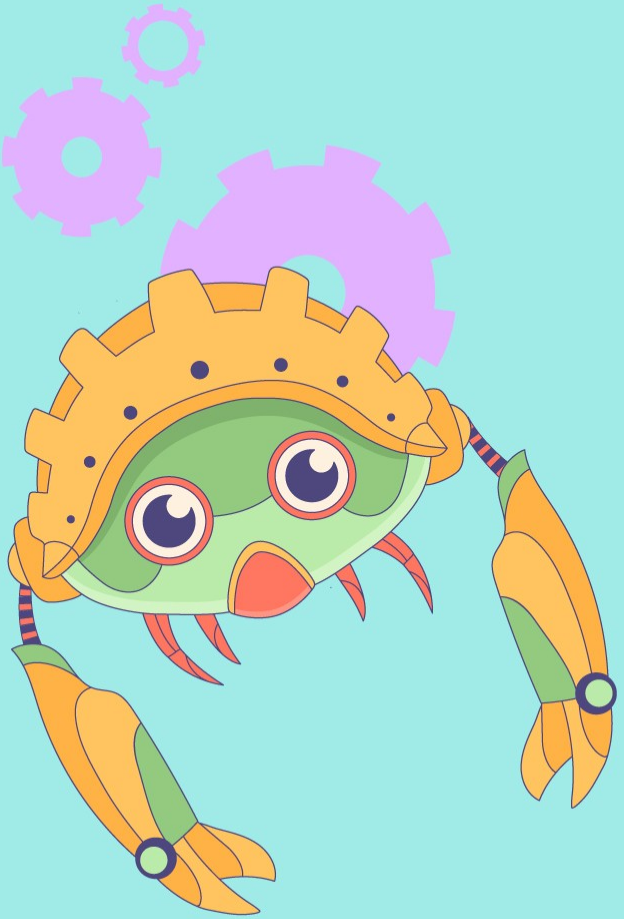
Andreas Klostermaier

CEO @ append [x] GmbH

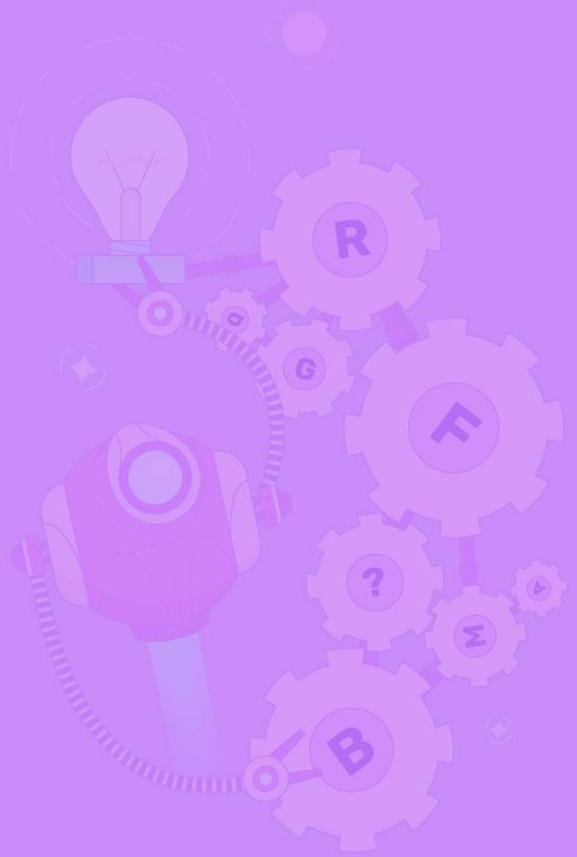
!#[no-pain]

