809
Socket._writeGeneric	17.22s	1640
publish	16.20s	1580

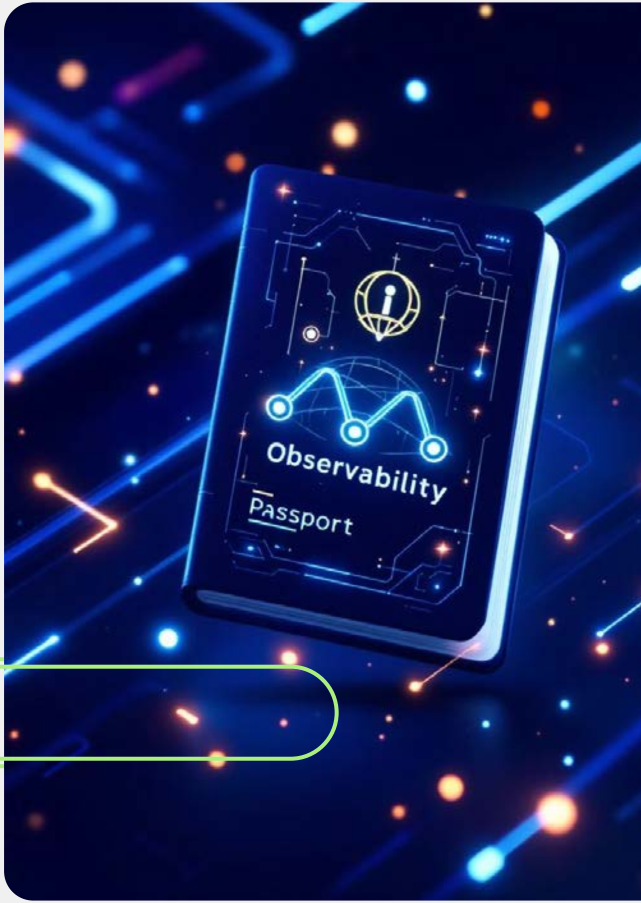
N /cars/ 4.76s ⌄

N GET /app/cars/api/orders/[id]/voucher 2.88s ⌄

TRANSACTION ↓	PROJECT	LAST SEEN	P75()	P95()	P99()	COUNT()
POST /cars/book/[id]	N cars-frontend	Jan 30, 2025 1:05:45 PM UTC	535.28ms	1.95s	4.07s	449
POST /cars/book/*/	N cars-frontend	Jan 24, 2025 3:25:23 AM UTC	14.14ms	22.18ms	32.90ms	26
POST /cars/	N cars-frontend	Jan 24, 2025 9:30:54 AM UTC	8.48ms	15.11ms	25.97ms	302
POST /cars	N cars-frontend	Jan 30, 2025 1:43:12 PM UTC	75.42ms	162.64ms	806.16ms	4.5k

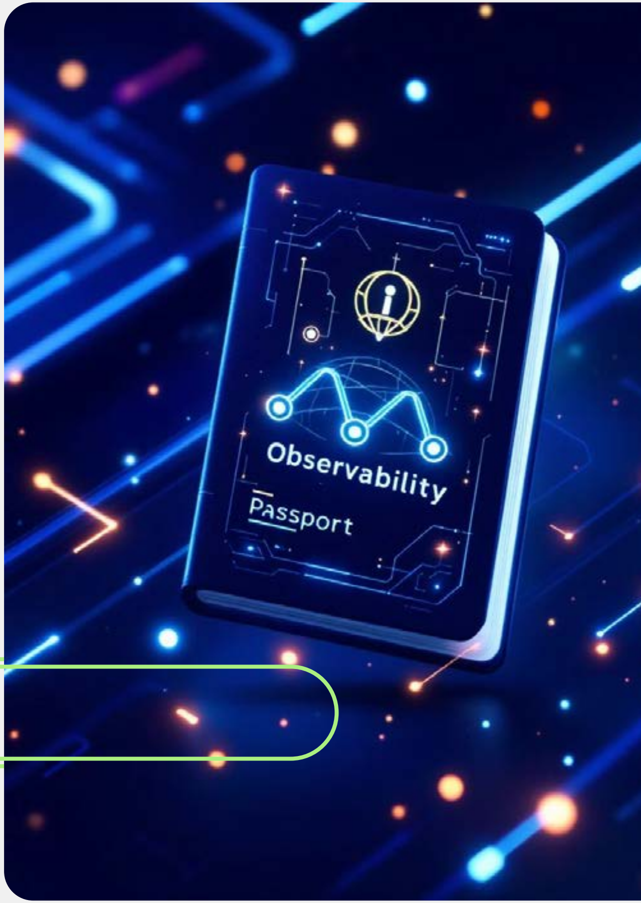
What's in my passport?

