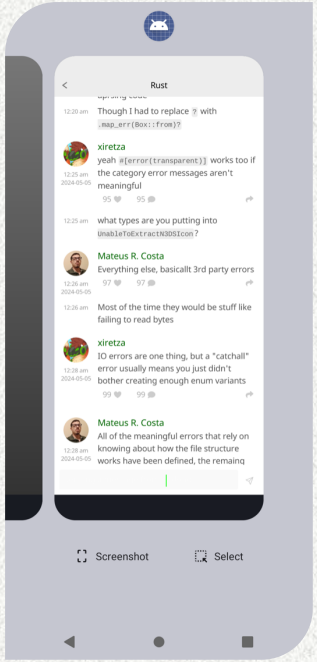
RustNL GOSIM 2024 EUROPE

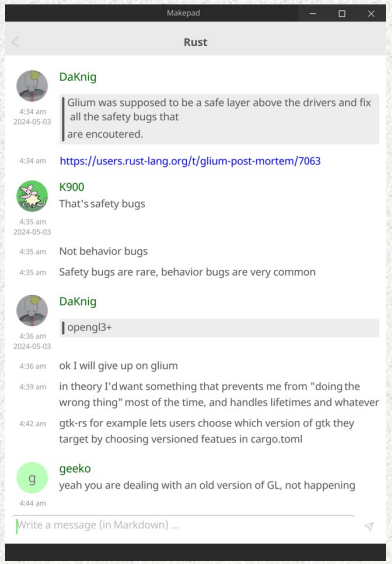
Robrix

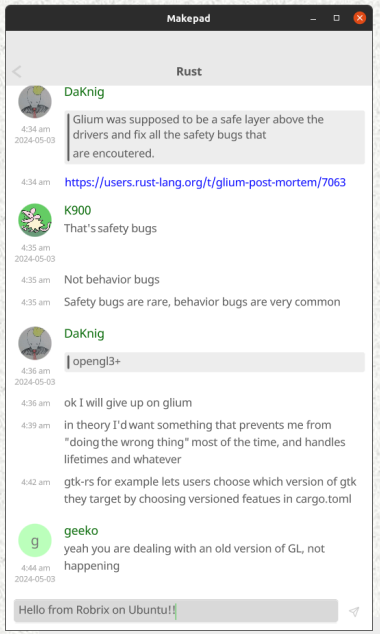
A multi-platform Rust app for Matrix chat and more

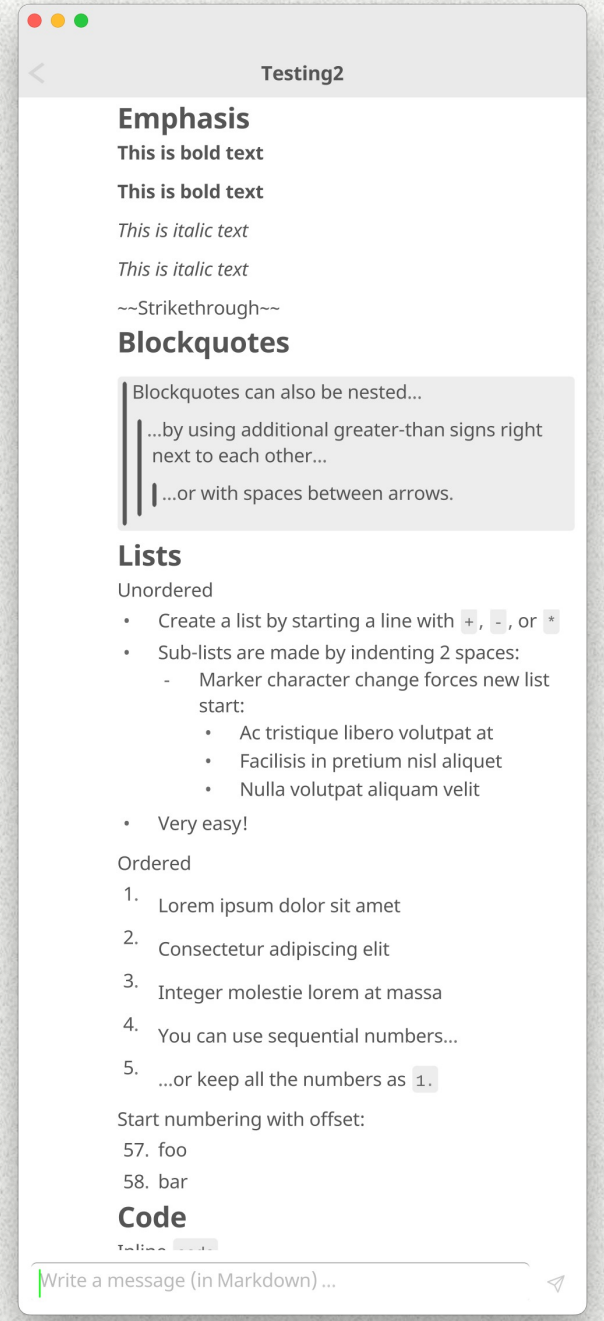
Kevin Boos
Principal Architect, Futurewei