Rust at 15%: Production-Ready Without the Pain



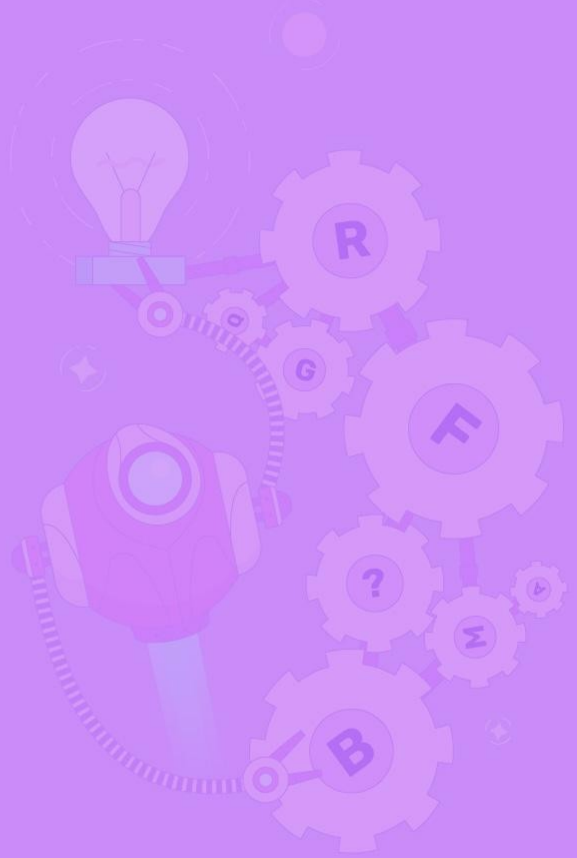
ABOUT ME

- CEO of append [x] GmbH (near Munich)
- 30+ years in system integration
- All-in on Rust since 2021
- Trainer for corporate Rust introduction courses
- Host of Rust Berlin Hack & Learn Meetup
- Dev of a hospital integration server in Rust



ABOUT ME

So, that makes me a Rust expert, right?



ABOUT ME

The Confession:
I am not!

WHAT THIS TALK IS ABOUT

In this talk, I want to show you how Rust — in a **greatly simplified form** — can be used for the **ad-hoc demands of system integration**.
My goal is to **encourage application developers** to try Rust, to **remove the fear** of its complexity, and to **build confidence** in Rust as a language for practical, real-world application development.

SYSTEM INTEGRATION

Analyze → Connect → Automate

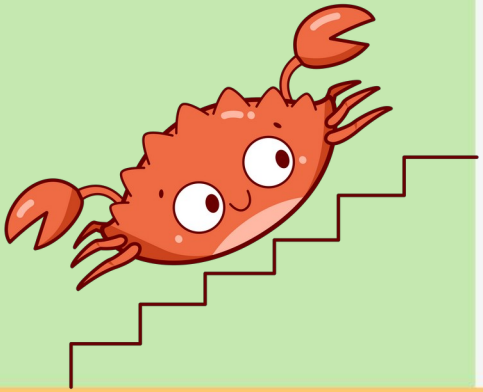
- **Sources:** REST/SOAP / LDAP / webhooks / file shares / HTTP / SFTP / SQL etc.
- **Formats:** CSV / XML / JSON / YAML / Excel 
- **Transform:** custom code, RegEx, XSLT, awk ...
- **Deliver**

SYSTEM INTEGRATION

Analyze → Connect → Automate

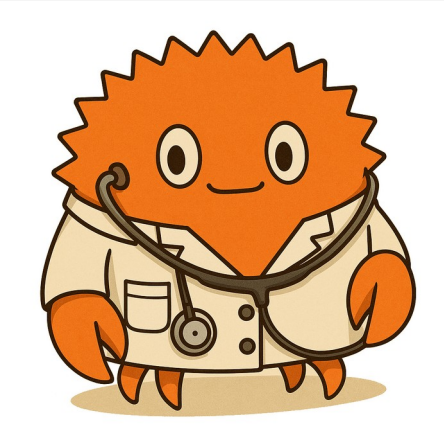
- If multiple systems are involved: **database mirror**
- If user interaction is required: **web frontend**
- Combine and deploy all components on-prem as an **Integration Server**

MY ROAD TO RUST



- Entry point Cloudflare Workers
- Side project migration: Java → Rust
- Steep learning curve (with frustration)
- Evaluation for systems integration
- Built COVID access system in 1 week
- Gained trust in Rust ecosystem
- All-in on Rust

