



Robrix

A pure Rust multi-platform app
for **[matrix]** chat and beyond

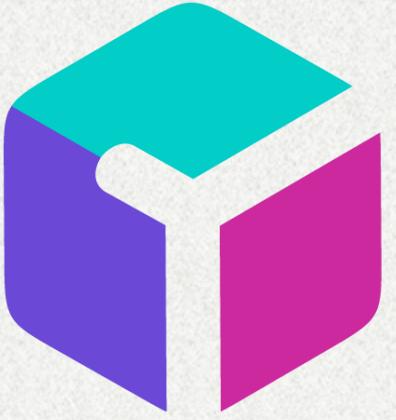
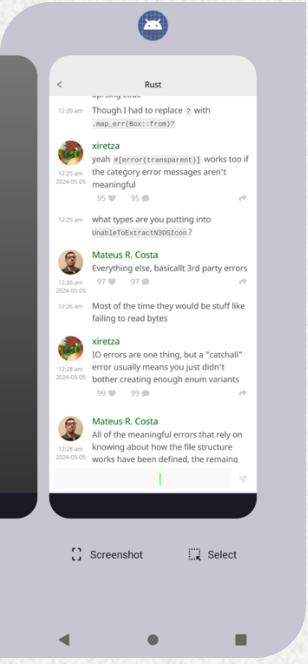
Kevin Boos

Tech Lead, Project Robius

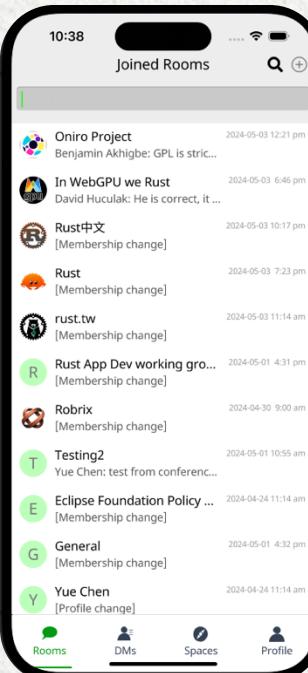
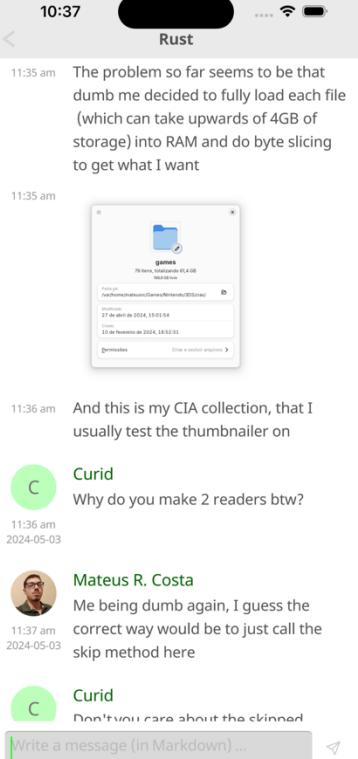
Futurewei Technologies

October 17, 2024

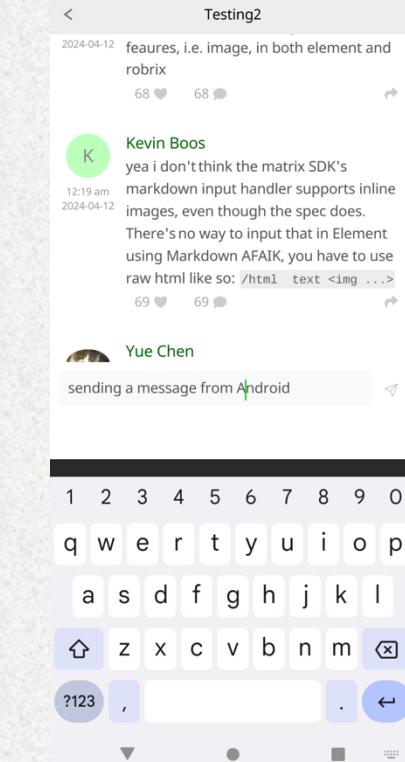
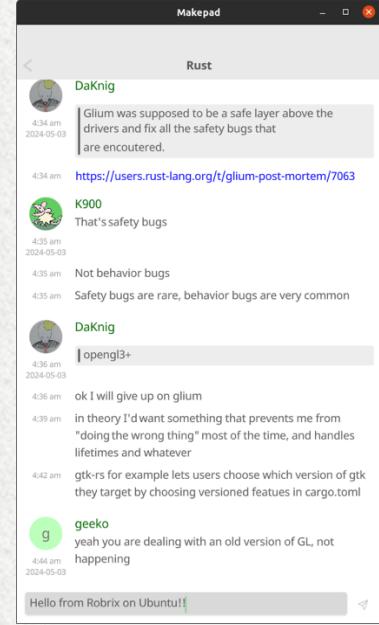
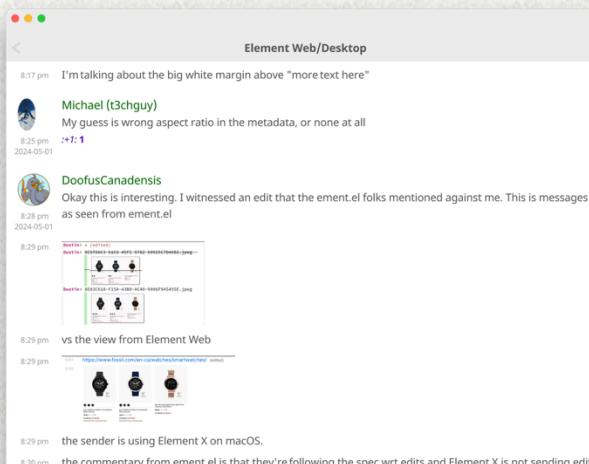




ROBRIX



One code base, many platforms



Emphasis
This is bold text
This is bold text
This is italic text
This is italic text
~~Strikethrough~~
Blockquotes
Blockquotes can also be nested...
...by using additional greater-than signs right next to each other...
...or with spaces between arrows.

Lists
Unordered

- Create a list by starting a line with +, -, or *
- Sub-lists are made by indenting 2 spaces:
 - Marker character change forces new list start:
 - Ac tristique libero volutpat at
 - Facilisis in pretium nisl aliquet
 - Nulla volutpat aliquam velit
 - Very easy!

Ordered

1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa
4. You can use sequential numbers...
5. ...or keep all the numbers as 1.

Start numbering with offset:

57. foo
58. bar

Code

```
Write a message (in Markdown) ...
```

Motivation

Why Robrix? Why Project Robius?



1. Create a multi-platform app dev experience fully in Rust

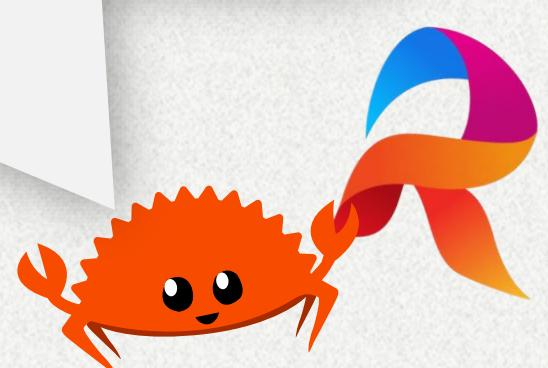
Devs
can:

- leverage the robust, safe, and performant Rust ecosystem
- avoid dealing with multiple languages/environments
- never write a single line of platform-specific code

2. Make Mobile a first-class concern in the Rust community

- the Rust experience on Mobile has felt neglected

Project Robius at a glance



Background – Project Robius app dev framework



- Fully open-source, decentralized, and community-driven
 - Independent collaborators across US, Europe, Aus/NZ, S. America, China
- All components written entirely in Rust

But why Robius? ... there are already great Rust UI toolkits out there

- egui, Iced, Slint, Leptos, Tauri, Dioxus, Makepad, Xilem, Freya, Ribir, ...
 - There's **more to app dev** than just drawing a GUI!
- Traditionally, Rust apps have been “less Rust, more other code”
(a small Rust core surrounded by big platform-specific wrappers)

Why *Robrix*, though?

1. Needed a key app to drive Robius development
 - “What kind of app is highly **demanding**?“ (of UI & platform abstractions?)
 - A modern chat client!
 - Requires network, geolocation, file/image access, audio/video capture & playback, system notifications, clipboard, rich text formatting & input, persistent device storage & cache, connectivity mgmt, biometric auth, secret key storage, and so much more!
 - Why Matrix?
 - Shares our values: open-source, decentralized, community-driven, and now using Rust

Why *Robrix*, though?

1. Needed a key app to drive Robius development

- A modern chat client is quite demanding
 - Matrix aligns with our values & goals

2. But then... since we've already started, why stop there?

Let's go further!