May 6th, 2024



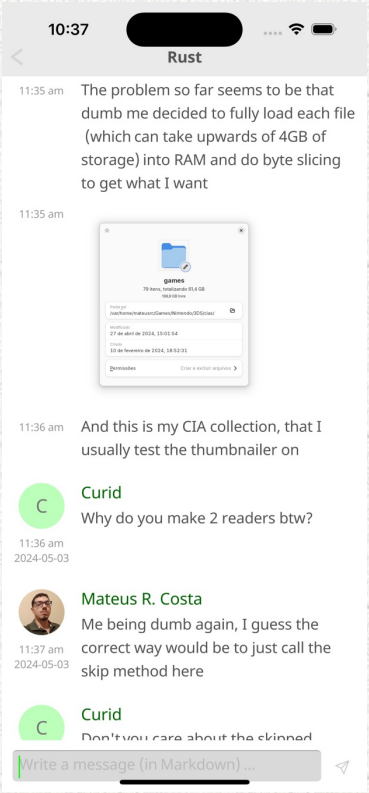


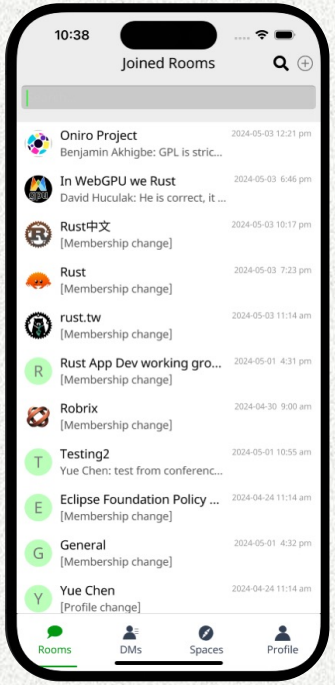


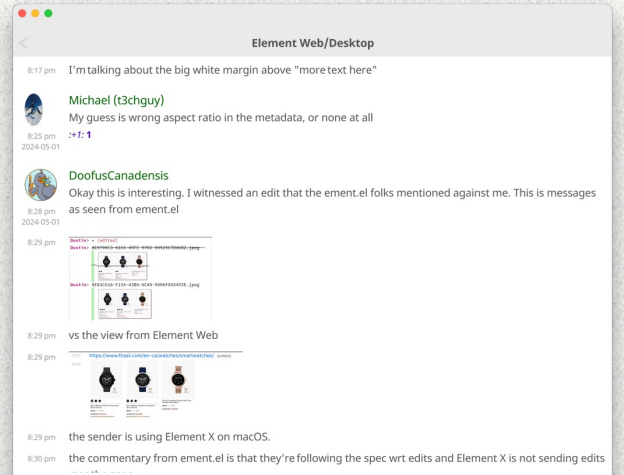


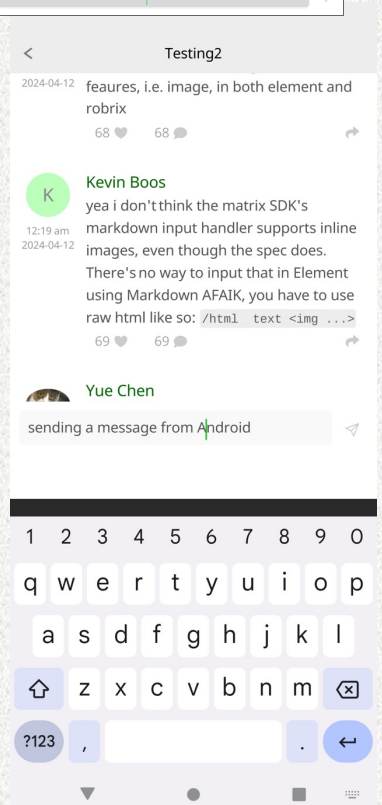
ROBRIX

One code base, many platforms









Motivation

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 feels 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 tons of great Rust UI toolkits out there