HOSPITAL PROJECT

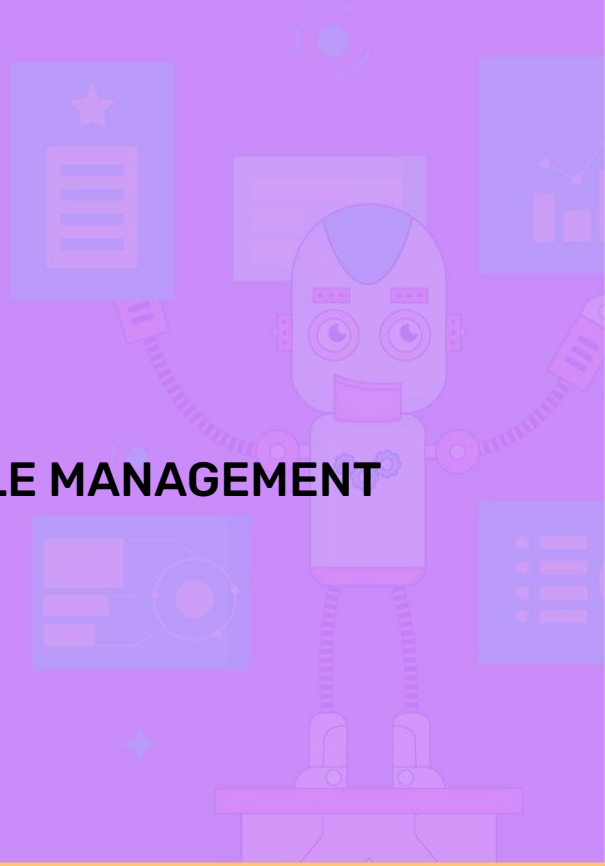


- Bavarian hospital group with ≈ 1200 employees
- 5 healthcare units across 2 locations
- Emergency care (Level 2), geriatrics, MVZ centers, training institutes
- ≈ 50 IT systems from different vendors
- Increasing process and compliance complexity in healthcare

Case Study

HOSPITAL PROJECT

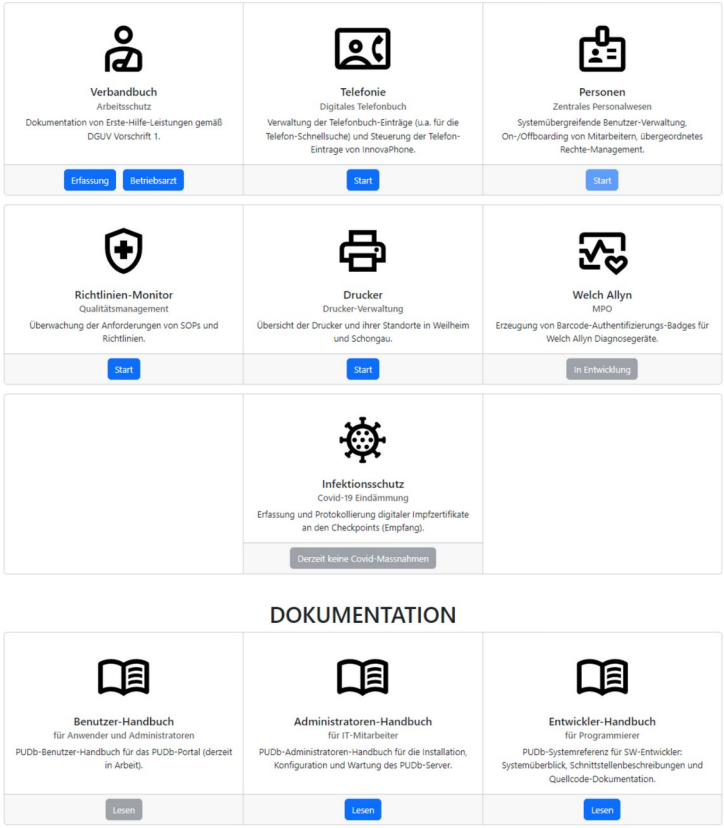
EMPLOYEE LIFECYCLE MANAGEMENT



Case Study

EMPLOYEE LIFECYCLE MANAGEMENT

- 28 IT systems with own user DBs
- Dump contracts, roles, access... to mirror DB
- Define *sources of truth*
- Sync identities across systems
- Push changes → tickets or auto-sync
- Automate on/off-boarding
- Bonus: interactive intranet phone directory and other services



