

[matrix]

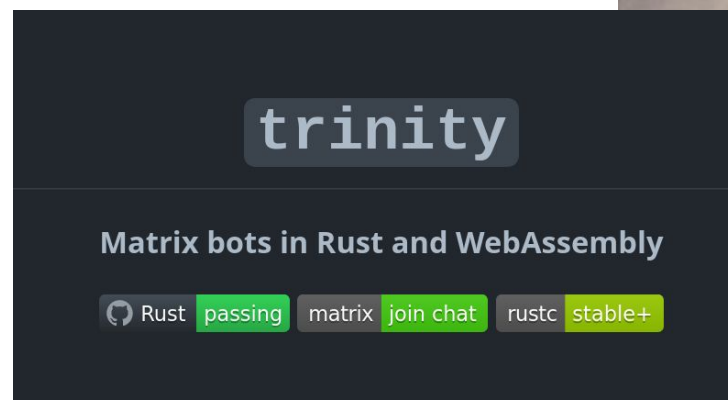
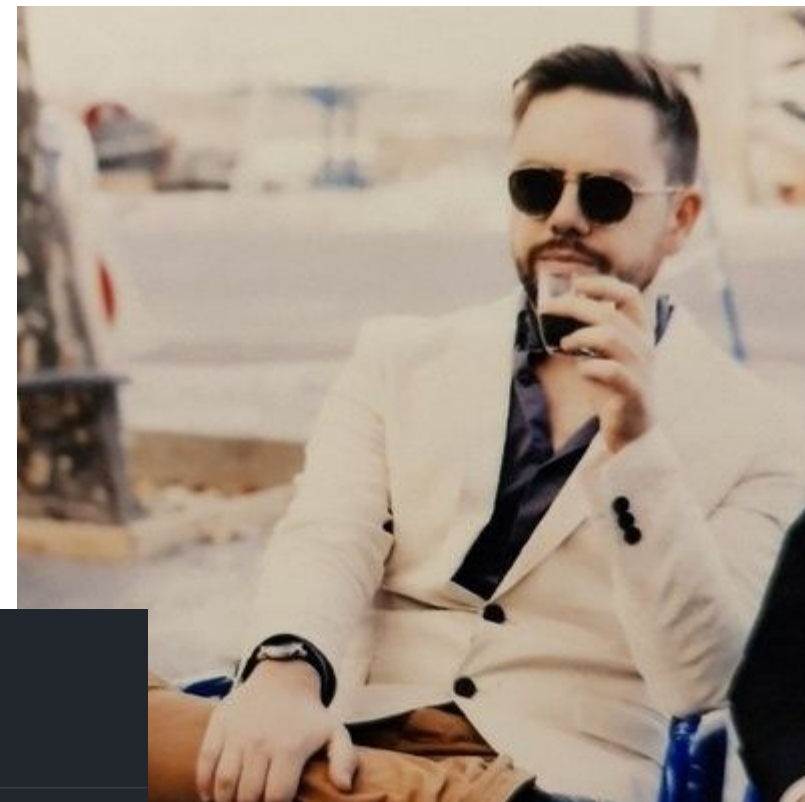
Strengthening the Base for a more robust Rust SDK

Matrix Conference 2024

mx: @benjib:element.io

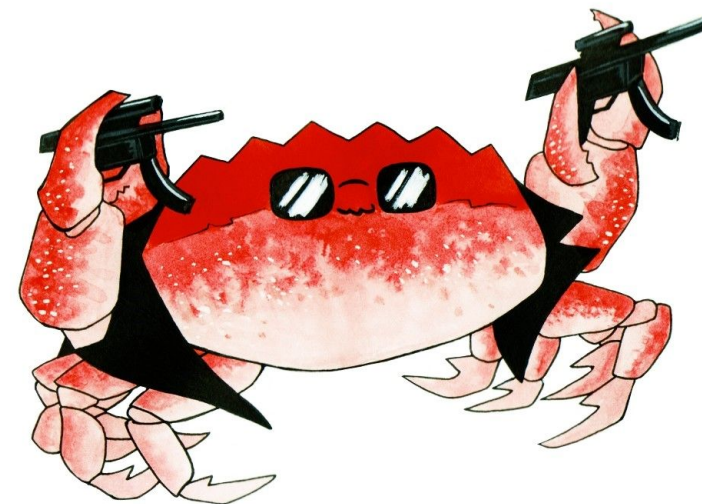
Who's that guy?