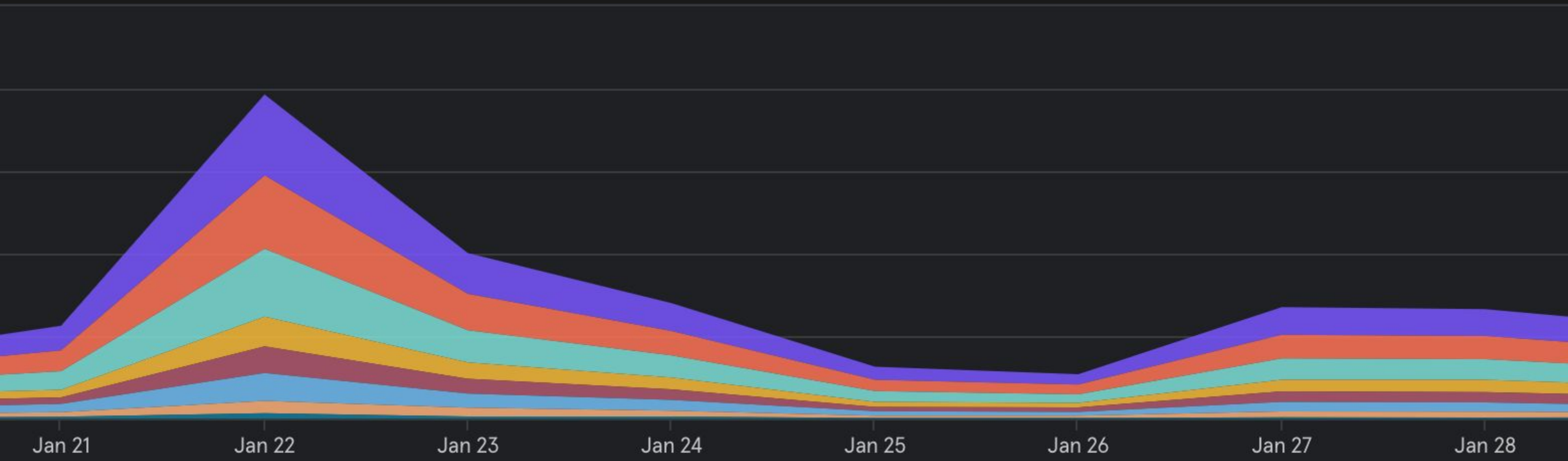
- Error tracker (**Sentry**)
- Code scanning + SLA for vulnerabilities (**DefectDojo**)
- Basic technical metrics (**prom-client**)
- Logs (**Kibana**)
- Profiling (**Sentry**)



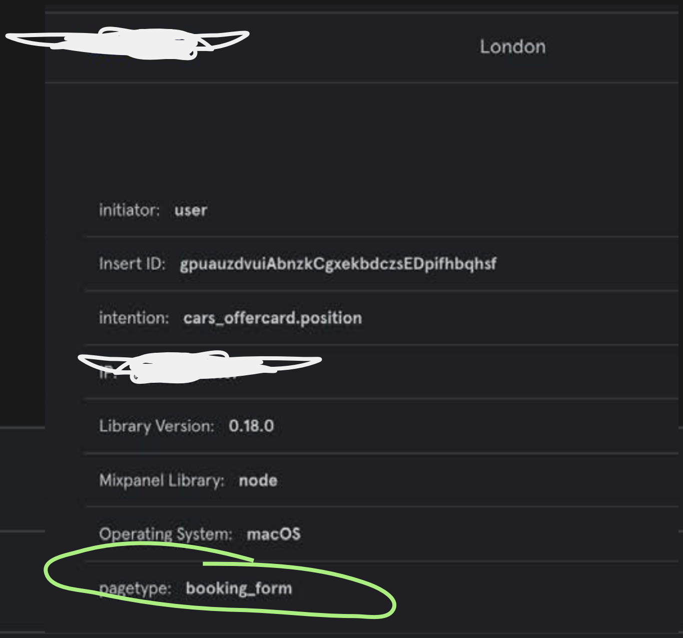
What's in my passport?