Case Study

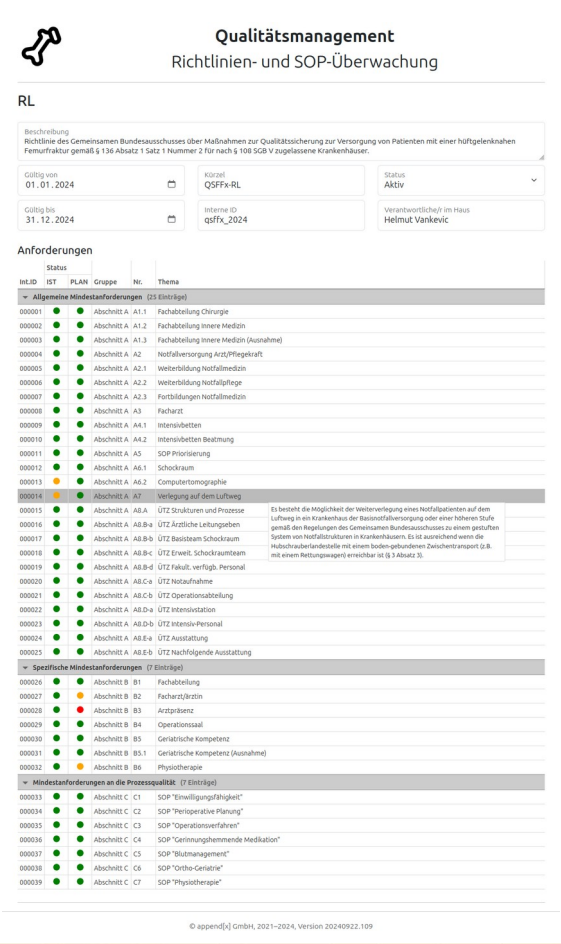
HOSPITAL PROJECT

REALTIME COMPLIANCE MONITORING

Case Study

REALTIME COMPLIANCE MONITORING

- Extract qualification rules (G-BA)
- Map to staff & shifts
- Merge timelines → detect gaps
- Proactive & retrospective proof of compliance



Qualitätsmanagement
Richtlinien- und SOP-Überwachung

RL

Beschreibung
Richtlinie des Gemeinsamen Bundesausschusses über Maßnahmen zur Qualitätssicherung zur Versorgung von Patienten mit einer hüftgelenknahen Femurfraktur gemäß § 136 Absatz 1 Satz 1 Nummer 2 für nach § 108 SGB V zugelassene Krankenhäuser.

Gültig von: 01.01.2024 | Kürzel: QSFFx-RL | Status: Aktiv
Gültig bis: 31.12.2024 | Interne ID: qsffx_2024 | Verantwortliche/r im Haus: Helmut Vankevic