- Benjamin “bnjbvr” Bouvier
- Software engineer in Rust team @ Element
- Previously at Embark Studios as Game Engine Hacker
- Previously at Mozilla as Compiler Engineer (Spidermonkey/Wasmtime)
- Other: Framasoft, cargo-machete, kresus, rouille...



The Rust SDK

- Rust client-server API library
- github.com/matrix-org/matrix-rust-sdk
- Apache 2.0 license
- Everything one would expect from a Matrix client
 - logging in, out, reading/writing settings...
 - sending and receiving events
 - listening to sync updates and reacting to specific events via *handlers*
- End-to-end encryption comes for free!



Logo made by Ursa Johnson 🙏

A history of the Matrix Rust SDK



November 2015



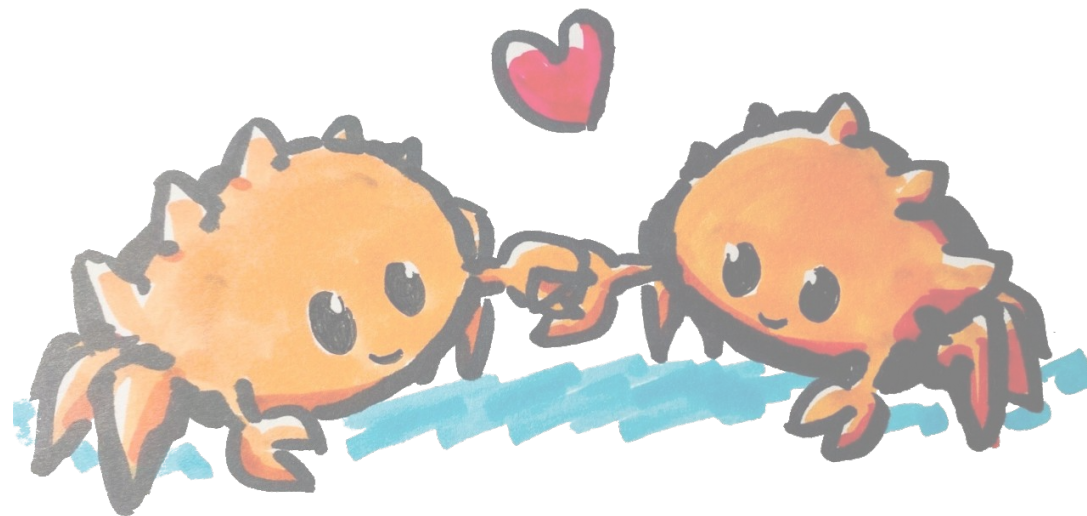
October 2019



December 2021

Why Rust?

- Pleasantly high-level *and* fast by default
- Small memory footprint
- Secure, memory-safe
 - Thanks to the ownership model
- Amazing tooling and ecosystem 🥰
- Compatible with FFIs using the C ABI
- Empowerment!