- egui, Iced, Slint, Leptos, Tauri, Dioxus, Makepad, Xilem, Freya, Ribir, …

→ There's **way more to app dev** than just drawing a GUI!

- Many existing Rust apps are "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 development of Robius components
   - "What kind of app requires lots of platform functionality?" → Matrix!
     - Network, geo-location, file/image access, audio/video capture & playback, system notifications, clipboard, rich text formatting & input, local storage/caching, connectivity mgmt., biometric authentication, secret storage

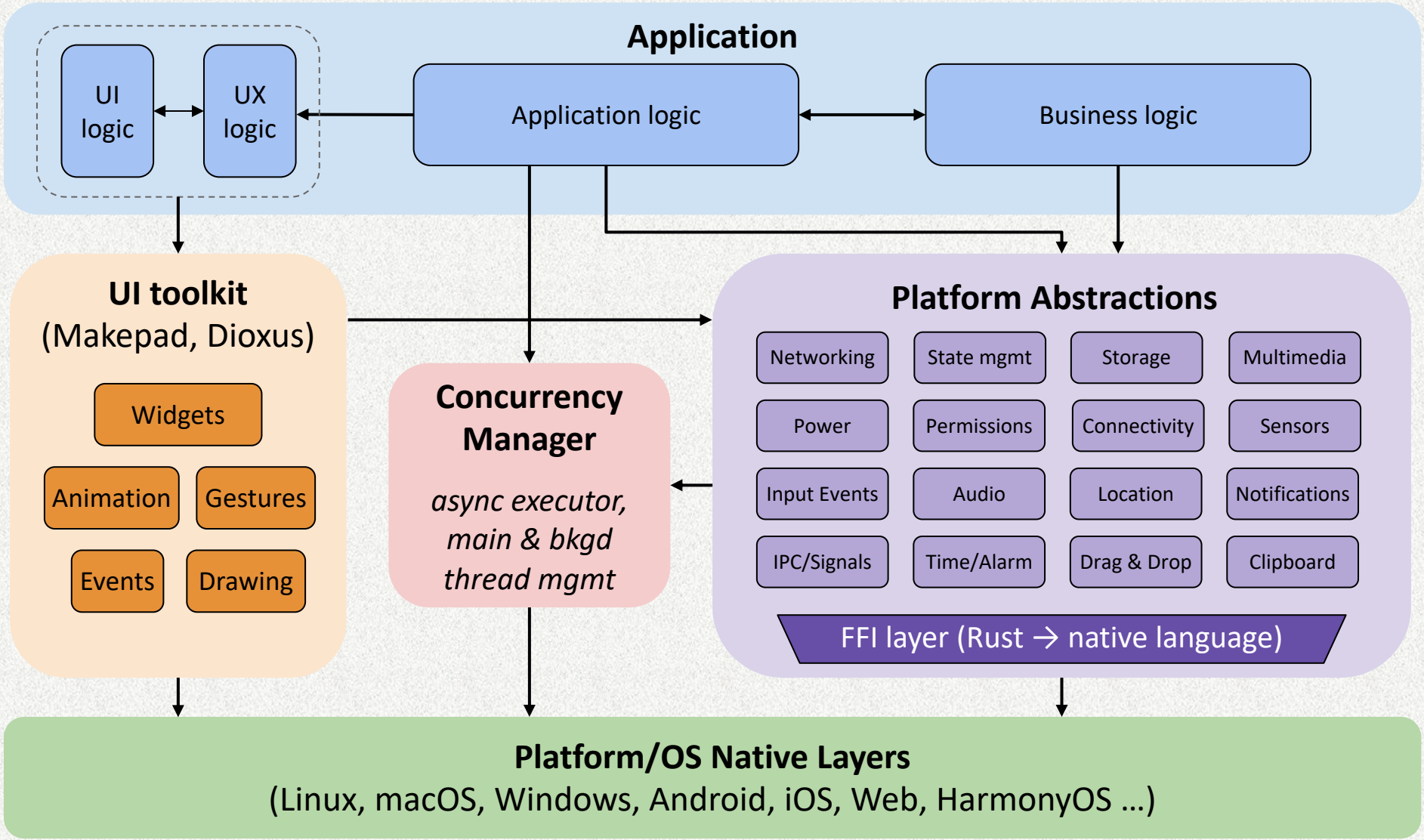
2. After that … why not take it further?
   - Make Robrix a central "hub" for using federated services
     - Focus on the needs of the open-source developer community
     - Matrix, ActivityPub, local AI LLMs, source code (re)view, etc.