Anforderungen

Int.ID	Status IST	Status PLAN	Gruppe	Nr.	Thema
Allgemeine Mindestanforderungen (25 Einträge)					
000001			Abschnitt A	A1.1	Fachabteilung Chirurgie
000002			Abschnitt A	A1.2	Fachabteilung Innere Medizin
000003			Abschnitt A	A1.3	Fachabteilung Innere Medizin (Ausnahme)
000004			Abschnitt A	A2	Notfallversorgung Arzt/Pflegekraft
000005			Abschnitt A	A2.1	Weiterbildung Notfallmedizin
000006			Abschnitt A	A2.2	Weiterbildung Notfallpflege
000007			Abschnitt A	A2.3	Fortbildungen Notfallmedizin
000008			Abschnitt A	A3	Facharzt
000009			Abschnitt A	A4.1	Intensivbetten
000010			Abschnitt A	A4.2	Intensivbetten Beatmung
000011			Abschnitt A	A5	SOP Priorisierung
000012			Abschnitt A	A6.1	Schockraum
000013			Abschnitt A	A6.2	Computertomographie
000014			Abschnitt A	A7	Verlegung auf dem Luftweg
000015			Abschnitt A	A8.A	ÜTZ Strukturen und Prozesse
000016			Abschnitt A	A8.B-a	ÜTZ Ärztliche Leitungsebene
000017			Abschnitt A	A8.B-b	ÜTZ Basisteam Schockraum
000018			Abschnitt A	A8.B-c	ÜTZ Erweit. Schockraumteam
000019			Abschnitt A	A8.B-d	ÜTZ Fakult. verfügb. Personal
000020			Abschnitt A	A8.C-a	ÜTZ Notaufnahme
000021			Abschnitt A	A8.C-b	ÜTZ Operationsabteilung
000022			Abschnitt A	A8.D-a	ÜTZ Intensivstation
000023			Abschnitt A	A8.D-b	ÜTZ Intensiv-Personal
000024			Abschnitt A	A8.E-a	ÜTZ Ausstattung
000025			Abschnitt A	A8.E-b	ÜTZ Nachfolgende Ausstattung
Spezifische Mindestanforderungen (7 Einträge)					
000026			Abschnitt B	B1	Fachabteilung
000027			Abschnitt B	B2	Facharzt/Ärztin
000028			Abschnitt B	B3	Arztpräsenz
000029			Abschnitt B	B4	Operationssaal
000030			Abschnitt B	B5	Geriatrische Kompetenz
000031			Abschnitt B	B5.1	Geriatrische Kompetenz (Ausnahme)
000032			Abschnitt B	B6	Physiotherapie
Mindestanforderungen an die Prozessqualität (7 Einträge)					
000033			Abschnitt C	C1	SOP "Einwilligungsfähigkeit"
000034			Abschnitt C	C2	SOP "Perioperative Planung"
000035			Abschnitt C	C3	SOP "Operationsverfahren"
000036			Abschnitt C	C4	SOP "Gerinnungshemmende Medikation"
000037			Abschnitt C	C5	SOP "Blutmanagement"
000038			Abschnitt C	C6	SOP "Ortho-Geriatrie"
000039			Abschnitt C	C7	SOP "Physiotherapie"

Es besteht die Möglichkeit der Weiterverlegung eines Notfallpatienten auf dem Luftweg in ein Krankenhaus der Basisnotfallversorgung oder einer höheren Stufe gemäß den Regelungen des Gemeinsamen Bundesausschusses zu einem gestuften System von Notfallstrukturen in Krankenhäusern. Es ist ausreichend wenn die Hubschrauberlandestelle mit einem boden-gebundenen Zwischentransport (z.B. mit einem Rettungswagen) erreichbar ist (§ 3 Absatz 3).

ARCHITECTURE OVERVIEW

- **Rust** + **Rocket** (0.5.1)
- **SQLite** 3 + **sqlx**
- Handful of helper crates:
  serde, crypto, anyhow …
- Frontend:
  plain HTML/JS, no Frameworks, no Node
- JS libs:
  Tabulator / Bootstrap / Autocomplete

ARCHITECTURE OVERVIEW

Single binary executable that:

- serves static pages with **Tera templates**
- delivers images, JS libs, and documentation
- provides the **API backend**
- schedules periodic tasks via **tokio-cron-scheduler**
- contains all the **business logic**.