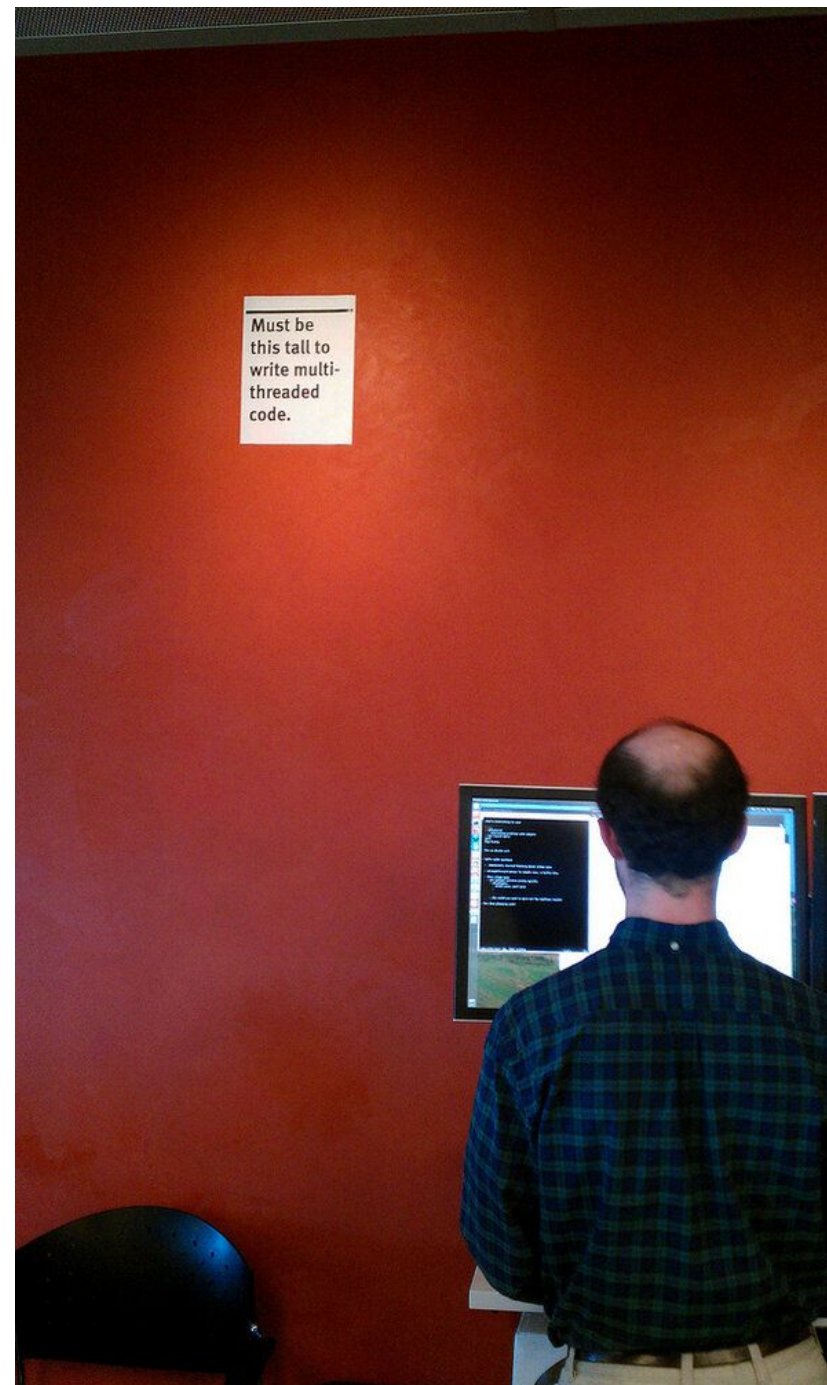


Picture by @aldeka@wandering.shop

Why Rust?

Fearless concurrency!

- Ownership model helps modeling concurrent ownership too.
- No data races *by default*.
- The ecosystem is going strong on asynchronicity.



[matrix]

Why the Rust SDK?