for more on Project Robius,

check out my RustNL talk,
Wednesday @ 10am

Goals & Roadmap (selected)

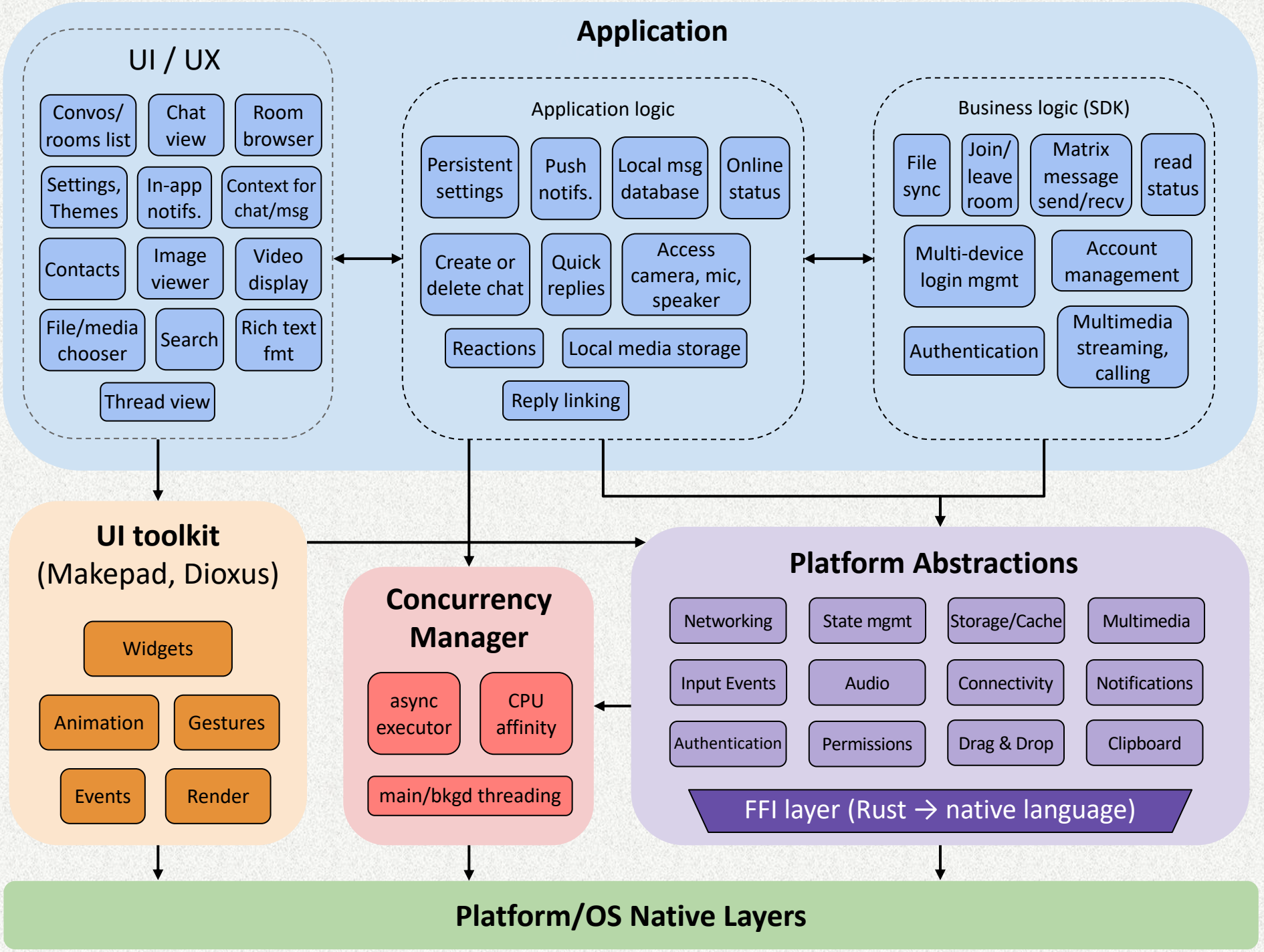
Create reference design of full Robius system stack