ARCHITECTURE OVERVIEW

Deployment on:

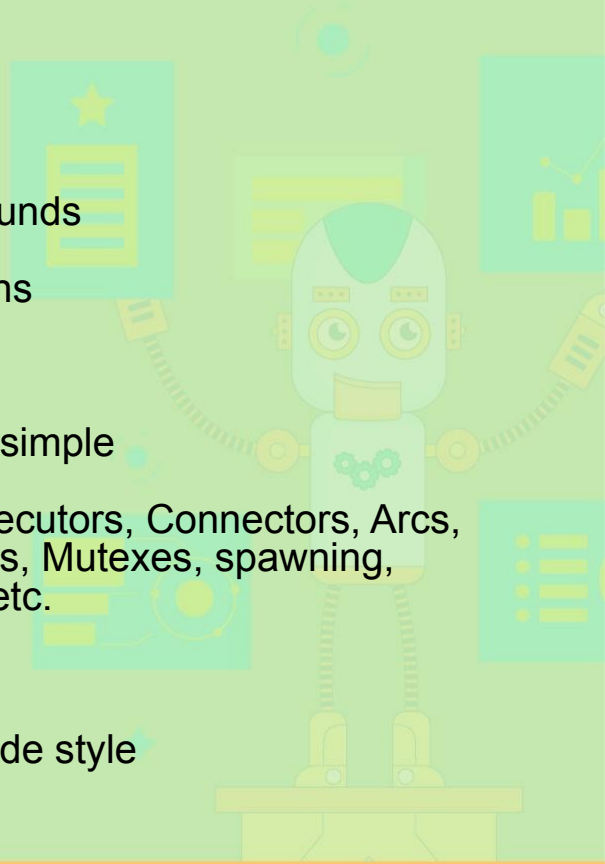
- **Virtual Linux Server (2 Cores / 8 GB RAM)**
- On premise
- Reverse proxy (nginx)
- Integrated with hospital monitoring

!#[no-pain]

What is 15% Rust?

If you look at our code, you'll find surprisingly little "Rust wizardry":

- NO generics / trait bounds
- NO lifetime annotations
- NO unsafe magic
- async/await: yes, but simple
- NO Pins, Futures, Executors, Connectors, Arcs, Atomics, Locks, Actors, Mutexes, spawning, blocking, polling etc. etc.
- Few macros
- Simple, imperative code style



What is 15% Rust?

“A Python dev could read our Rust code!”

The Challenge

Our Main Challenge is not business logic complexity, nor Rust, nor tight deadlines.

The real challenge is **operational responsibility!**

- Our code becomes mission-critical — it must stay stable for years!
- It should run maintenance-free for as long as possible!
- And even years later, others must be able to maintain or extend it!

Building the 100-Year Integration Server

Inspired by Alexander Petros'
"Building the Hundred-Year Web Service":

- **Useful** for future generations
- **Maintainable** for future generations
- **Adaptable** to the needs of future generations

“Reduce Complexity!”

The Three Types of Constraints of ‘15% Rust’

- **Natural:** advanced features not needed
- **Deliberate:** SQLite, no ORM, no JS frameworks
- **Rust-specific:** Rocket, Anyhow, clarity over cleverness, Web-GUI (Rust-GUIs are not 100-year-ready yet)

Why Rust?

- **Performance**
- **Runtime Safety**

Why Performance Matters!

- Our most complex blocking function runs under 500 ms
- Everything else is even faster
- We never need to optimize for speed
- We can stay with SQLite — no need for heavy transactional DBs like Postgres
- **High speed** → less tuning → fewer layers → **lower complexity**