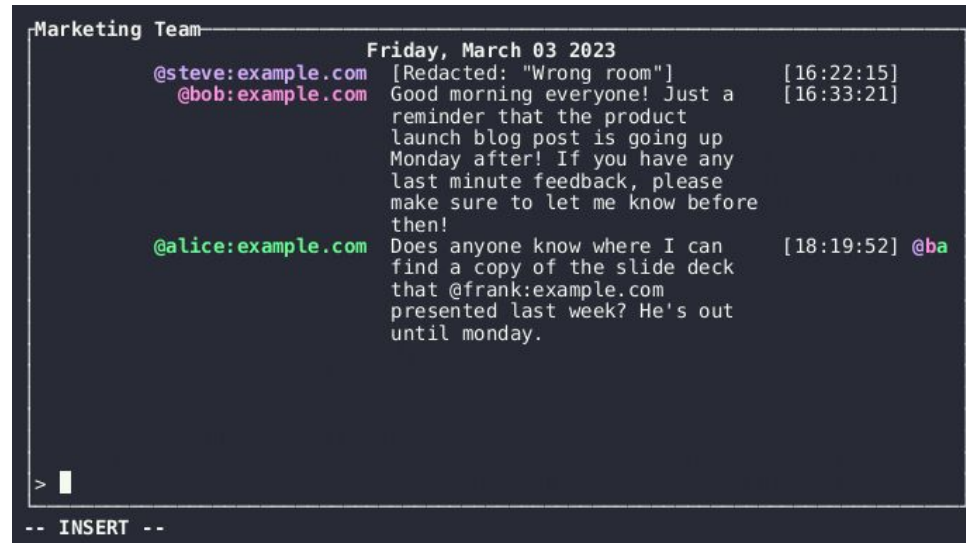
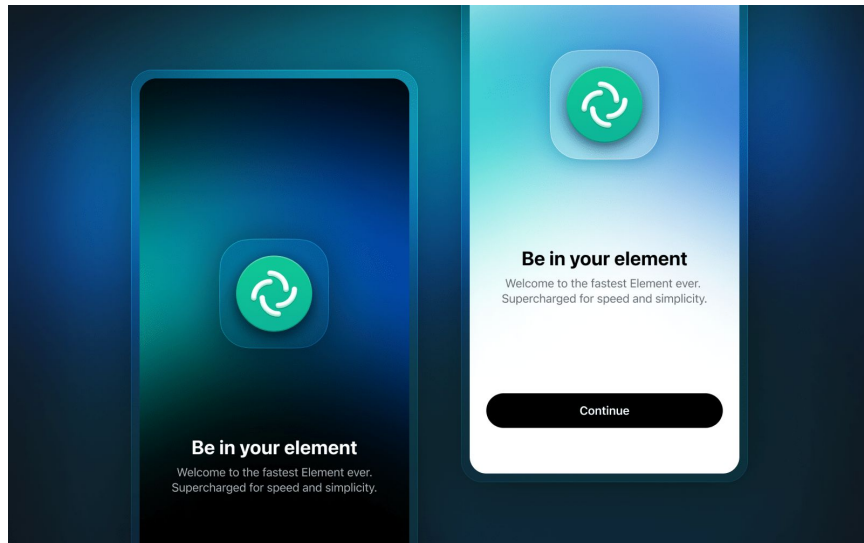
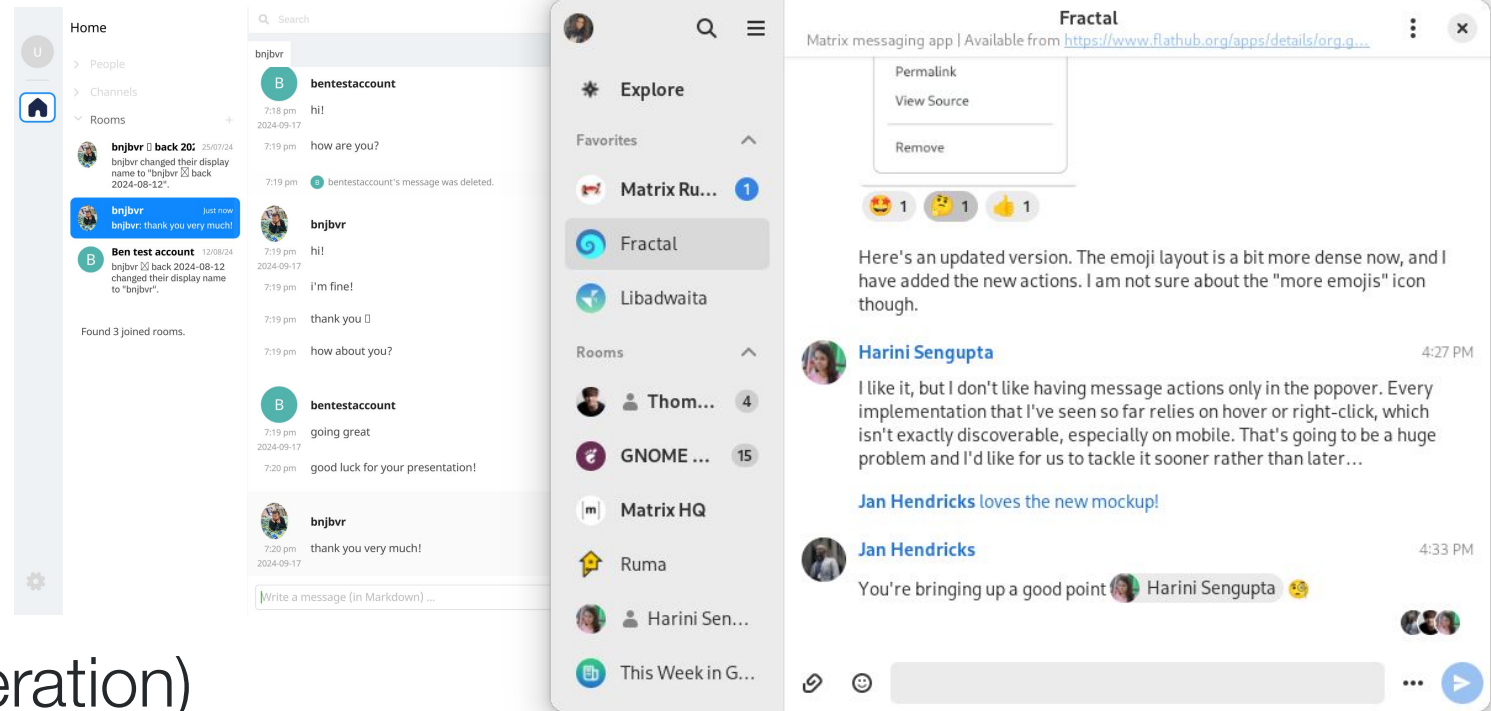
- Everything is reusable
 - a single crypto stack!
- High test coverage, fuzzing, benchmarking...
- A central place where to add features and fix bugs.



