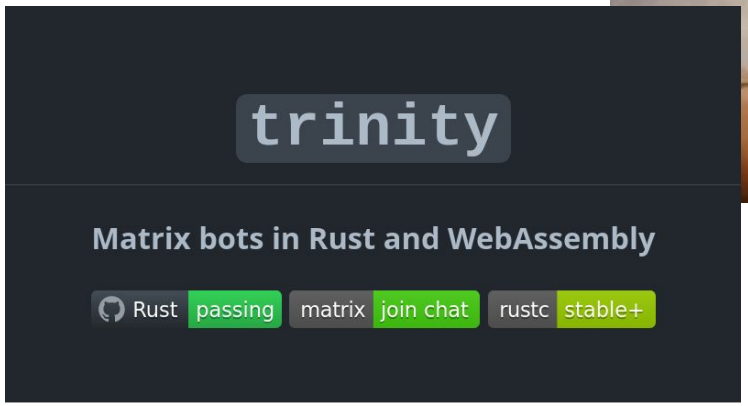
The state of the Rust SDK in 2023
FOSDEM 2024

mx: @bnjbvr:delire.party

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

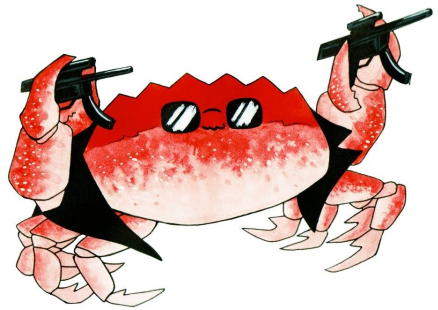
- Rust client-server API library
- github.com/matrix-org/matrix-rust-sdk
- Apache 2.0 license
- Everything one would expect from a Matrix client
  - logging in, out, reading/writing settings…
  - sending and receiving events
  - listening to sync updates and reacting to specific events via *handlers*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

Logo made by Ursa Johnson.

A history of the Matrix Rust SDK

RUMA

November 2015



October 2019

December 2021

Why Rust?