Robrix App Architecture

(Matrix client only)

Robrix's needs guide Robius development and implementation priority



Meta-goals for Robrix as a flagship Robius app

- Well-structured, well-commented code

- Easy to fork and modify
  - 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 Open Wallet Foundation

Stage 1: Robrix chat

Stage 1: the basics

- Realize simple Matrix client
  - Login, rooms list (sliding sync), basic messaging, annotations, replies, rich text formatting, images, E2EE support, app persistence
- Publish Robrix to app stores
  - Requires developing 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	✅	✅

Feature status tracker github.com/project-robius/robrix

Basic room views and fundamental actions

- [x] View list of joined rooms
- [x] View timeline of events in a single room
- [x] Fetch and display room avatars
- [x] Fetch user profiles (displayable names)
- [x] Fetch and display user profile avatars
- [x] Backwards pagination (upon viewing a room timeline)
- [ ] Dynamic backwards pagination based on scroll position/movement
- [ ] Loading animation while waiting for pagination request
- [x] Stable positioning of events view during timeline update
- [x] Display simple text-only messages
- [x] Display image messages (PNG, JPEG)
- [x] Rich text formatting for message bodies
- [ ] Display multimedia (audio/video/gif) message events
- [x] Display reactions (annotations)
- [ ] Handle opening links on click
- [ ] Linkify plaintext hyperlinks
- [ ] Inline link previews
- [ ] Inline reply view
- [x] Send messages (standalone, no replies)
- [ ] Interactive reaction button, send reactions
- [ ] Reply button, send reply
- [ ] Error display banners: no connection, failure to login, sync timeout.
- [ ] Collapsible/expandable view of contiguous "small" events
- [ ] Encrypted rooms, decrypting messages

Some basics to save for Stage 2 (later 2024)

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

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 displaying notifications (and so much more)

all across disparate platforms

- Managing concurrency: multi-threading, async/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 support crates must be easy to integrate with UI system internals

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

and so much more …

Status — just getting started, lots to do!

- 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

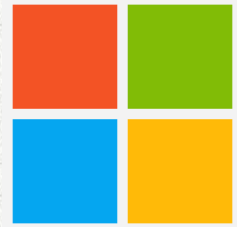
and so much more …

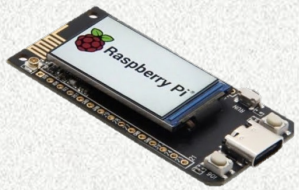
Platforms of interest

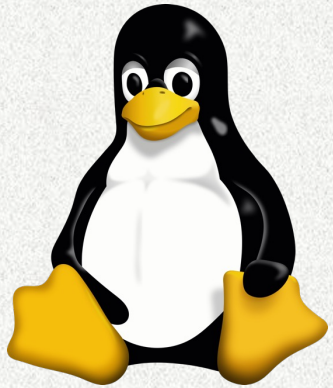
- Desktop
  - macOS
  - Linux (primarily Debian-based)
  - Windows
- Mobile
  - Android
  - iOS
  - [Future] OpenHarmony OS
- Web, wasm
- Others?
  - Select microcontrollers/peripherals
  - tvOS, watchOS

WEBASSEMBLY

OpenHarmony







Demo time!