Fig 1: Figuring out SDKs at Element

Who's using it?

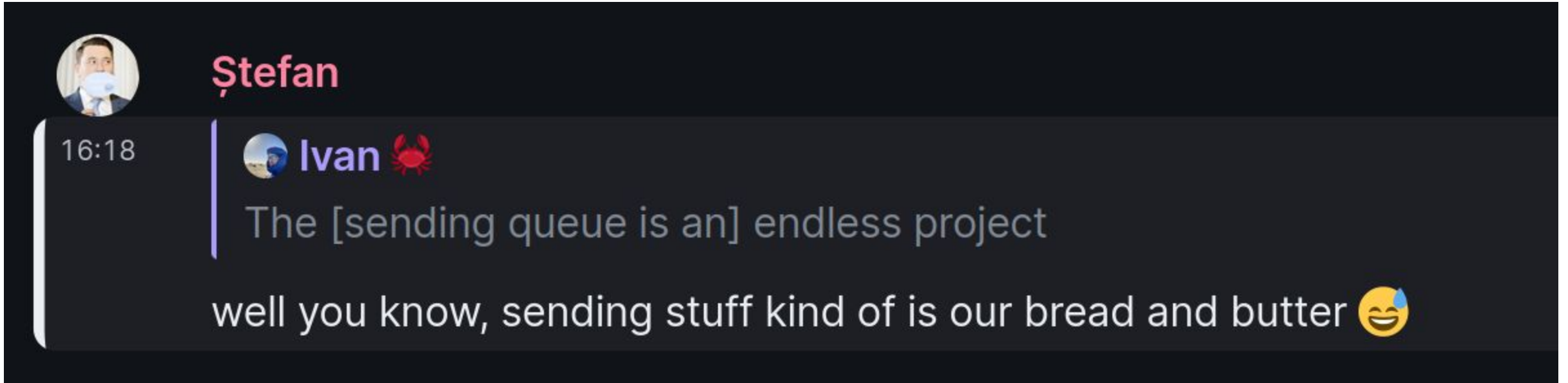
- Fractal, a GTK client
- iamb, a TUI client
- Robrix, a Desktop client
- ElementX apps
- *ElementR* (previous generation)



Improving the Foundation

[matrix]

- What's the goal of the SDK?



- Receiving events: **Simplified Sliding Sync, Event Cache**
- Sending events: **Send Queue**
- With encryption “for free”: **robust crypto**

Receiving events: simplified sliding sync

[matrix]

New synchronization protocols to receive events from the server.

- **Goal:** near-instantaneous synchronization, independently of the account's size.
 - Load incremental batches of rooms in the room list
- Sliding sync ([MSC3575](#)): a successful experiment 🧪
- Simplified sliding sync ([MSC4186](#)):
 - Simpler & better: rooms are sorted client-side
 - in Synapse and Rust SDK today 🎉
- **More details at Ivan's talk, tomorrow at 10am.**

Storing events: the Event Cache

- Observable store of events coming from *any* source (sync, pagination,...).
- Extra processing on those events: figures out read receipts, unread counts, retry decryption, etc.
- In-memory for now.
 - Efficient memory representation.
- Still a work-in-progress. In the future:
 - Persist events on disk.
 - Deduplicate events based on their orderings.
 - What's the right ordering? Topological? Sync? [MSC4033](#)?
 - Connect together islands of disjoint timelines.

Sending events: a Send Queue

- Full rewrite of the send mechanism we had.
- Highly tested in isolation.
- Pending messages are observable via updates
 - Local echoes = displayed *before* sent to the Matrix server.
- Full of useful features:
 - Queue messages while offline
 - Save pending messages on disk
 - Edit/abort/react to pending messages
 - And always do the Right Thing™
 - Automatically retry sending if network's stuttering
 - And stop if wedged for any reason
 - Manually retry sending too
- Paves the way for more “local” echoes (media uploads, etc.).