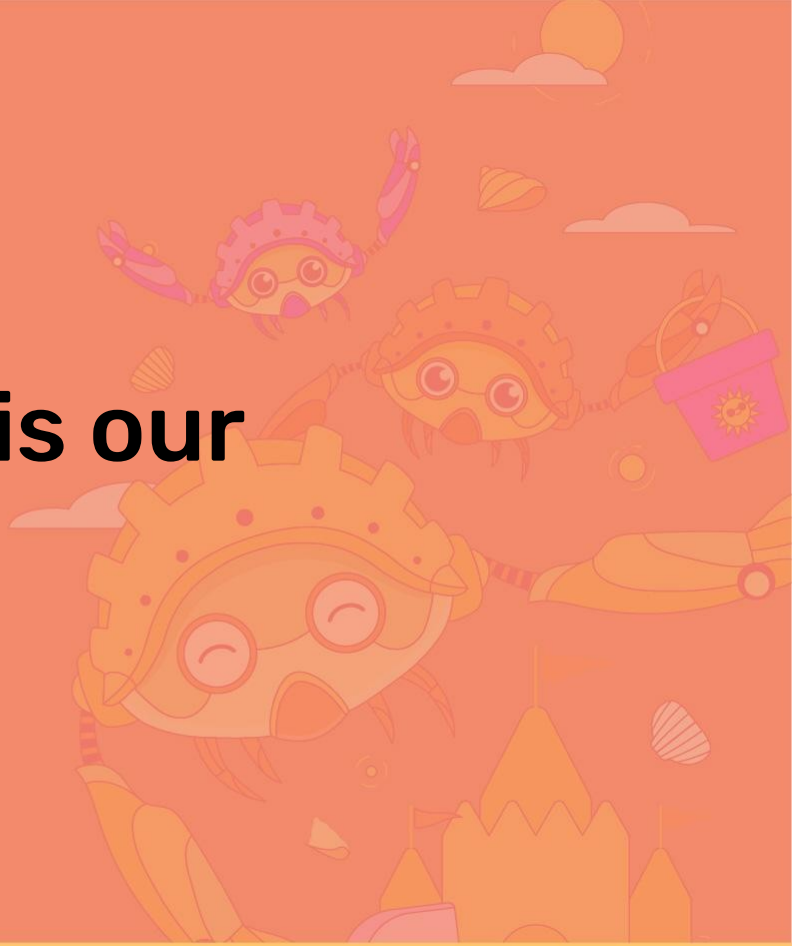
Why Rust?

“(Free) performance is neither a luxury, nor a fetish – it reduces complexity!!”

Operational Safety Matters!

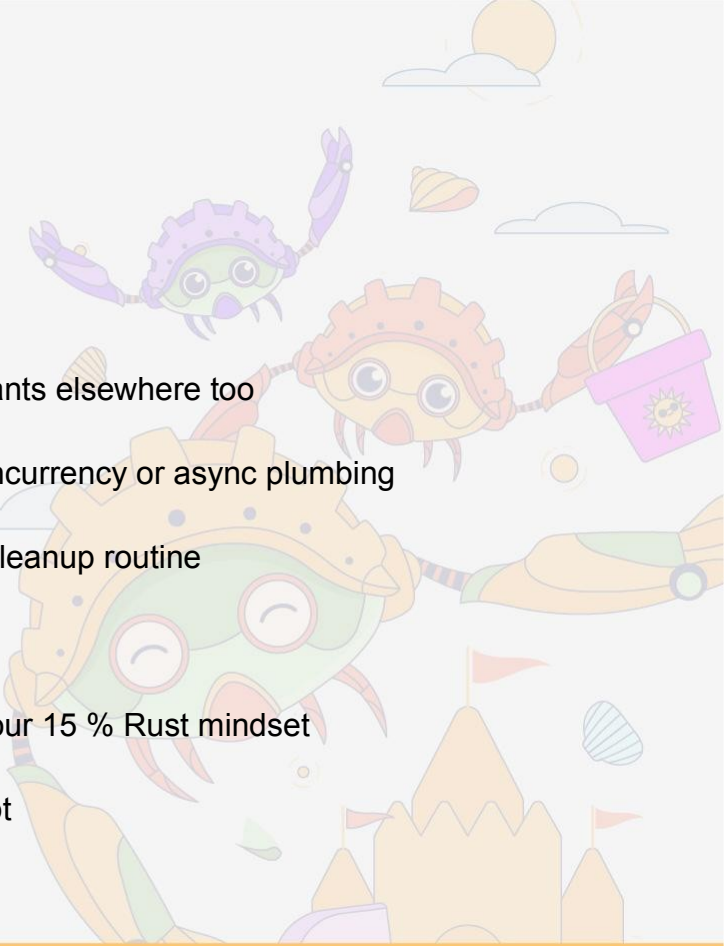
- Rust's safety guarantees drastically reduce support effort
- No runtime crashes, no memory leaks, no undefined behavior
- Production systems can run mostly unattended
- A small team can operate mission-critical software with confidence
- No need for a X-person 24/7 support department

“Rust’s reliability is our scalability!”