matrix

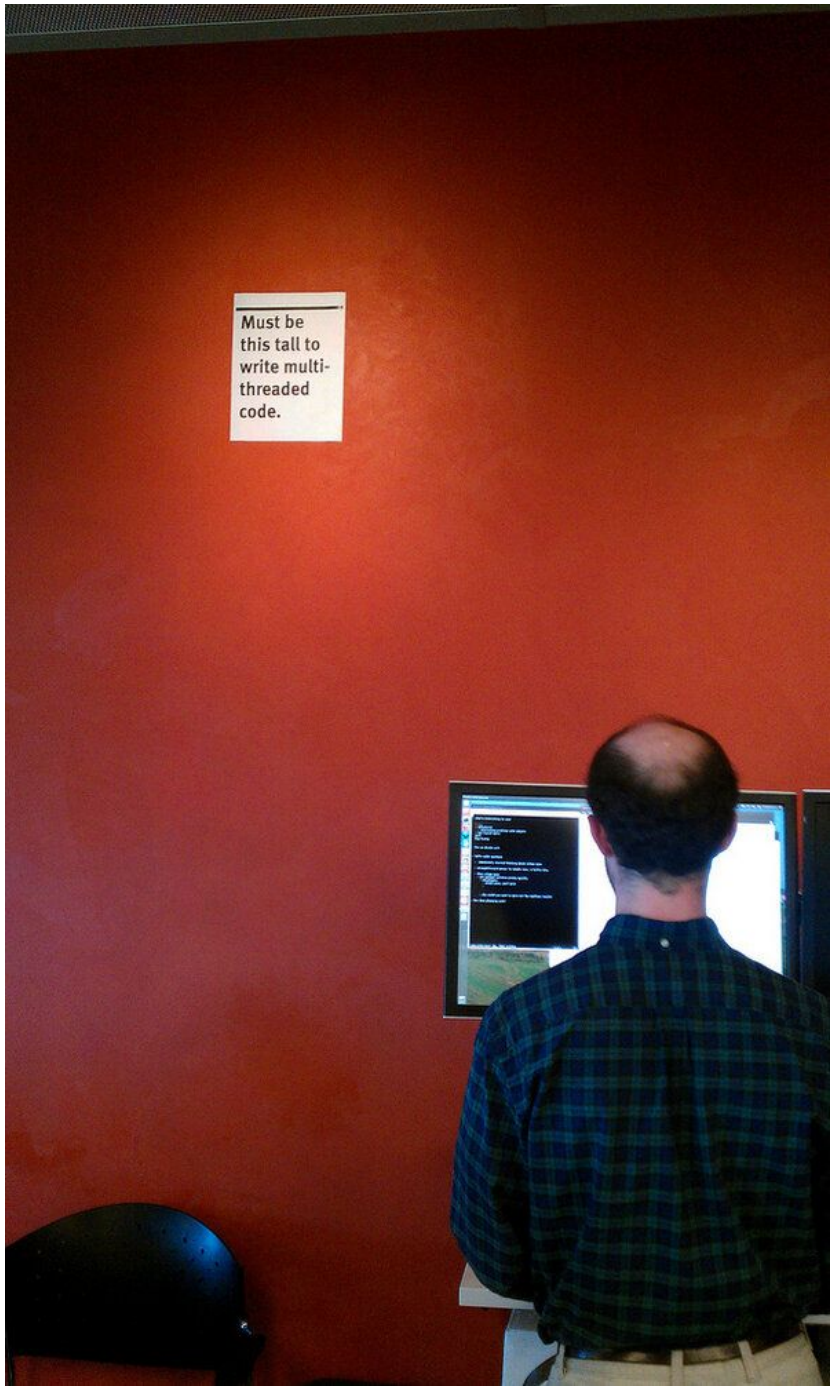
- Pleasantly high-level *and* super-fast
- Small memory footprint
- Secure, memory-safe
  - Thanks to the ownership model
- Amazing tooling and ecosystem 🥰
- Compatible with FFIs using the C ABI
- Empowerment!

Picture by @aldeka@wandering.shop

Why Rust?

Fearless concurrency!

Ownership model helps modeling concurrent ownership too.



matrix

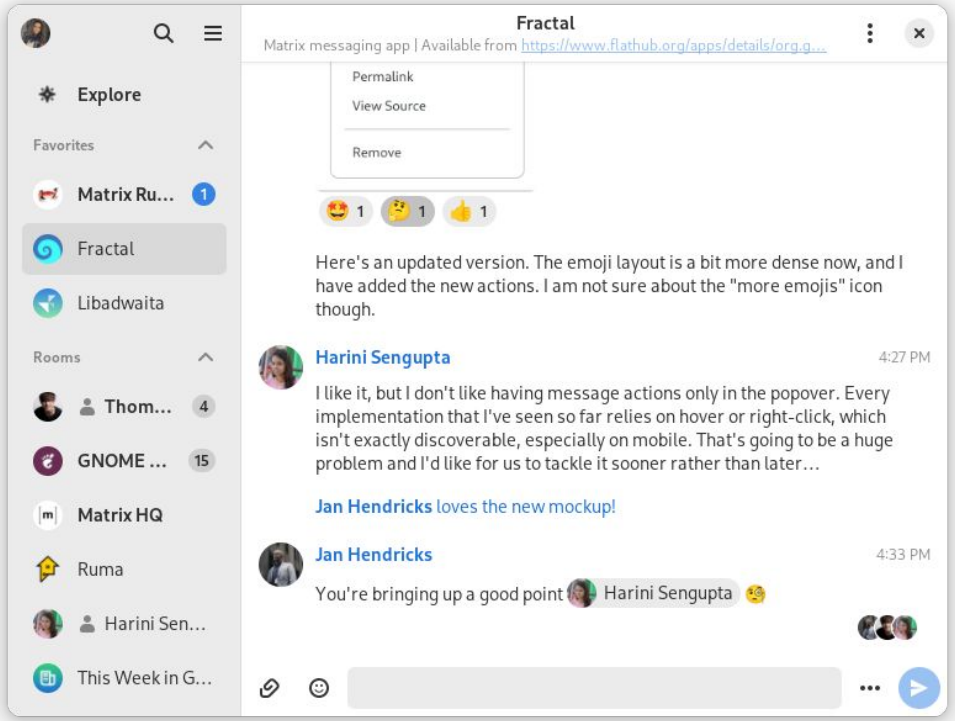
Why the Rust SDK?