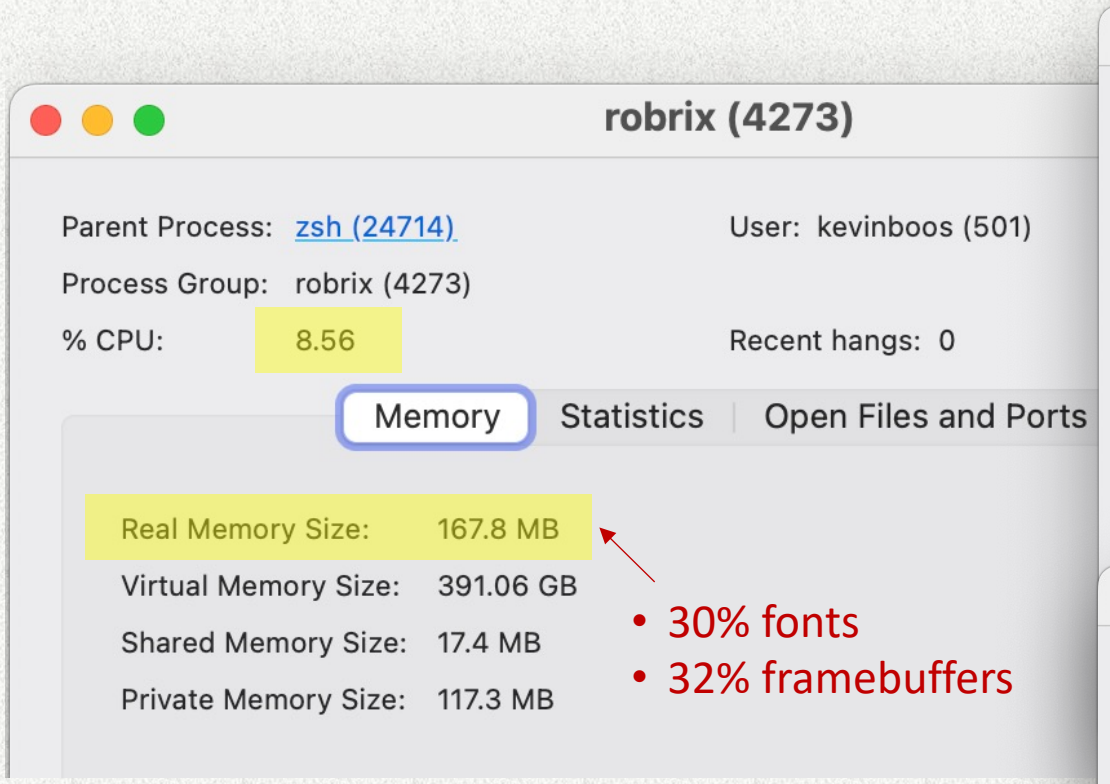
<demo>

for future/offline viewers:

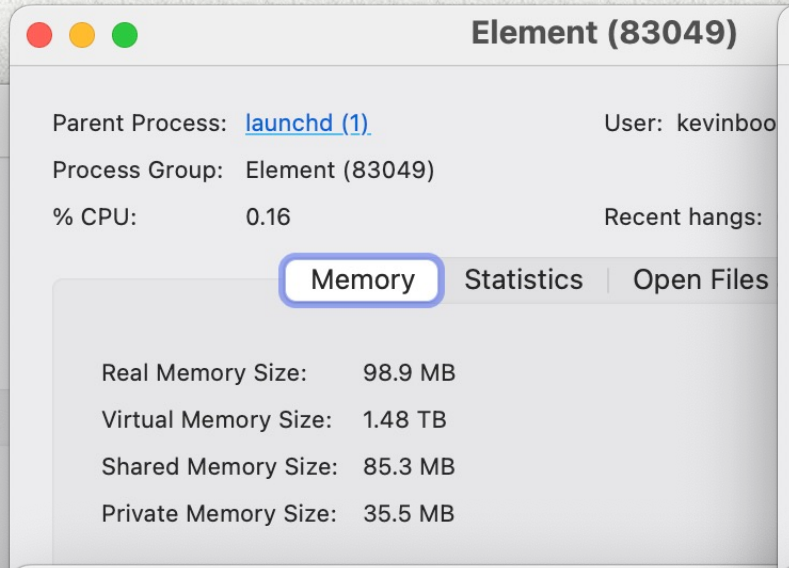
the demo included an overview of Robrix running on macOS and Android.
Check out our git repo and run it on your own machine here:
https://github.com/project-robius/robrix