- Can we make a super Matrix client for power users?
- Can we make Robrix into a central “hub” for federated services?
 - Focus on the needs of open-source community & developers
 - Combine Matrix, ActivityPub, local AI LLMs, source code tools, etc.

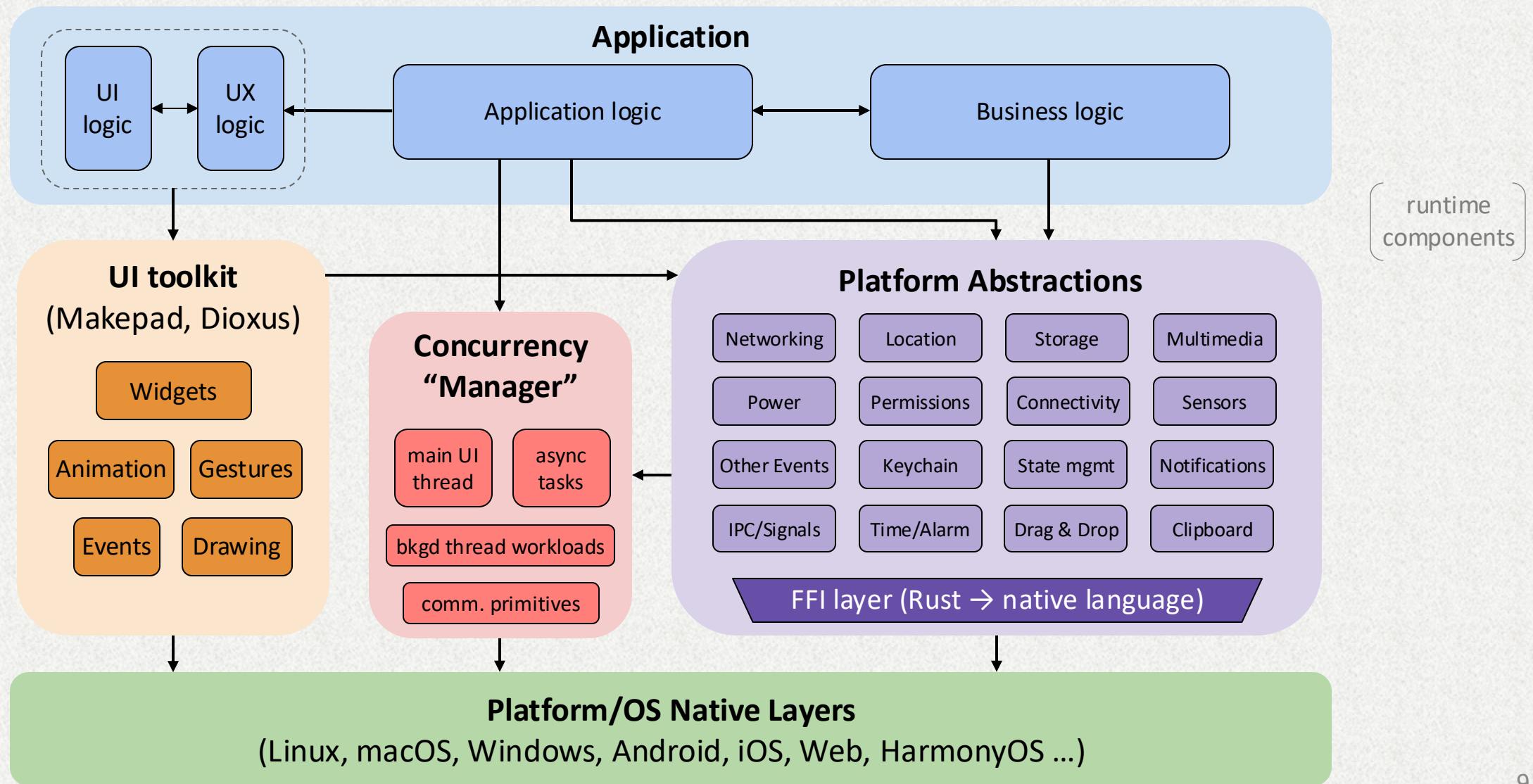
The better the app, the stronger the case for Rust app dev!

Goals & Roadmap

(select high-level goals)



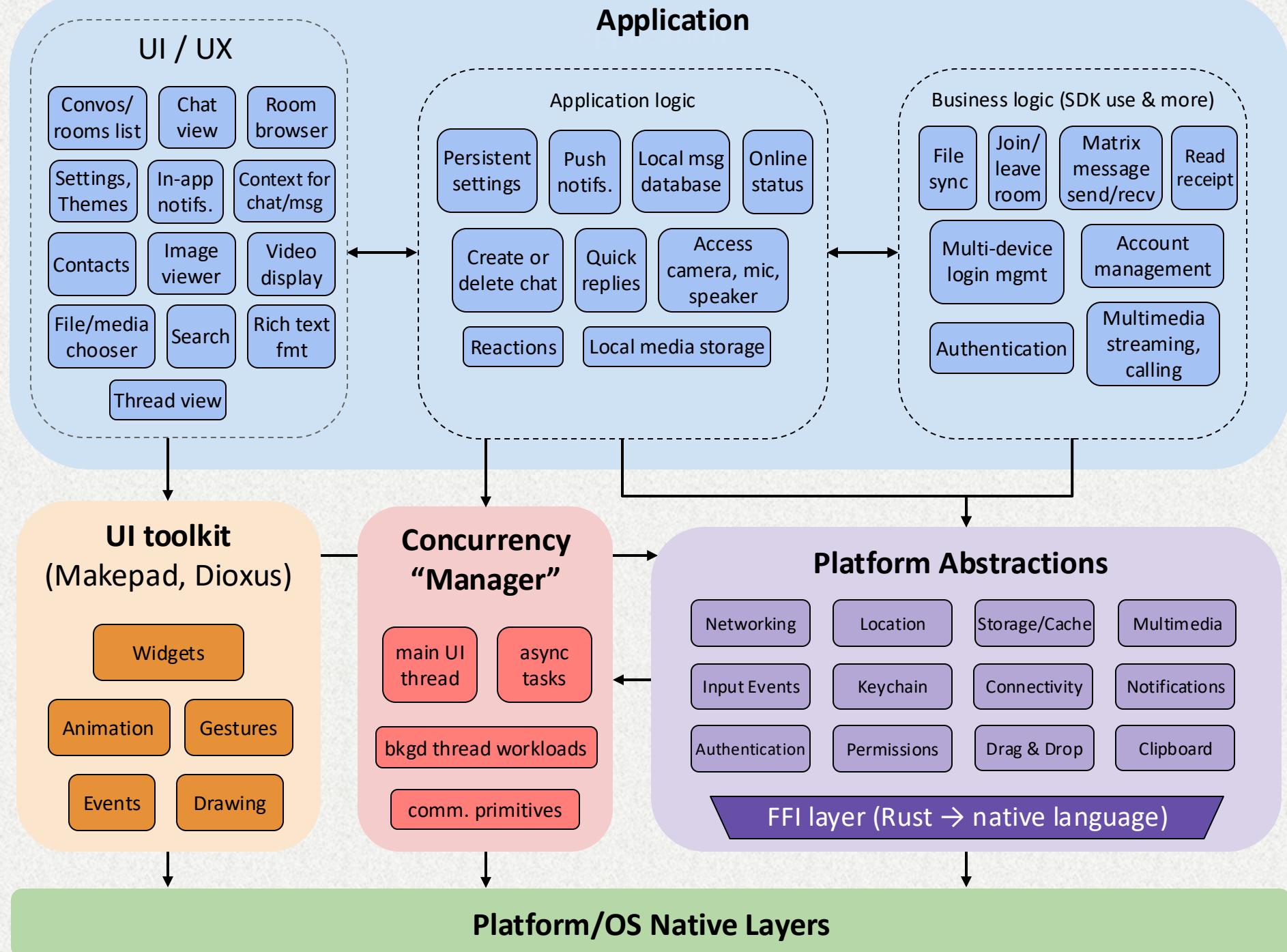
Primary goal: create reference design of full system stack



Robrix App Architecture

(Matrix parts)

- Robrix's needs guide Robius development and implementation priority
- Writing an actual app forces us to dogfood Robius (and Rust UI toolkits)





Meta-goals for Robrix as a flagship Robius app



- Well-structured, well-commented code



- Easy to fork and modify
 - Serves as a ready-made **complex app template**
 - Easy to contribute features back to upstream



- Publish detailed walkthrough that guides a dev through how to make a (simplified) Robius app



- Exemplify the utility & power of Robius framework
 - Demonstrate how nice & flexible Makepad apps can be

Roadmap



Stage 1: publish app with fundamental Matrix client features

- Standard messaging, login, app persistence, E2EE support
- Continue building full App Dev ecosystem and UI toolkit