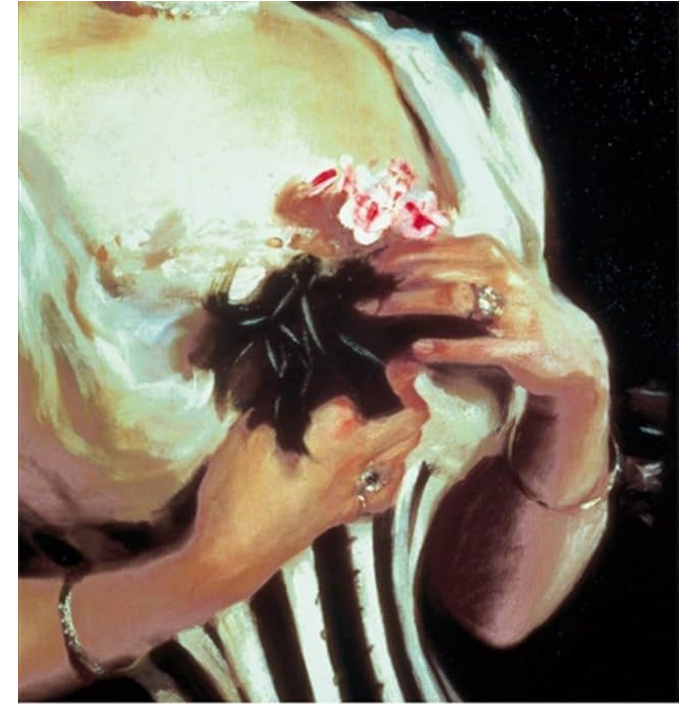
Tech is boring, let's talk literature now

[matrix]

- *Anna Karenina Principle*

“All happy families are alike; each unhappy family is unhappy in its own way.”

— Leo Tolstói, *Anna Karenina*



PENGUIN CLASSICS

LEO TOLSTOY

Anna Karenina

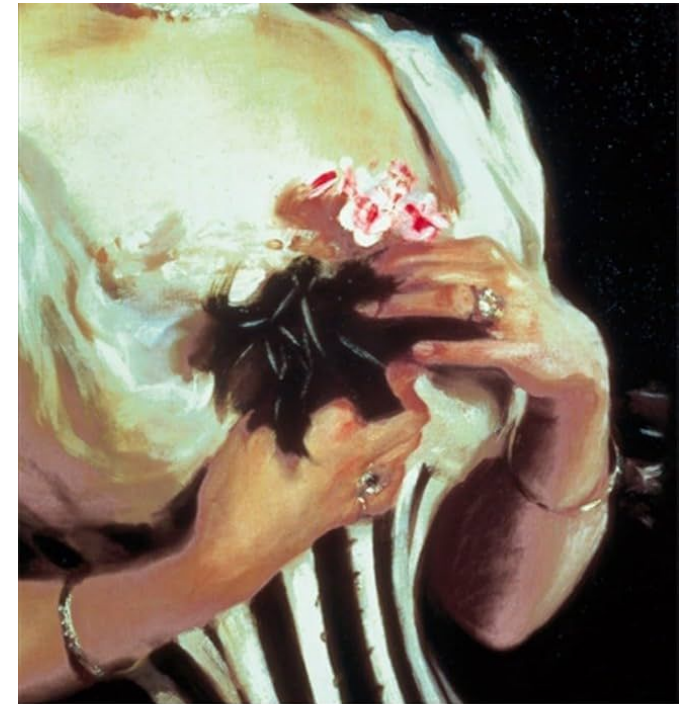
Encryption “for free”

- *Anna Karenina Principle*

“All happy families are alike; each unhappy family is unhappy in its own way.”

— Leo Tolstoi, *Anna Karenina*

- Ideally: encryption works 100% of the time.
- A complex process (like end-to-end encryption) that involves many complex sub-steps will fail if a *single step* fails.
 - Leading to messages “unable to decrypt” (UTD)
 - Resilience is learned, not a given!



PENGUIN CLASSICS

LEO TOLSTOY

Anna Karenina

The UTD Hunt

- Step 1: one can only improve what is observed and measured
- **UTD Hook:** new SDK component to keep track of UTDs
 - Observes UTDs and events decrypted “late” (transient UTDs)
 - Deduplicates them
 - Used for telemetry purposes:
 - How does the ratio of UTDs vs non-UTDs evolve over time?
 - How does a fix impact this ratio?