- Error tracker (**Sentry**)
- Code scanning + SLA for vulnerabilities (**DefectDojo**)
- Basic technical metrics (**prom-client**)
- Logs (**Kibana**)
- Profiling (**Sentry**)
- Product analytics (**Mixpanel**)





Jan 21
Jan 22
Jan 23
Jan 24
Jan 25
Jan 26
Jan 27
Jan 28



[redacted] London

initiator: user

Insert ID: gpuauzdvuiAbnzkCgxekbdczsEDpifhbqhsf

intention: cars_offercard.position

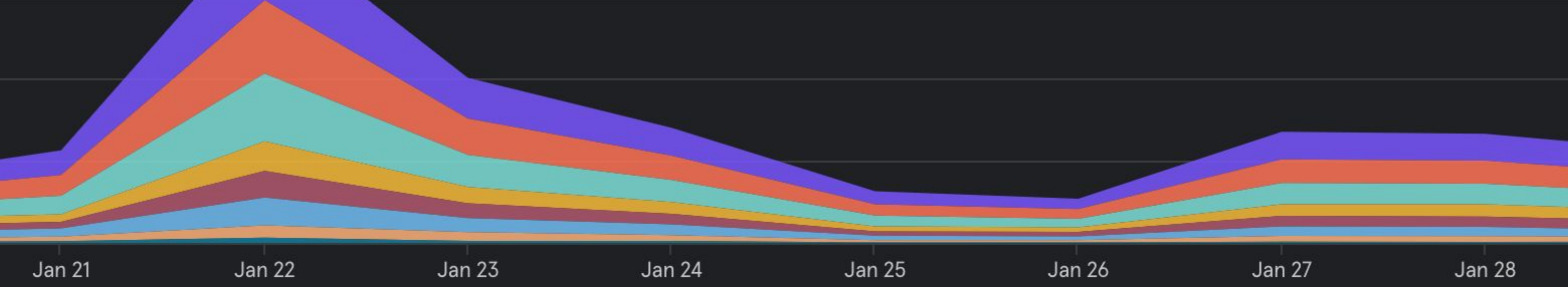
[redacted]