Stage 2: Matrix client for “power” users

- Feature parity with existing major clients (incl. administrative features)
- Responsive UI design with side-by-side multi-room view
- Local LLM runtime for AI chat bots & conversation summaries
 - Key point: does not jeopardize E2EE or data sovereignty



Stage 3: central “hub” for federated & open-source services

- Collect multiple services: chat, discussion forums, code + issues + PRs, announcements/microblogs, feed of notifications/activity/news
- Identity management via integration with OpenWallet

Roadmap: Stage 1

Fundamental Matrix chat features



Stage 1: the basics



- Realize Matrix client fundamentals
 - Login, rooms list (sliding sync), basic messaging, annotations, replies, rich text formatting, images, E2EE support, app persistence
- Publish alpha version of Robrix
 - Requires enhanced build tooling for releasing, signing, packaging

- Complete platform support

Host OS	Target Platform	Builds?	Runs?
macOS	macOS	✓	✓
macOS	Android	✓	✓
macOS	iOS	✓	✓
Linux	Linux	✓	✓
Linux	Android	✓	✓
Windows	Windows	✓	✓
Windows	Android	✓	✓

- View list of joined rooms
- View timeline of events in a single room
- Fetch and display room avatars
- Fetch user profiles (displayable names)
- Fetch and display user profile avatars
- Backwards pagination (upon viewing a room timeline)
- Dynamic backwards pagination based on scroll position/movement: [#10](#)
- Loading animation while waiting for pagination request: [#109](#)
- Stable positioning of events during simple timeline update
- Stable positioning of events during complex/multi-part timeline update
- Display simple text-only messages
- Display image messages (PNG, JPEG)
- Rich text formatting for message bodies
- Display reactions (annotations)
- Handle opening links on click
- Linkify plaintext hyperlinks
- Reply previews above messages: [#82](#)
- Send messages (standalone, no replies)
- Interactive reaction button, send reactions: [#115](#)
- Reply button, send reply: [#83](#)
- Re-spawn timeline as focused on an old event after a full timeline clear:
- Display multimedia (audio/video/gif) message events: [#120](#)
- Collapsible/expandable view of contiguous "small" events: [#118](#)
- E2EE device verification, decrypt message content: [#116](#)



Some basics to save for Stage 2 (early 2025 beta)

Auxiliary/admin features: login, registration, settings

- Persistence of app session to disk: [#112](#)
- Username/password login screen: [#113](#)
- SSO, other 3rd-party auth providers login screen: [#114](#)
- Side panel showing detailed user profile info (click on their Avatar)
- Ignore and unignore users (see known issues)
- User settings screen
- Dedicated view of spaces
- Dedicated view of direct messages (DMs): [#139](#)
- Link previews beneath messages: [#81](#)
- Keyword filters for the list of all rooms: [#123](#)
- Search messages within a room: [#122](#)
- Room browser, search for public rooms
- Room creation
- Room settings/info screen



Features that don't drive
Project Robius development

Significant implementation challenges

- Exposing & abstracting platform APIs / OS services
 - Handling complex multi-step builds
 - Rust compilation: invoking & configuring Cargo
 - Standardizing compilation of non-Rust system glue layers
 - Supporting platform-specific toolchains and linking conventions
 - Defining and executing pre- and post-compilation steps
 - Signing, bundling, packaging, distributing apps
 - Required for system notifications, location (and so much more)
 - Managing concurrency: multi-threading, async vs. sync, hetero CPUs...
 - UI & other select workloads must run on “main” thread
 - Certain APIs *require* async, others *forbid* async
 - Contending with multiple *vastly* different UI toolkits
 - Platform feature crates must be easy to integrate with UI system internals
- 
- all across
disparate
platforms

Targeted platform abstractions / OS service APIs

- System notifications
- Biometrics & authentication
- File/image picker dialogs
- Context menus (OS-native)
- Menu bar (desktop only)
- Rich clipboard
- Drag & Drop
- Default URI/Intent handling
- Content sharing to foreign app
- Native font access
- Input Method Editors (IME)
- Native gesture recognition
- Connectivity management
 - WiFi, Bluetooth, NFC, nearby devices
- GPS/Location access
- Multimedia capture
 - Device discovery & configuration
 - Camera – snapshots, video, settings
 - Audio – input, MIDI, playback
- Sensors
- Native alarms, timers
- Keychain (secret storage)
- Power management/status
- App lifecycle state transitions
- OS-standard data directories

and so much more ...

Current status (autumn 2024)

- System notifications
- Biometrics & authentication
- File/image picker dialogs
- Context menus (OS-native)
- Menu bar (desktop only)
- Rich clipboard
- Drag & Drop
- Default URI/Intent handling
- Content sharing to foreign app
- Native font access
- Input Method Editors (IME)
- Native gesture recognition
- Connectivity management
 - WiFi, Bluetooth, NFC, nearby devices
- GPS/Location access
- Multimedia capture
 - Device discovery & configuration
 - Camera – snapshots, video, settings
 - Audio – input, MIDI, playback
- Sensors
- Native alarms, timers
- Keychain (secret storage)
- Power management/status
- App lifecycle state transitions
- OS-standard data directories

and so much more ...

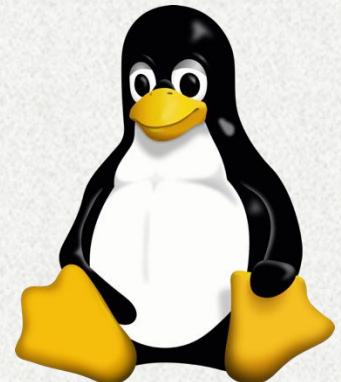
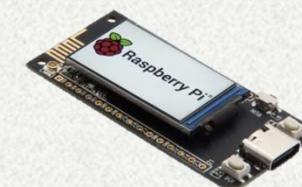
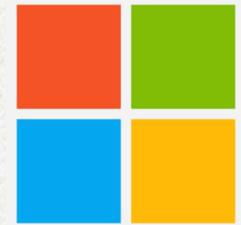
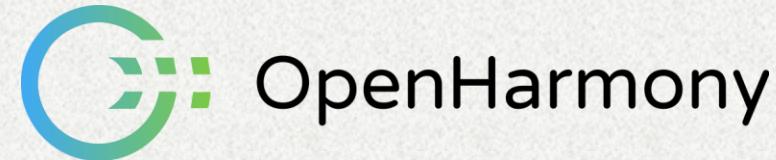
Project Robius contributions used in real apps

- robius-location
- robius-keychain (will upstream to keyring-rs)
- robius-android-env
- android-build
- robius-directories (fork of directories plus Android)
- robius-open
- robius-authentication
- robius-url-handler
- cargo-packager
- Makepad



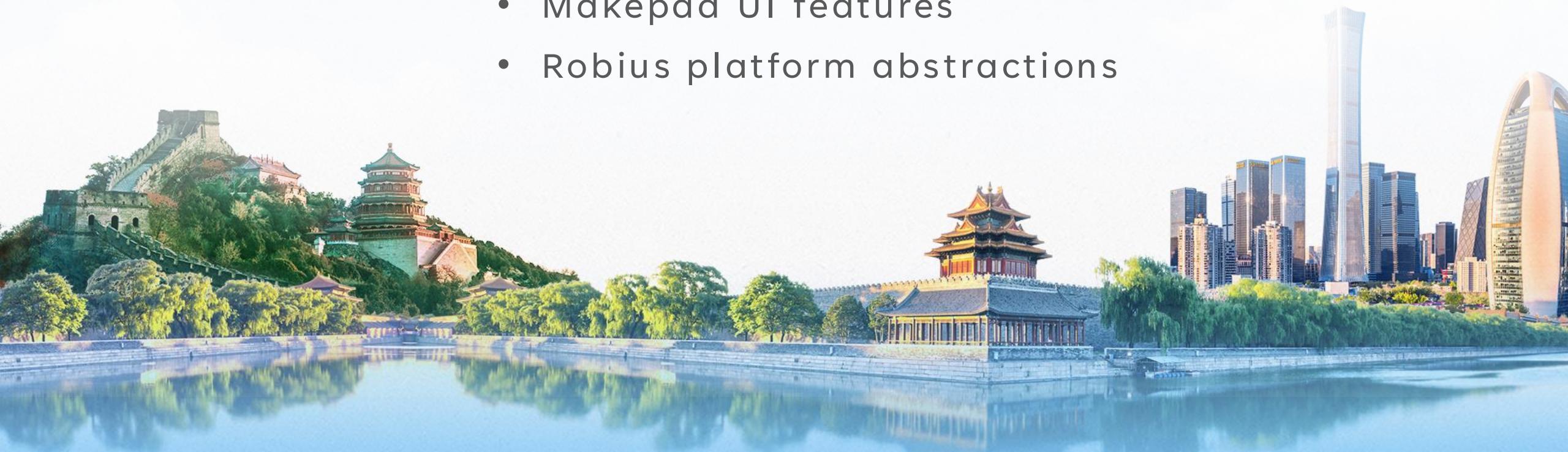
Platforms of interest

- Desktop
 - macOS
 - Linux (Debian-based, Arch)
 - Windows
- Mobile
 - Android
 - iOS
 - [soon] OpenHarmony OS
- Web, wasm
- Others?
 - Select microcontrollers/peripherals
 - tvOS, watchOS



Demo time!