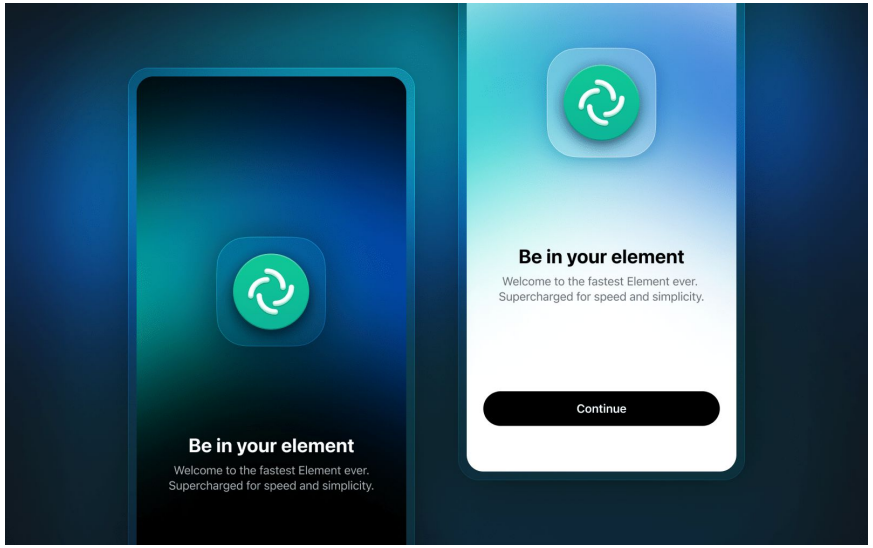
- Everything is reusable
  - a single crypto stack!
- High test coverage, fuzzing
- A single place where to add features and fix bugs

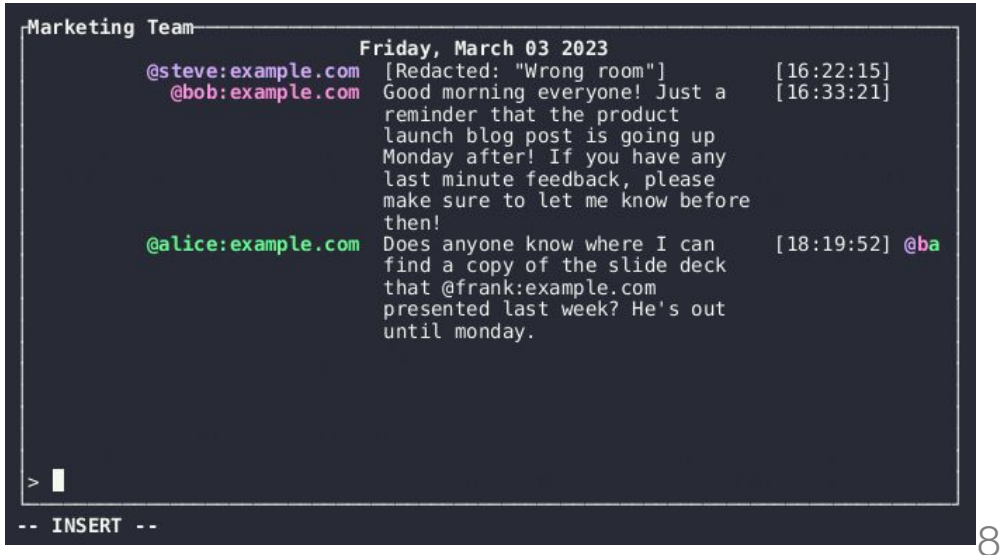
Fig 1: Figuring out SDKs at Element

Who’s using it?