The UTD Hunt

- Step 2: investigate and fix the UTDs
 - Gigantic work by the crypto team at Element!
- Two real-world examples (no crypto knowledge involved)
 - To know who to send keys to, one must know who's in the room
 - Send an initial **/members** request on the Matrix server and update based on subsequent **m.room.member** events.
 - What if this initial request failed and...
 - it's never retried?
 - and a concurrent observer waited for it and assumed it succeeded?
- **More at Kegan's talk, tomorrow at 2:30pm.**

Invisible Crypto ([MSC4153](#))

- Goal: hold a cryptographic proof that the sender of a message is who they claim they are
- The gist:
 - mandate users to sign their devices with the cross-signing key
 - limit sending to a user who hasn't done that yet
 - ignore any message received from an insecure device
 - transition period: show insecure badge
 - plan and implementation still in flux
- **More on Valere's talk today at 2:15pm.**

OIDC support and QR code login

- Long-standing project to move from the custom Matrix authentication system to **OpenId Connect**
 - See also areweoidcyet.com
- Works with a [Matrix Authentication Service](#)
 - actual OIDC provider and/or specialized proxy to upstream provider
 - also written in Rust: reusing code is possible
- Full support in the SDK, including login via QR code
 - super nice UX!
 - cross-signing key and key backup for free!
- **experimental-oidc** Cargo feature

High-level components: Sync service

- A single service that spawns multiple syncs: one for encryption, one for room events.
- **Goal:** retrieve the simplicity of “fire the sync and forget about it”.
 - Minimal setup to benefit from all the best UX practices + best performance using simplified sliding sync!

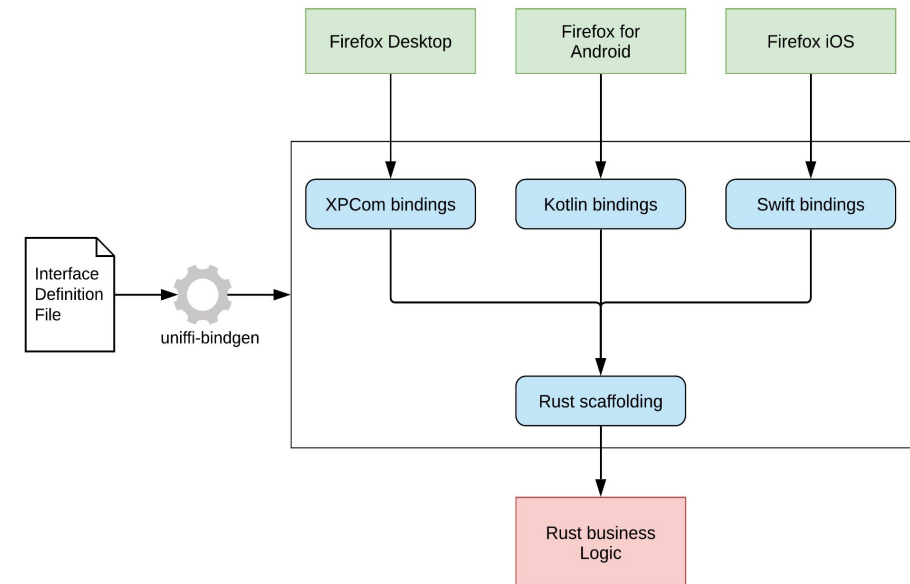
```
let sync_service = SyncService::builder(client.clone()).build().await?;  
sync_service.start().await;
```

High-level components: Timeline

- Now that we have a list of rooms and decrypted events, how do we display them?
- Enter the **Timeline API!**
- Room view *MVC* on steroids!
- Related events are aggregated into a single timeline item (reaction, read receipts, updates, redaction, etc.).
- Everything is *observable* in a reactive way.
 - Notifies when an item has been added/removed/updated.
- Recent developments: focus on a particular event (e.g. permalink), or only the *pinned* events.