- Makepad UI features
- Robius platform abstractions



<demo>

for future/offline viewers:

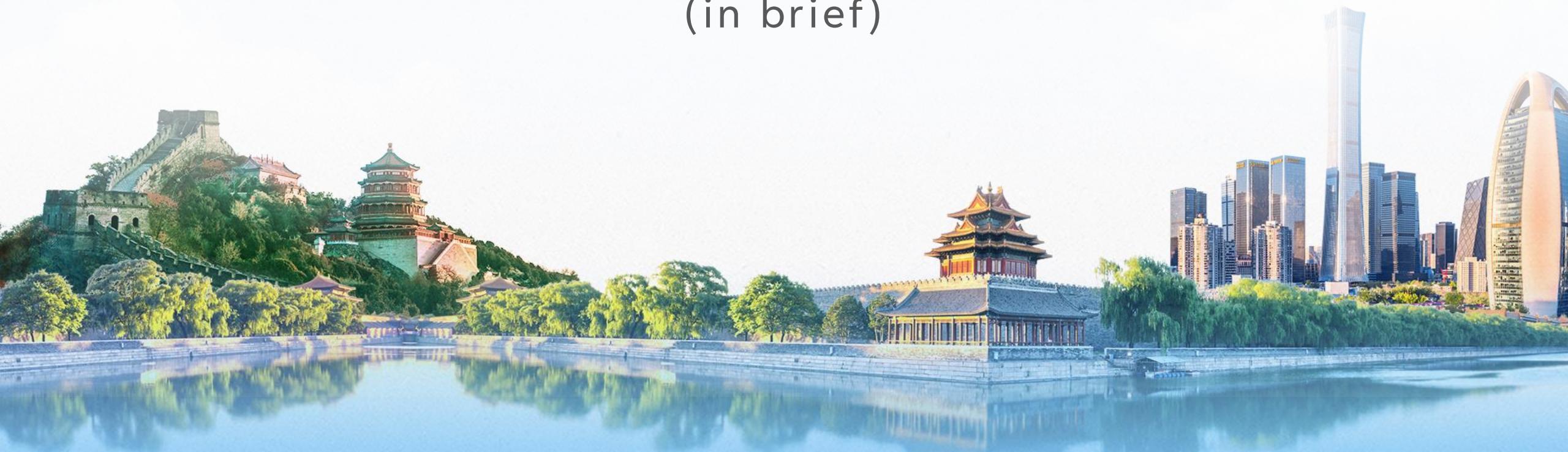
the demo included an overview of Robrix running on macOS and Android.

Check out our git repo and run it on your own machine here:

<https://github.com/project-robius/robrix>

Select Challenges

(in brief)



Robrix development revealed a concurrency challenge

A typical first attempt to fetch and draw an image

```
impl Widget for TimelineView {  
    fn draw_walk(&mut self, cx: &mut Cx2d, scope: &mut Scope, walk: Walk) -> DrawStep {  
        while let Some(item_id) = self.list.next_visible_item(cx) {  
            let Some(timeline_item) = self.tl_items.get(tl_idx) else { ... };  
            if let TimelineItemKind::Event(event_tl_item) = match timeline_item.kind() {  
                if let TimelineItemContent::Message(message) = event_tl_item.content() {  
                    if let MessageType::Image(image) = message.msgtype() {  
                        let (item, existed) = list.item(cx, item_id, live_id!(ImageMessage))?  
                        let Some(mimetype, width, height) = image.info.as_ref() else { return };  
                        let text_or_image_widget = item.text_or_image(id!(content.message));  
                        let media_request = MediaRequest { source: image.source, format: ... };  
                        let data → matrix_client.media().get_media_content(&media_request, true).await?;  
                        text_or_image_widget.show_image(|img| utils::load_png_or_jpg(&img, cx, &data));  
                        ...  
                    }  
                }  
            }  
        }  
    }  
}
```

error[E0728]: `await` is only allowed inside `async` functions and blocks

|
1048 | fn draw_walk(
| ----- this is not `async`
...
1132 | let data = matrix_client.media().get_media_content(&media_request, true).await?;

only allowed inside `async` functions and blocks ^^^^^



Robrix development revealed a concurrency challenge

Ok, let's try
the request in
a spawned
async task

```
impl Widget for TimelineView {  
    fn draw_walk(&mut self, cx: &mut Cx2d, scope: &mut Scope, walk: Walk) -> DrawStep {  
        while let Some(item_id) = self.list.next_visible_item(cx) {  
            let Some(timeline_item) = self.tl_items.get(tl_idx) else { ... };  
            if let TimelineItemKind::Event(event_tl_item) = match timeline_item.kind() {  
                if let TimelineItemContent::Message(message) = event_tl_item.content() {  
                    if let MessageType::Image(image) = message.msgtype() {  
                        let (item, existed) = list.item(cx, item_id, live_id!(ImageMessage))?;  
                        let Some(mimetype, width, height) = image.info.as_ref() else { return };  
                        let text_or_image_widget = item.text_or_image(id!(content.message));  
                        let media_request = MediaRequest { source: image.source, format: ... };  
                        let data -> tokio::spawn(async {  
                            matrix_client.media().get_media_content(&media_request, true).await?  
                        });  
                        text_or_image_widget.show_image(|img| utils::load_png_or_jpg(&img, cx, &data));  
                        ...  
                }  
            }  
        }  
    }  
}
```

thread 'main' panicked at src/home/room_screen.rs:1128:25:

there is no reactor running, must be called from the context of a Tokio 1.x runtime

Robrix development revealed a concurrency challenge

Obvious: *must not block the main UI thread*

- Invoking a blocking (sync) function from the main thread will hang the UI
 - Especially if it causes a syscall, where the OS may put the thread to sleep

Less obvious: **async functions can also block the main thread**

- Even if the function is async, that native thread could still be blocked while the async executor is polling the future being awaited
- Same issue can occur on background threads, too (but that's ok)

Key point:

- Don't invoke **any** blocking functions, even “non-blocking” async
 - (on the main UI thread)
- Results in a very performant UI, as you saw in the demo

Robrix development revealed a concurrency challenge

```
impl Widget for TimelineView {  
    fn draw_walk(&mut self, cx: &mut Cx2d, scope: &mut Scope, walk: Walk) -> DrawStep {  
        while let Some(item_id) = self.list.next_visible_item(cx) {  
            let Some(timeline_item) = self.tl_items.get(tl_idx) else { ... };  
            if let TimelineItemKind::Event(event_tl_item) = match timeline_item.kind() {  
                if let TimelineItemContent::Message(message) = event_tl_item.content() {  
                    if let MessageType::Image(image) = message.msgtype() {  
                        let (item, existed) = list.item(cx, item_id, live_id!(ImageMessage)).unwrap();  
                        let Some(mimetype, width, height) = image.info.as_ref() else { return };  
                        let text_or_image_widget = item.text_or_image(id!(content.message));  
                        if let MediaSource::Plain(mxc_uri) => &image.source {  
                            match self.media_cache.try_get_media_or_fetch(mxc_uri.clone(), None) {  
                                MediaCacheEntry::Loaded(data) => {  
                                    text_or_image_widget.show_image(|img| utils::load_png_or_jpg(&img, cx, &data));  
                                }  
                                MediaCacheEntry::Requested => {  
                                    text_or_image_widget.set_text(&format!("Fetching image from {:?}", mxc_uri));  
                                }  
                                MediaCacheEntry::Failed => {  
                                    text_or_image_widget.set_text(&format!("Failed to fetch image from {:?}", mxc_uri));  
                                }  
                            } ...  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

Solution:

A concurrency-aware impl that requests an

async function to be run in a different context

```
MediaCacheEntry::Requested => {  
    text_or_image_widget.set_text(&format!("Fetching image from {:?}", mxc_uri));  
}  
MediaCacheEntry::Failed => {  
    text_or_image_widget.set_text(&format!("Failed to fetch image from {:?}", mxc_uri));  
}  
}
```



Robrix development revealed a concurrency challenge

```
// Tries to get the media from the cache, or submits an async request to fetch it.  
//  
// This method *does not* block or wait for the media to be fetched,  
// and will return `MediaCache::Requested` while the async request is in flight.  
// If a request is already in flight, this will not issue a new redundant request.  
pub fn try_get_media_or_fetch(&mut self, mxc_uri: OwnedMxcUri ...) -> MediaCacheEntry {  
    let value_ref = match self.entry(mxc_uri.clone()) {  
        Entry::Vacant(vacant) => vacant.insert(MediaCacheEntry::Requested),  
        Entry::Occupied(occupied) => return occupied.get()...,  
    };  
  
    async_ctx::submit_async_request(  
        MatrixRequest::FetchMedia {  
            media_request: mxc_uri.into(),  
        },  
    );  
    ...  
}
```



/// Submits a request to the worker thread
/// to be executed in an async context.