- Fractal, a GTK Matrix client
- iamb, a TUI Matrix client
- ElementX apps
- *ElementR*: previous generation of Element apps







So what happened since last FOSDEM?

matrix

- New release **0.7.0** 🥳
  - all crates now have the same minor
- Published on January 5th, 2024
- Still not 1.0

Full support for MSC 3575: Sliding Sync!

matrix

- A proposal to replace `/sync` v2
- Subscribe to specific rooms *and* list of rooms
  - Receive a *sliding* window into those lists
- Modular design using opt-in extensions
  - typing notices, read receipts, encryption, account data.
- The SDK supports All The Things™
  - low-level, highly versatile API
- Feature-gated: `experimental-sliding-sync`

```rust
let sliding_sync = client
    .sliding_sync("room-list")?
    .with_account_data_extension(
        assign!(AccountDataConfig::default(), { enabled: Some(true) }),
    )
    .with_receipt_extension(assign!(ReceiptsConfig::default(), {
        enabled: Some(true),
        rooms: Some(vec![RoomReceiptConfig::AllSubscribed])
    }))
    .add_cached_list(
        SlidingSyncList::builder("all-rooms")
            .sync_mode(SlidingSyncMode::new_selective().add_range(0..=19))
            .timeline_limit(1)
            .required_state(vec![
                (StateEventType::RoomAvatar, "".to_owned()),
                (StateEventType::RoomEncryption, "".to_owned()),
                (StateEventType::RoomMember, "$LAZY".to_owned()),
                (StateEventType::RoomMember, "$ME".to_owned()),
                (StateEventType::RoomPowerLevels, "".to_owned()),
            ])
            .sort(vec!["by_recency".to_owned(), "by_name".to_owned()])
            .filters(Some(assign!(SyncRequestListFilters::default(), {
                is_invite: Some(false),
                is_tombstoned: Some(false),
                not_room_types: vec!["m.space".to_owned()],
            })))
            .bump_event_types(&[
                TimelineEventType::RoomMessage,
                TimelineEventType::RoomEncrypted,
                TimelineEventType::Sticker,
            ]),
    )
    .await?
    .build()
    .await?;

let sync = sliding_sync.sync();
pin_mut!(sync);

while let Some(update) = sync.next().await {
    // !!!
}
```

Support for OIDC

matrix

- 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
- The SDK implements enough of it to be able to use it in ElementX apps!
  - (Re)load metadata, register a new OIDC client, do the login flow
- `experimental-oidc` Cargo feature

New default storage backend!

matrix

- Storage backends can be implemented using *traits* (interfaces).
  - One store instance per logged-in user.
- Our previous default for on-disk persistence was using *sled*.
- Support for *sled* has been removed and replaced with *sqlite*.
- We still have an in-memory backend + an *indexeddb* backend when compiling for the Web with WebAssembly.

New cryptography features

- Secret Storage
  - An encrypted key/value store backed by the user account data.
  - Using a key generated or derived from the passphrase.
- Key backup and restoration
  - Store room keys into secret storage.
  - Restore those room keys from other devices.
- Automatic cross-signing identity bootstrapping
  - Used to verify one's own and other's devices.
  - Private keys are stored in secret storage too.

New high-level primitives!

- `matrix-sdk-ui` crate
- Highly experimental.
- Highly opinionated.
  - Sliding sync and OIDC are enabled by default.
  - By default, implements the best practices in terms of UX and performance.
- Same robustness and test guarantees as the rest of the SDK.
- Sliding Sync serves as a foundation.