Edge Cases

- Sometimes, advanced Rust features can’t be avoided
- Example: Rocket forces async, so we prefer async variants elsewhere too
- We make sane trade-offs — async yes, but no deep concurrency or async plumbing
- Occasional lifetime errors? → handled via our “janitor” cleanup routine
- No deep theory required — just tidy up, test, move on
- For handover we have a 2-day onboarding course into our 15 % Rust mindset
- When in doubt, we start from library examples and adapt



“Don’t be afraid to be a Rust janitor!”

Takeaways

- Rust doesn’t have to be hard — not for app devs
- The difficulty of using Rust is proportional to the complexity of your problem. You only need as much Rust as your problem demands.
- Rust at 15% - more often than not, that's all it needs!
- You definitely can get real-world results with a gentle subset of the language
- Even janitors have colleagues — the Rust community always helps

What is 15% Rust?

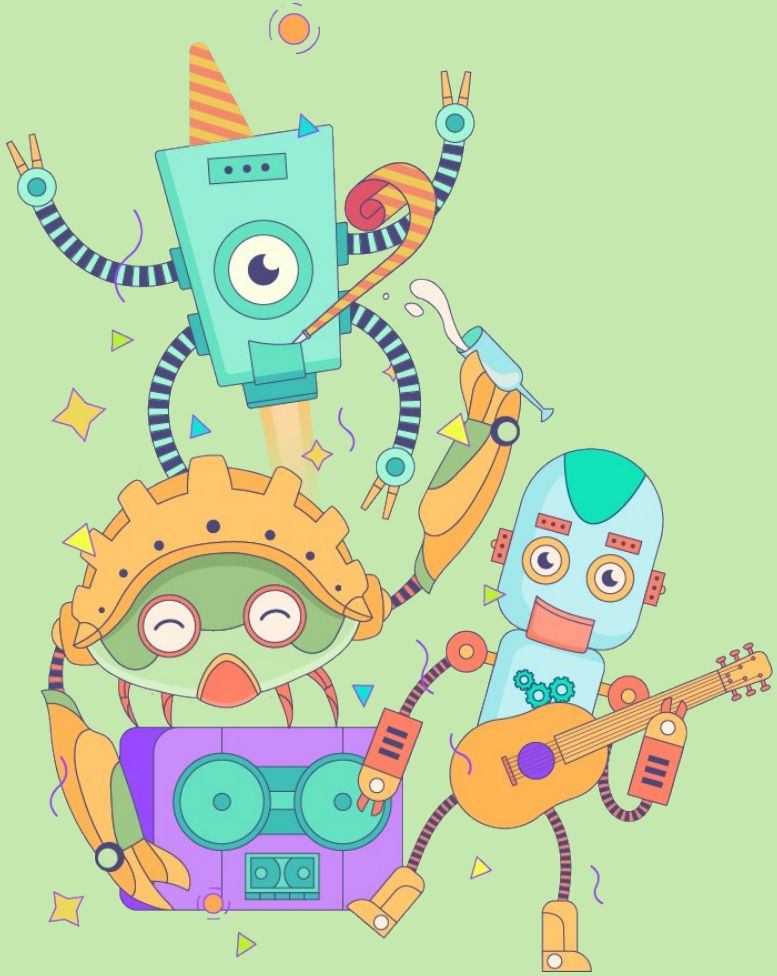
“The 15% isn’t about features or syntax — it’s the portion of time in any Rust talk or meetup when I can fully understand what’s being discussed.
The rest of the time, I’m learning by osmosis.”

Q&A

- Do we need **Rust Lite** or **RustScript**?
- Do we need a guide **Rust for Application Developers**?
- What are your experiences with **pragmatic Rust** in application development?

Thanks to the RustLab team

and Maria Letta



ANDREAS KLOSTERMAIER

a.klostermaier@appendx.de