```
pub fn submit_async_request(req: MatrixRequest) {  
    REQUEST_SENDER.get().send(req);  
}
```

Robrix development revealed a concurrency challenge

```
// Submits a request to the worker thread  
// to be executed in an async context.  
pub fn submit_async_request(req: MatrixRequest) {  
    REQUEST_SENDER.get().send(req);  
}
```

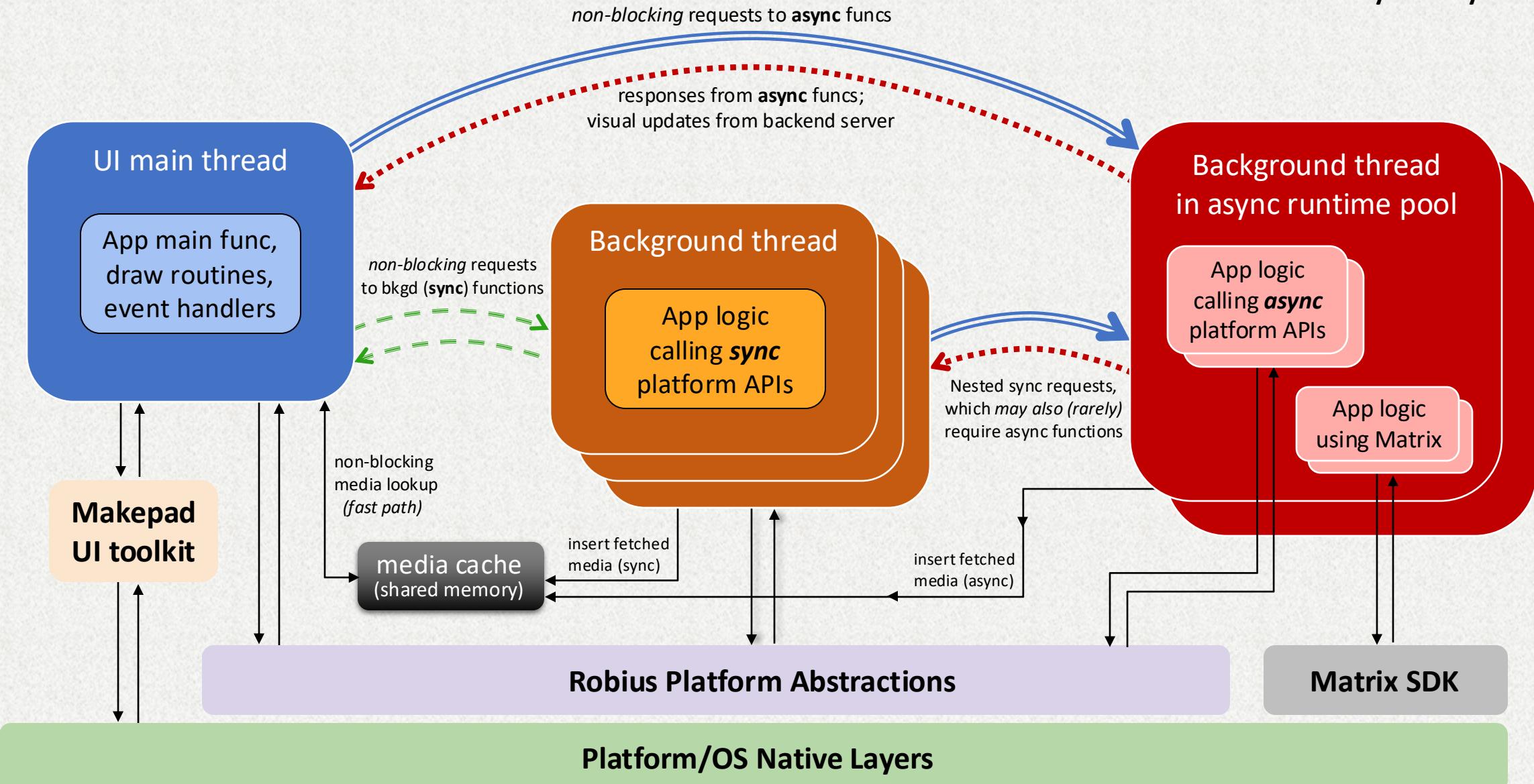
```
pub enum MatrixRequest {  
    FetchMedia {  
        media_request: MediaRequest,  
        on_fetched: fn(&Mutex<MediaCacheEntry>, MediaRequest, ...),  
        destination: Arc<Mutex<MediaCacheEntry>>,  
        update_sender: crossbeam_channel::Sender<TimelineUpdate>,  
        ...  
    } }
```