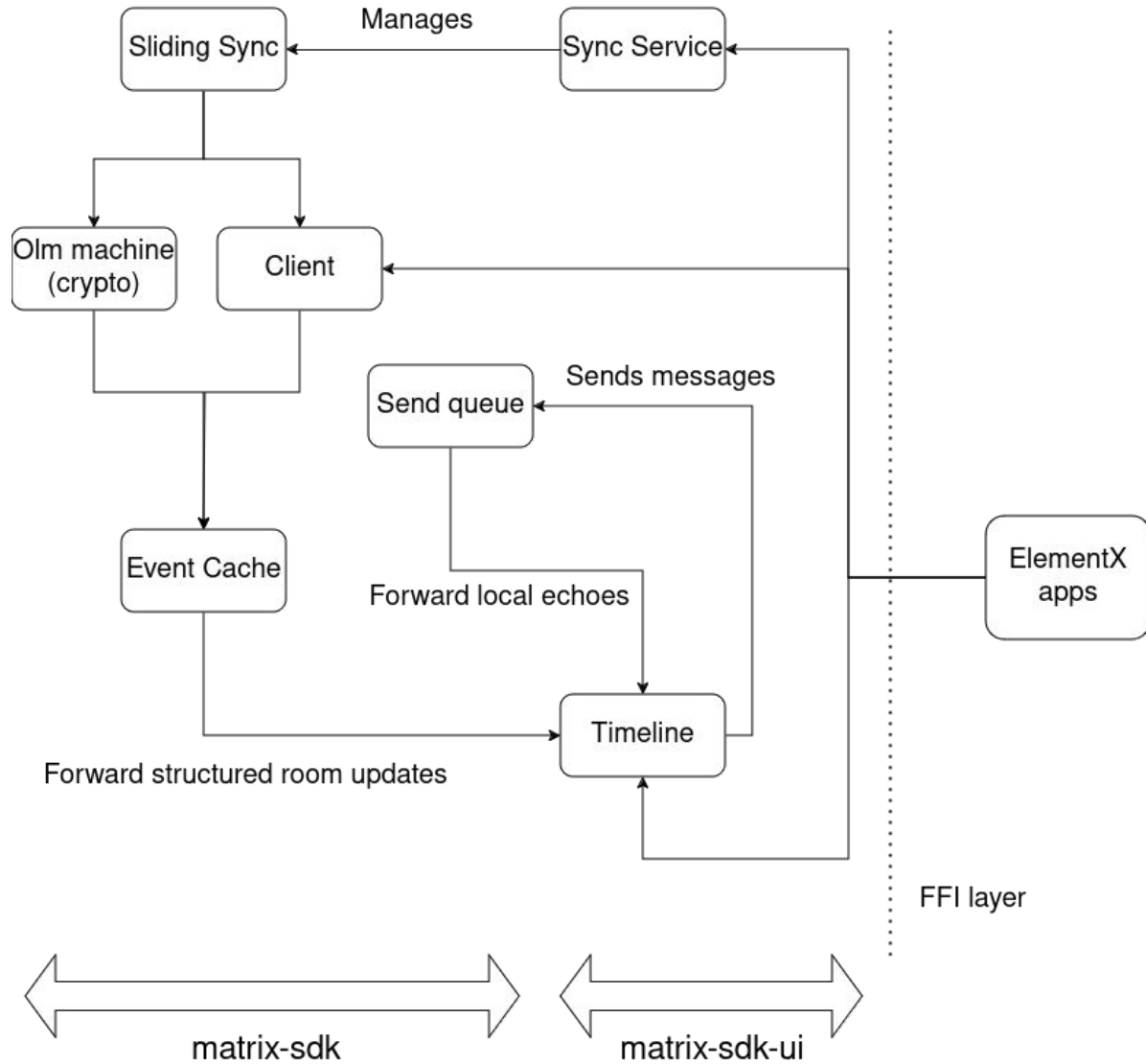
How is this all used in ElementX?

- Using [Mozilla's UniFFI](#), an automated bindings generator for Rust to and from other languages
 - FFI = **F**oreign **F**unction **I**nterface
- We generate bindings for Swift (iOS) and Kotlin (Android)
 - UniFFI can also generate bindings for Python and Go
- Requires integration with the foreign language's runtimes
- We added support for async code!
 - Simpler concurrent/background processing



A slightly disorienting yet complete map

[matrix]



Appendix: Reactive programming in Rust


[matrix]

- *Principle*: notify subscribers whenever an object / vector has changed.
- [Eyeball](#), one of our contributions to the Rust ecosystem.
- **Eyeball-im**: Diff based extension for collections.
 - only notify about the added/removed/updated item, not the entire collection.
- Extra querying facilities
 - batching, transactions
 - filtering, limiting
 - *sorting*

Some contribution stats

- Starting from September 1st, 2023
- **2884** commits from **1115** pull requests
- From **47** contributors
 - 14 of them did 10+ PRs (incl. 1 external contributor)
 - 7 with 50+ PRs (incl. same person)
 - 4 with 100+ PRs (all Element)

A big thank you!

- Thanks to all the contributors of the Rust SDK!
 - Shoutout to Kevin Commaille from Gnome/Fractal 
 - You can [contribute too!](#)
- Thanks to all who helped fund this work!

Thank you for listening!

[matrix]

Questions?



Fig 1: people asking questions about the Rust SDK.
Photo by [Raphael Bick](#) on [Unsplash](#)

mx room: [#matrix-rust-sdk:matrix.org](#)
[github.com/matrix-org/matrix-rust-sdk](#)
mx: @benjib:element.io