Room List Service

matrix

- An observable lazy state machine that loads the list of all yours rooms
- Goal: **show something to the user as soon as possible**
- Set of heuristics to quickly get rendering updates:
  - Retrieve the latest event from 20 rooms to have *something* to display
  - Once that's done,
    - start fetching the latest event from all rooms in batches
    - start fetching invites in batches
    - load more events for all the *visible* rooms, as indicated by the app

Encryption and notifications

matrix

- **Encryption service**
  - Takes care of running *only* the encryption-related extensions.
  - Encryption can run concurrently with the room list sync loop!
    - Allows receiving encrypted events while e.g. scrolling a room list
- **Notification service**
  - Specialized client to handle push notifications
    - Given an event and a room identifiers, should the decrypted event generate a notification or not?
  - Makes use of the encryption service for e2ee rooms.
  - On iOS, the notification can be modified, but in a separate process
    - Cross-process global mutable state?!?1!?

Sync service

matrix

- Unified API that wraps and synchronizes a *Room List service* and an *Encryption service*.
- Goal: retrieve the simplicity of “fire sync and forget about it”.
  - Synchronize error states too.
  - Minimal setup to get all the good UX practices + best performance using sliding sync!

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

sync_service.start().await;
```

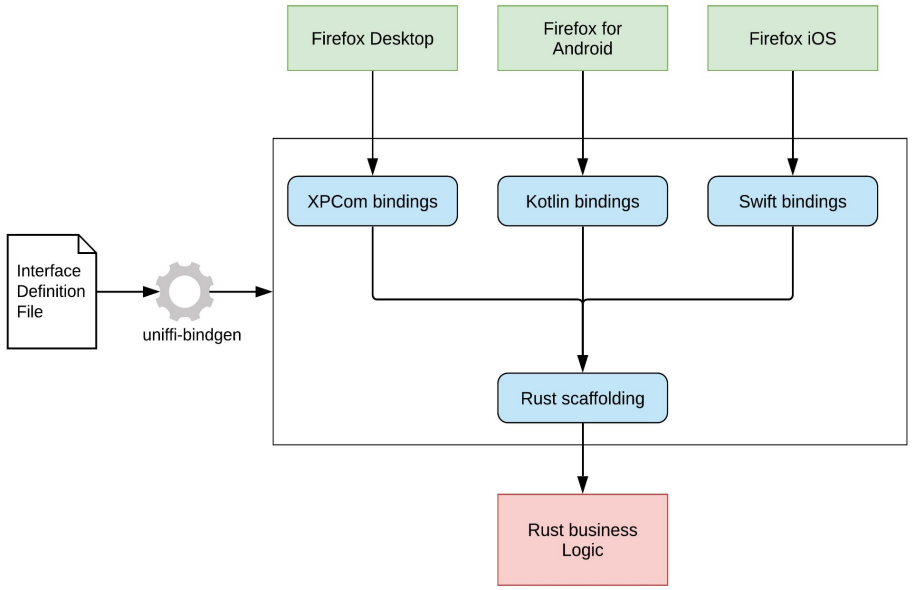
Timeline

matrix

- 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.).
  - Other freebies: handles local echoes, send the minimal read receipts, etc.
- Everything is *observable* in a reactive way.
  - Notifies when an item has been added/removed/updated.

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



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

Future Work™

matrix

- Eventually support all the major features a modern Matrix client would expect.
- Post-quantum cryptography (3DH step, compatible? with libsignal).
- Do more things client-side
  - End-to-end encryption limits what a server can compute.
    - Is a notification in an encrypted room worth reporting to the user?
    - Should a new event in an encrypted room mark the room as unread?
  - Only the client should be able to decrypt events!
    - Sort the room list client-side
    - Compute unread badges client-side

Thanks!

- Thanks to all the contributors of the Rust SDK!
  - Shoutout to Kevin Commaille from Fractal 💪
  - You can contribute too!
- You can support Matrix too!

matrix.org/support && matrix.org/membership

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: @bnjbvr:delire.party