```
async fn async_worker(mut receiver: UnboundedReceiver<MatrixRequest>) -> Result<()> {  
    while let Some(request) = receiver.recv().await {  
        match request {  
            MatrixRequest::FetchMedia { media_request, on_fetched, destination, update_sender } => {  
                Handle::current().spawn(async move {  
                    let img_data = client.media().get_media_content(&media_request, true).await;  
                    on_fetched(&destination, media_request, img_data, update_sender);  
                });  
            };  
        } } }
```

This same pattern is used for back pagination, sending messages, etc

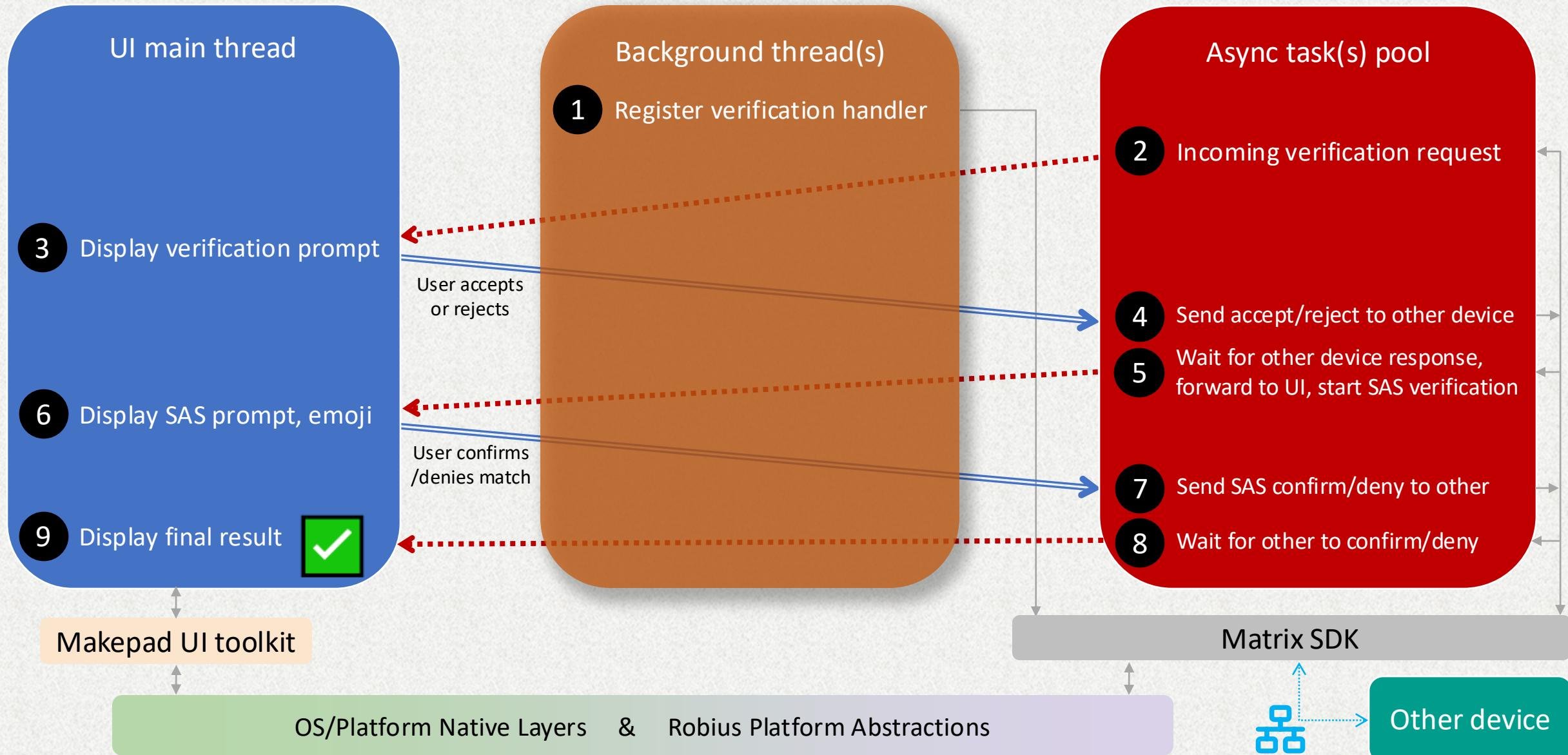
Generic solution for mixed concurrency contexts

→ direct function call
- - - → sync → sync channel
→ sync → async channel
- - - → async → sync channel



More challenging example: device verification

→ direct function call
- - - → sync → sync channel
→ sync → async channel
- - - → async → sync channel



<demo>

for future/offline viewers:

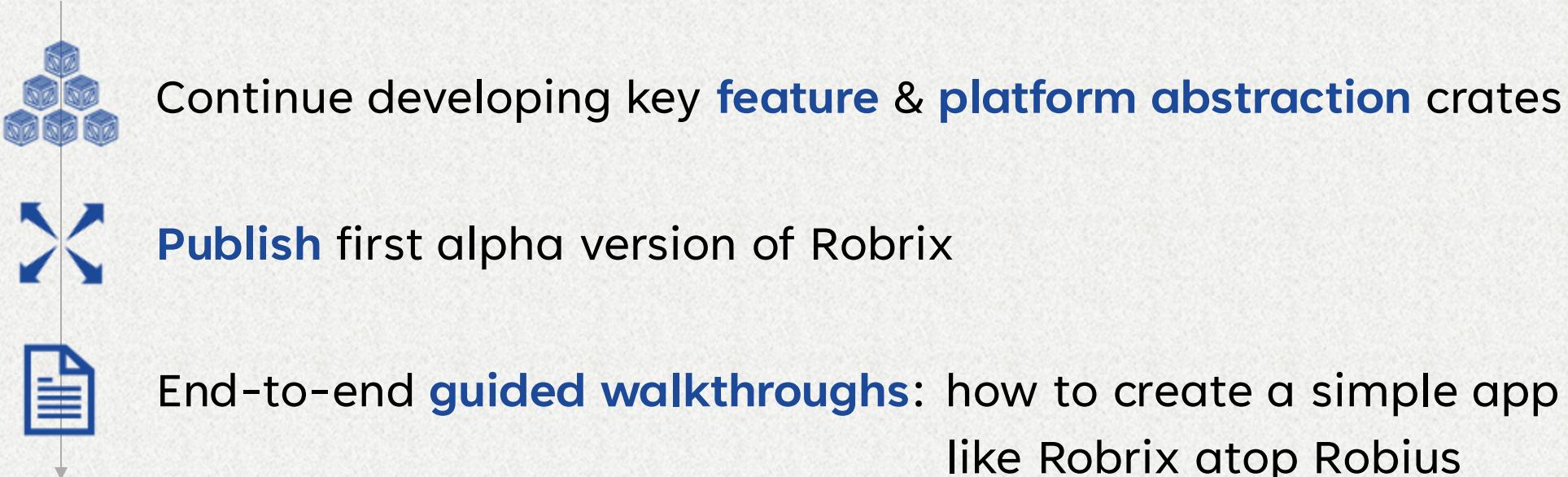
the demo included an example of device verification working between Robrix and the Element Desktop client.

Check out our git repo and run it on your own machine here:
<https://github.com/project-robius/robrix>

Stage 1: summary + looking forward

- Robrix (and Robius) are only just getting started!
 - Many diverse challenges remain across UI, platform, build, tooling
 - Active collaboration with UI toolkits to drive daily improvements

2024 Roadmap



Future Roadmap Stages

Stage 2: Matrix for power users

Stage 3: Fediverse community hub



Stage 2 – a Matrix client for power users

A. Feature parity with existing clients

B. Responsive UI with side-by-side panes & dockable tabs

- Visually beautiful and customizable across desktop, tablet, foldable, mobile
- Fast navigation with keyboard-driven interaction mode

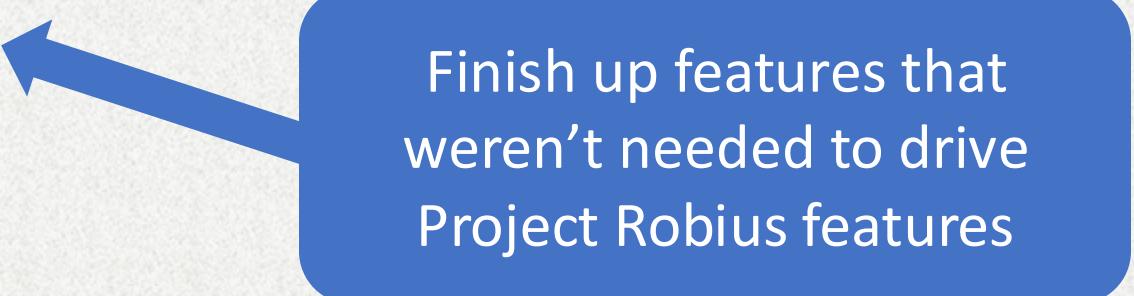
C. Integration with *local* LLMs to leverage AI benefits

- Automated topic analysis, chat synopses of “what you missed”, etc.
- Chatbot response channels to help newcomers in large open-source communities
- Local LLM runtime preserves E2EE & data sovereignty

A. Reach feature parity with existing clients

Auxiliary/admin features: login, registration, settings

- SSO, other 3rd-party auth providers login screen
- Dedicated view of spaces
- Dedicated view of direct messages (DMs)
- Search messages
- Room browser / search
- Room creation
- Room settings/info screen
- Room members pane
- User profile settings screen



Finish up features that weren't needed to drive Project Robius features

Futurewei

People +

- Alex Robins**
- Patryk
- Yue
- Kristie
- David

Channels +

- python 39
- test-dev 12
- neutron-dev 26

Search

python **Alex Robins**

Brad Hugh 04/09/2024 11:06 AM
Hi! I'm locked out of my account for our testing env. Can you help sort this out for me?

Alex Robins 04/09/2024 12:22 PM
Yeah no problem. Can you send me your email and github username? I'll need some help from [@Emily](#) for tracking down the ID you use so feel free to ping her as well. Thanks!

Can you send me your email and github username?

Brad Hugh 01/09/2024 12:22 PM
Here you go:

Github.com
Github: @Brad_Hugh
Developer with various interests including python, rust, javascript, etc.

test-dev **Kristie**

Kristie 04/11/2024 09:24 AM
Due to a time limit issue of only 3 seconds in sending responses in SlacksOps, the permission modal will now be a plain text message. Is this ok?

Alex Robins 04/11/2024 11:43 PM
It's unclear which behavior you're referring to. Can you ping me separately so I can investigate and better understand what you're talking about? In the past we had no such issues so unless the API has changed, I don't understand.

Today

Brad Hugh 04/12/2024 07:15 AM
As [@Kristie](#) mentioned, this is in fact a real issue. I've tried to find a workaround but I've got nothing yet. Let's take a look together later today when you have a moment.

David 04/12/2024 09:52 AM
Hi All, I found a solution. I'll push a PR and share an update once I've confirmed 100% it works.

B. Multi-pane dockable tab UI

(the original design proposal)

Robius: App Dev in Rust

+ Message Alex Robins

+ Message test-dev

40

Robrix's actual (brand new) "IDE" UI

- Auto-adapts to screen sizes
- Same code works on Desktop, Mobile, Tablet...

Unlock productivity for work/social chats (just like an IDE)