Library Version: 0.18.0

Mixpanel Library: node

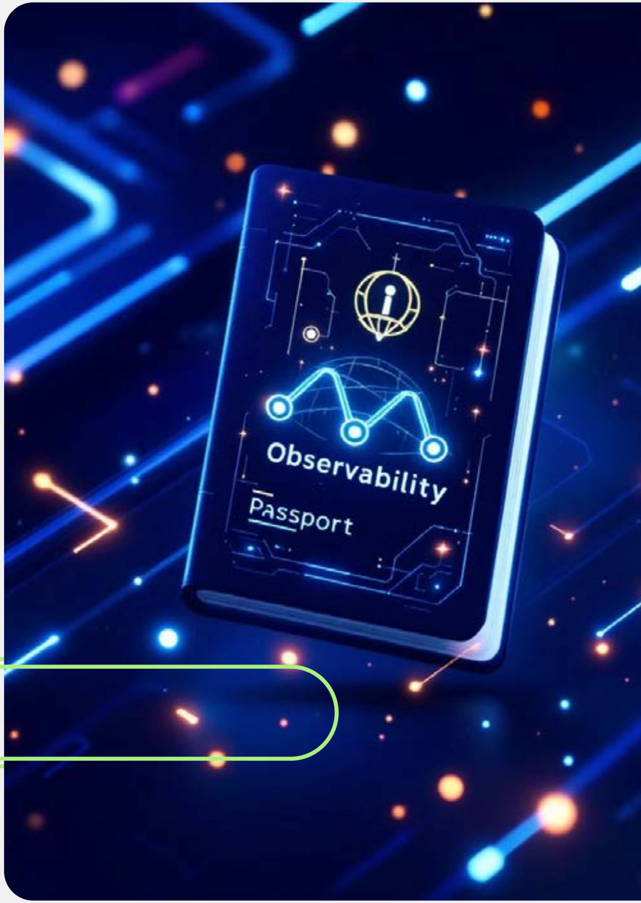
Operating System: macOS

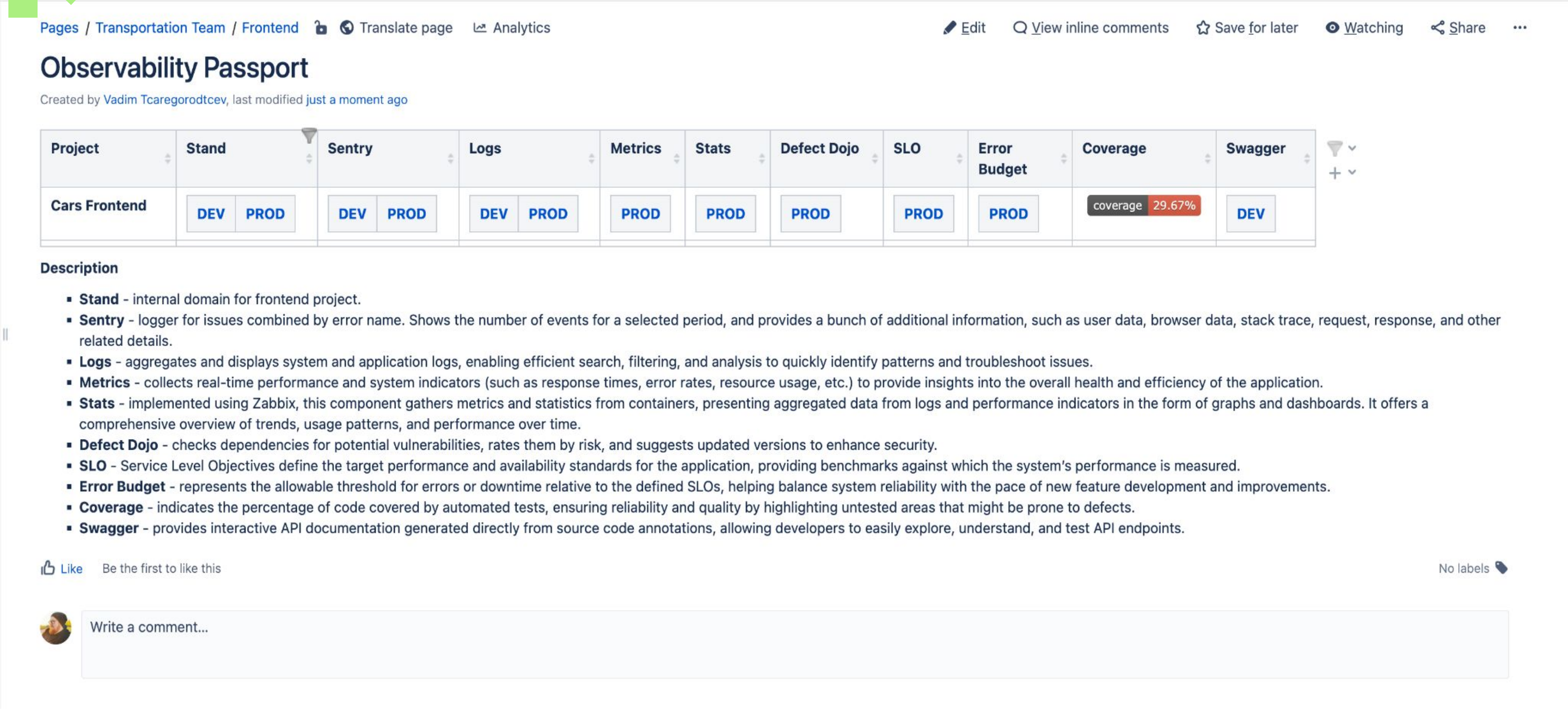
pagetype: booking_form



What's in my passport?

- Error tracker (**Sentry**)
- Code scanning + SLA for vulnerabilities (**DefectDojo**)
- Basic technical metrics (**prom-client**)
- Logs (**Kibana**)
- Profiling (**Sentry**)
- Product analytics (**Mixpanel**)
- SLA for availability
- Test coverage
- Swagger





Observability Passport

Created by Vadim Tcaregorodtcev, last modified just a moment ago

Project	Stand	Sentry	Logs	Metrics	Stats	Defect Dojo	SLO	Error Budget	Coverage	Swagger
Cars Frontend	DEV PROD	DEV PROD	DEV PROD	PROD	PROD	PROD	PROD	PROD	coverage 29.67%	DEV

Description

- **Stand** - internal domain for frontend project.
- **Sentry** - logger for issues combined by error name. Shows the number of events for a selected period, and provides a bunch of additional information, such as user data, browser data, stack trace, request, response, and other related details.
- **Logs** - aggregates and displays system and application logs, enabling efficient search, filtering, and analysis to quickly identify patterns and troubleshoot issues.
- **Metrics** - collects real-time performance and system indicators (such as response times, error rates, resource usage, etc.) to provide insights into the overall health and efficiency of the application.
- **Stats** - implemented using Zabbix, this component gathers metrics and statistics from containers, presenting aggregated data from logs and performance indicators in the form of graphs and dashboards. It offers a comprehensive overview of trends, usage patterns, and performance over time.
- **Defect Dojo** - checks dependencies for potential vulnerabilities, rates them by risk, and suggests updated versions to enhance security.
- **SLO** - Service Level Objectives define the target performance and availability standards for the application, providing benchmarks against which the system's performance is measured.
- **Error Budget** - represents the allowable threshold for errors or downtime relative to the defined SLOs, helping balance system reliability with the pace of new feature development and improvements.
- **Coverage** - indicates the percentage of code covered by automated tests, ensuring reliability and quality by highlighting untested areas that might be prone to defects.
- **Swagger** - provides interactive API documentation generated directly from source code annotations, allowing developers to easily explore, understand, and test API endpoints.