The image shows a screenshot of the Robrix Matrix client interface. On the left, there is a sidebar with various room and user icons. The main area has three tabs: "Home", "Element Web/Desktop", and "Robrix". The "Element Web/Desktop" tab is active, showing a conversation with "Matrix Rust SDK". The message reads:

12:29 pm Even the successful cults dont have a bus factor of 1. I mean i can name a cult that has persisted for a while with a bus factor thats not 1 and with international reach.

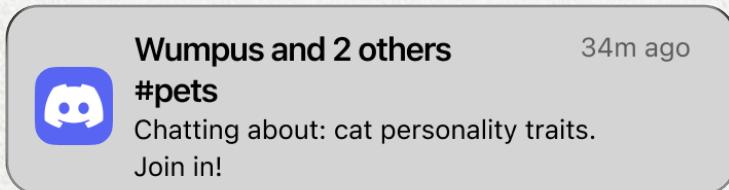
Below this, there is a "New Messages" section with a message from "@joshsimmons:matrix.org":

entirely unrelated to the previous topic, I'm pleased to offer this charismatic tuxedo cat on this auspicious Friday:

12:56 pm

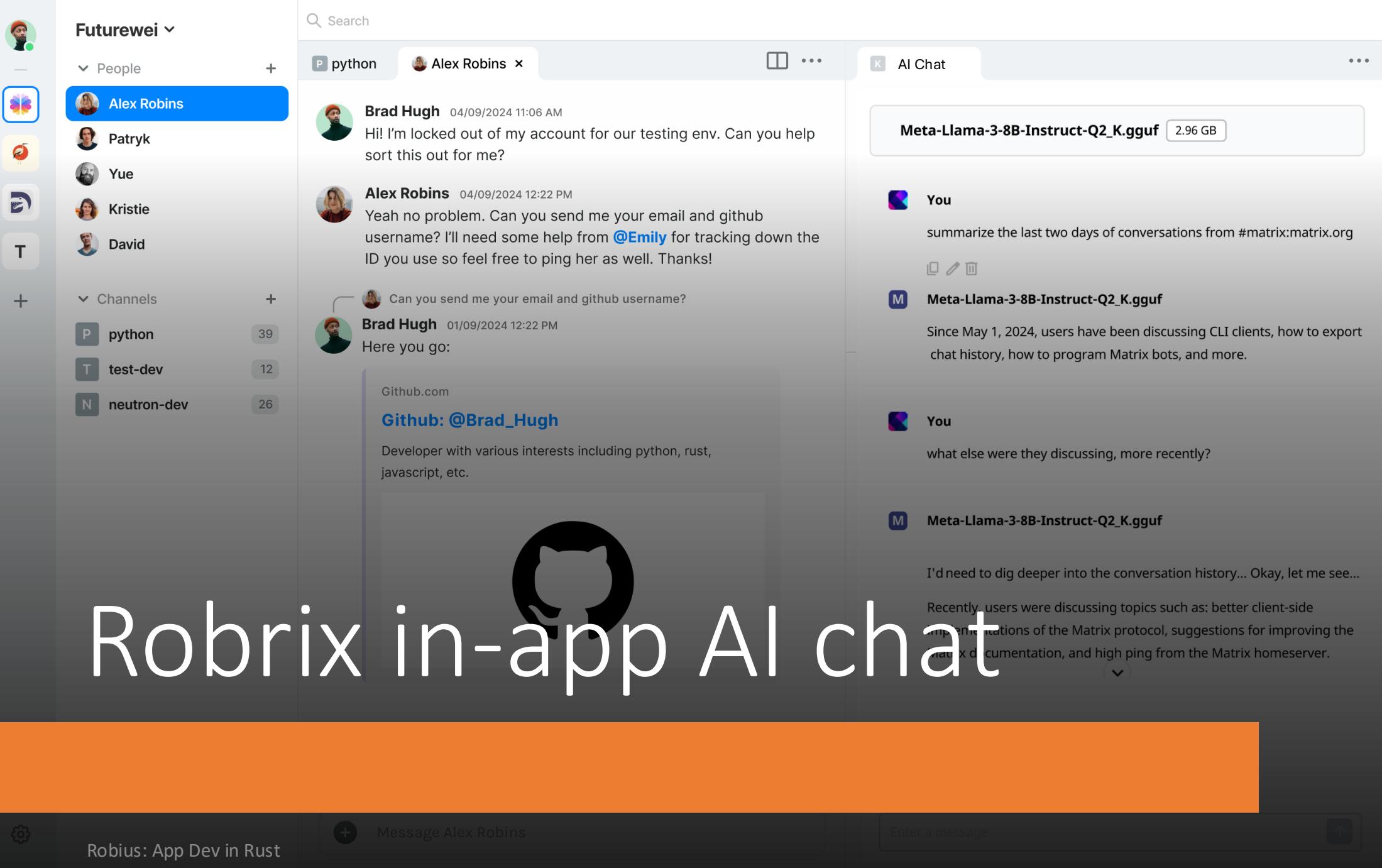
At the bottom of the screen, there is a message input field with placeholder text: "Write a message (in Markdown) ...".

C. AI features – local LLM integration via Moxin



← Discord's new chat topic summaries

- Wouldn't it be great to have these for Matrix chats?
 - But how can this work without giving an AI service access to all of your data?
- **Local LLMs** — benefit from AI without jeopardizing E2EE, data sovereignty
 - “**What’s happening**”: AI-driven conversation summaries & topic analysis
 - “**Action items**”: AI identifies important messages & to-dos from a long backlog
 - Automated chatbots for **handling public FAQs** in an open-source project community
- Longer-term: share LLM agents’ recipes and bot demos via Matrix
 - Currently not offered by existing tools like ChatGPT



Robrix in-app AI chat

Futurewei

People: Alex Robins, Patryk, Yue, Kristie, David

Channels: python (39), test-dev (12), neutron-dev (26)

Search: python, Alex Robins

AI Chat: Meta-Llama-3-8B-Instruct-Q2_K.gguf (2.96 GB)

Message Alex Robins: Message sent at 04/09/2024 11:06 AM

Alex Robins: Hi! I'm locked out of my account for our testing env. Can you help sort this out for me?

Alex Robins: Yeah no problem. Can you send me your email and github username? I'll need some help from [@Emily](#) for tracking down the ID you use so feel free to ping her as well. Thanks!

Can you send me your email and github username?

Brad Hugh: 01/09/2024 12:22 PM
Here you go:

Github.com
[Github: @Brad_Hugh](#)
Developer with various interests including python, rust, javascript, etc.

Meta-Llama-3-8B-Instruct-Q2_K.gguf: 2.96 GB

You: summarize the last two days of conversations from #matrix:matrix.org

Meta-Llama-3-8B-Instruct-Q2_K.gguf: Since May 1, 2024, users have been discussing CLI clients, how to export chat history, how to program Matrix bots, and more.

You: what else were they discussing, more recently?

Meta-Llama-3-8B-Instruct-Q2_K.gguf: I'd need to dig deeper into the conversation history... Okay, let me see...
Recently, users were discussing topics such as: better client-side implementations of the Matrix protocol, suggestions for improving the Matrix documentation, and high ping from the Matrix homeserver.

AI model is user's choice

Stage 3 – a community hub (for open-source devs)

Chat (Matrix)

The screenshot shows the Mastodon web interface. At the top, there's a search bar with the placeholder "Search or paste a URL". Below it, a sidebar on the left lists the user "Kevin Boos" (@kevinboos) and a link "What's on your mind?". The main content area has a header "Back" with a back arrow icon. A search result for "RustNL" is shown, with the handle "@rustnl@hostodon.org" and a profile picture. The post text reads: "'Robust, Immersive and Seamless Multi-platform App Development in Rust' is the talk @kevinboos will give at RustNL 2024." To the right of the post, the timestamp "Mar 29" is displayed. Below the post, a link "Details here: 2024.rustnl.org/speakers/" is provided. The interface includes a sidebar with navigation links: Home, Notifications, Explore, Live feeds, Private mentions, Bookmarks, Favorites, and Lists. Another search result for "#rust2024" is shown below, featuring a profile picture of Kevin Boos and the text "Kevin Boos - Architect & Developer of Threisus.CS & Principal Architect @ Futureless". The timestamp "Mar 27" is shown to the right. A link "Details here: 2024.rustnl.org/speakers/" is also present. The bottom of the screen shows a footer with links: "#rust2024", "#hostodon.org", and "#weedegefif".

Microblogs (Mastodon)

Lemmy Communities Create Post Create Community ☰

Q Login Sign Up

Posts Comments Subscribed Local All Active ↺

literally no clue
media.infosec.exchange
@Masinatuutu@lemmy.ee to Memes • 23 hours ago
1.91K ▲ 148 ↴

We're doomed
lemmy.world
@hypertown@lemmy.world to Memes • 2 days ago
1.37K ▲ 64 ↴

survival optional.
fedit.de
Hofmaiaer to Memes • 1 day ago
1.08K ▲ 130 ↴

TIL You can use `systemd-analyze plot > plot.svg` to plot the service startup time to find bottlenecks ☰
644 ↺

Trending communities

Landlord Love
Tankie
asonix pokes Lemmy
Pikmin Bloom
Beeper
lemmy-meter.info

Create a Community

Explore Communities

Lemmy ☰

A community of privacy and FOSS enthusiasts, run by Lemmy's developers

Community Forums (Lemmy)

-  Activity
-  Chat Unread only
-  Teams
-  Calendar
-  Calls
-  OneDrive
-  ...
-  Mats Lundgren mentioned 11:54 AM
Kevin Boos Do we plan to bring Robi...
Robius Internal
-  Mats Lundgren mentioned 11:41 AM
you
Kevin Boos Yue Chen Edward Tan Go...
Robius Internal
-  Mats Lundgren mentioned 4/29/2023
you
Looks great Kevin Boos. Can we add ...
Robius Internal
-  Yu Chen reacted to your 4/25/2023
message
not sure if markdown supports text c...
Robius Internal

Notifications, activity pane

Code [re]view (git)

Writing is telepathy ➔ Evergreen notes

Ideas / Writing is telepathy

Writing is telepathy

#evergreen

From [On Writing](#)

Ideas can travel through time and space

Ideas can travel through time and space without being uttered out loud. The process of telepathy requires two places:

- A sending place, a transmission place — where the writer sends ideas, such as a desk
- A receiving place — where the reader receives the ideas/imagery such as a couch, a comfortable chair, in bed

Quote

Look, here's a table covered with red cloth. On it is a cage the size of a small fish aquarium. In the cage is a white rabbit with a pink nose and pink-rimmed eyes. On its back, clearly marked in blue ink, is the number 12.72.

Books

On Writing

Calmness is a superpower

Writing is telepathy

garion to your former self

Evergreen notes turn ideas into objects that you can manipulate

Everything is a remix

Chasm

Telepathy is a superorganism

Creativity is combinatory uniqueness

Evergreen notes

1 bookmark 206 words 1139

Notes

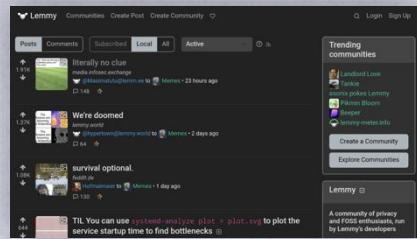
The screenshot shows the LinkedIn News feed with several cards displayed:

- Media Mentions** (16h ago): "Wrestlemania could harness larger advertiser network after Endeavor acquisition" by Display.
- Media Mentions** (6h ago): "Publishers test generative AI tools to boost SEO" by Display.
- Media Mentions** (6h ago): "Why advertisers are still waiting on the CTV promised land" by Display.
- Media Mentions** (16h ago): "Google search quality ratters shift focus to chatbot and source ranking" by Search Engine Land.
- Automation** (16h ago): "SPICED: A Uniform Framework Across the Entire Customer Journey (Article 3 of 3)" by MarketingRiffs Daily.
- Automation** (16h ago): "What's the impact of AI means for businesses with Katie Couric and James Manyika" by Think with Google.
- Automation** (21h ago): "Why CTV is the next big branding channel for sports advertisers" by Display.
- Automation** (1d ago): "AI makes its mark at Havas Media Group" by Digital Day.

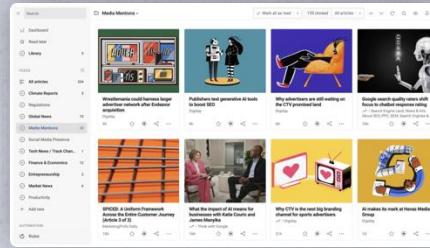
News feeds (RSS)



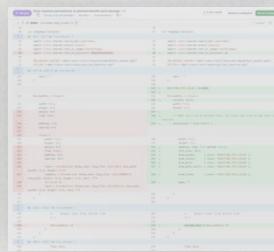
Microblogs
(Mastodon)



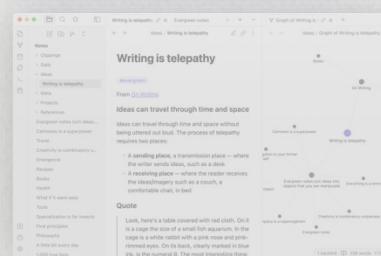
Community Forums
(Lemmy)



News feeds (RSS)



Code [re]view
(git)



Notes



Chat
(Matrix)

Collect multiple federated services in a single experience

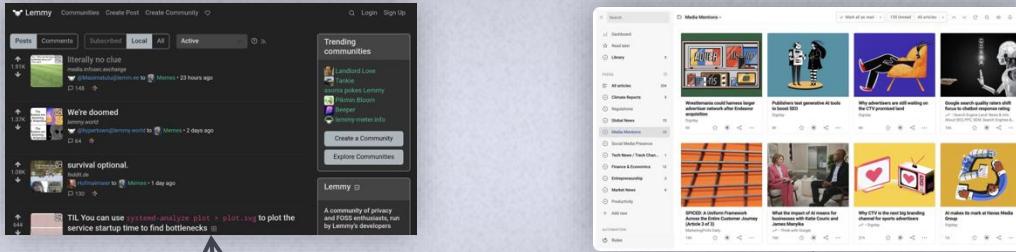
ROBRIX



Potential for powerful combo features & actions

Identity mgmt. via OpenWallet

- Decentralized ID provider
- *[long-term]* integrate into Matrix specification



OpenWallet
FOUNDATION

ROBRIX

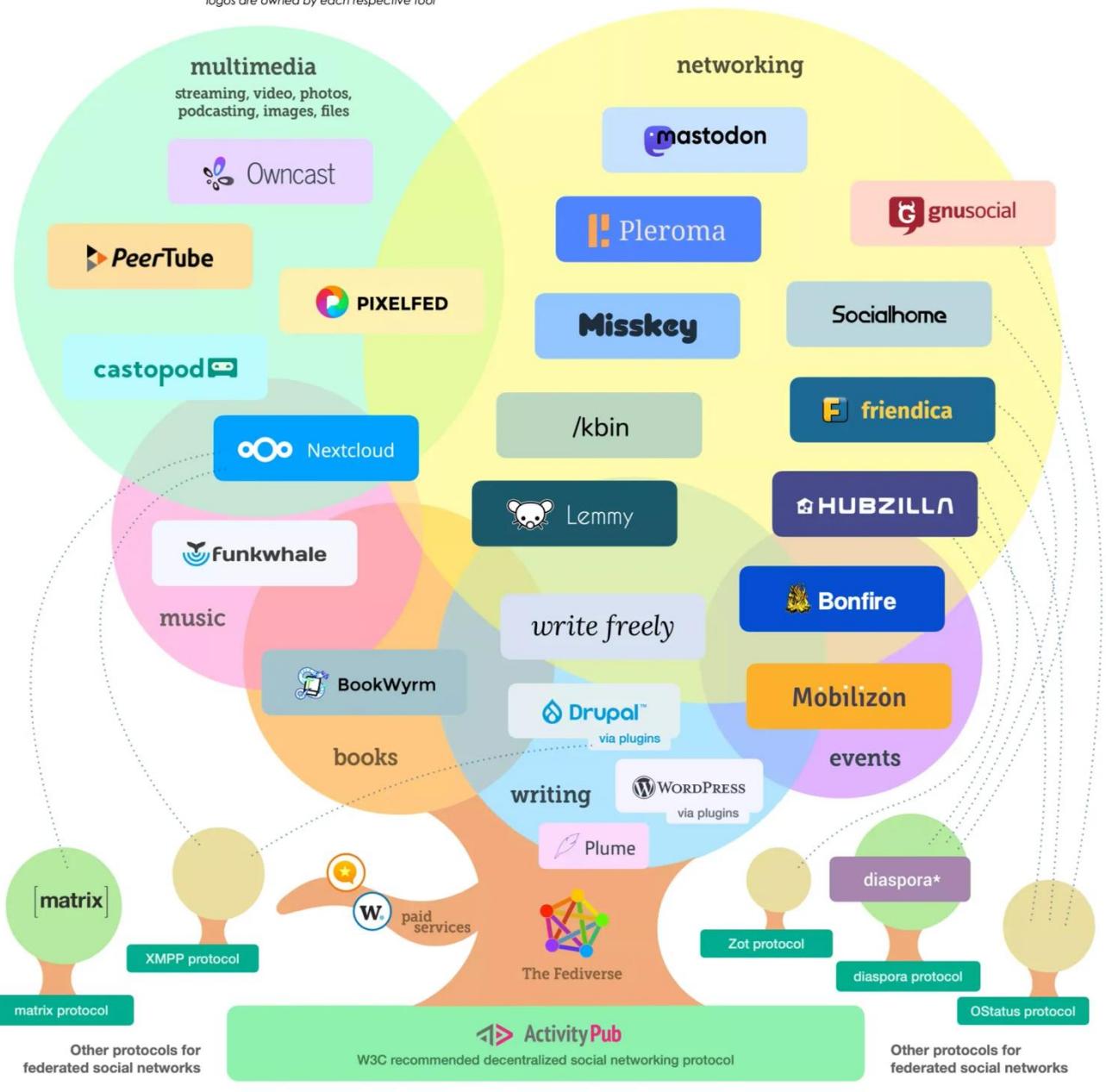
The many branches of the Fediverse

axbom.com/fediverse • CC BY-SA Per Axbom

version 3.0 • January 17, 2023

logos are owned by each respective tool

Note: this is an overview and not a complete mapping of the Fediverse.



Many other service categories exist

- Most based on ActivityPub

Image sharing	Pixelfed
Music sharing	Funkwhale
Podcasting	Castopod
Video hosting	PeerTube
Full blogging	Wordpress, Plume, etc.

make it easier to use & discover
decentralized/federated services

Acknowledgments

COSIM CHINA 2024



Rik Arends



Eddy Bruël



Edward Tan



Julian
Montes de Oca



Jorge Bejar



Sebastian
Michailidis



Klim Tsoutsman



THANK YOU



Interested? Find us here:



@project-robius/robrix



 @kevinaboos



#robius@matrix.org
#robrix@matrix.org