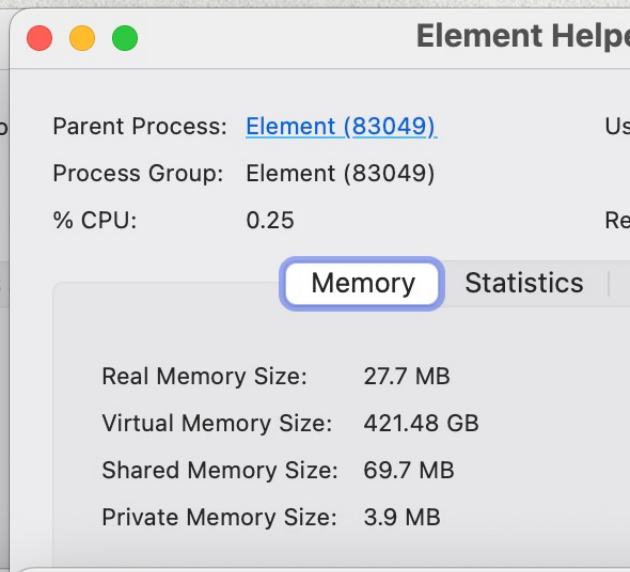
21

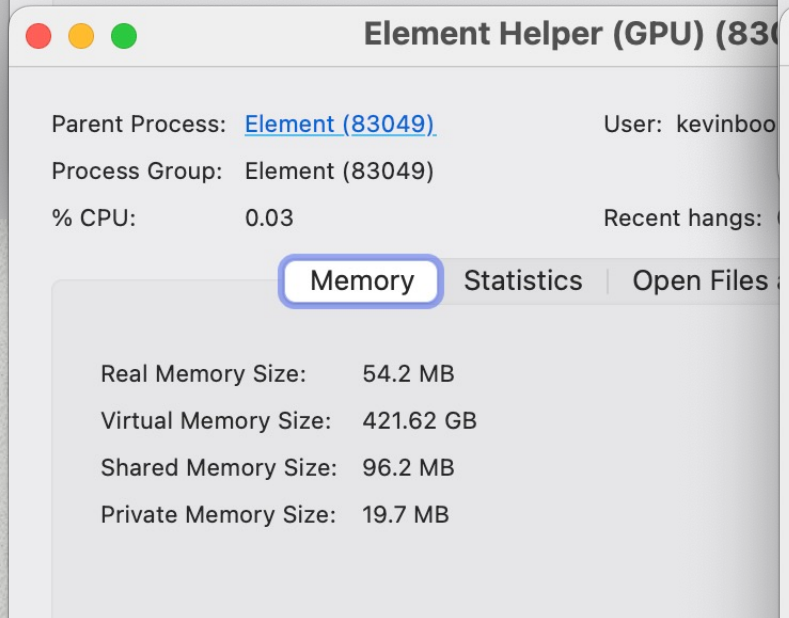
Makepad + Robius stack is efficient

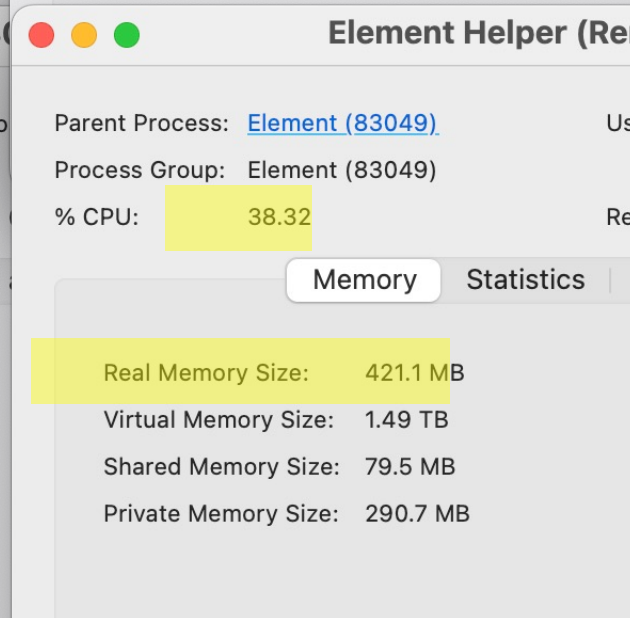


- 30% fonts
- 32% framebuffers

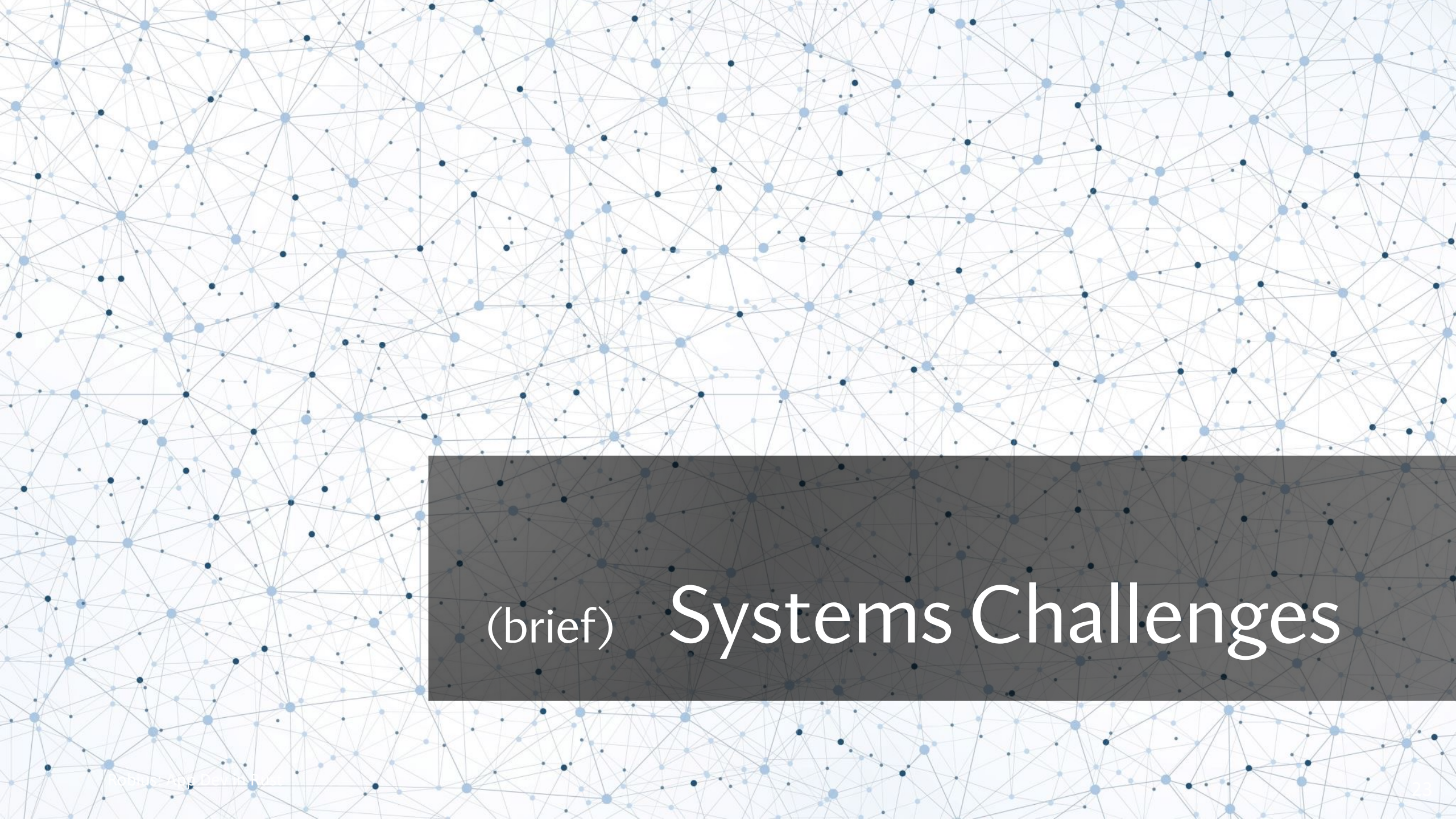








→ Related benchmarks for Makepad:
robius.rs/blog/performance-benchmarking-2/


(brief) Systems Challenges

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 = Handle::current().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)

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

Robrix development revealed a concurrency challenge

```rust
/// 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(),
    },
  );
  ...
}
```

```rust
/// 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

```rust
/// 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);
}
```

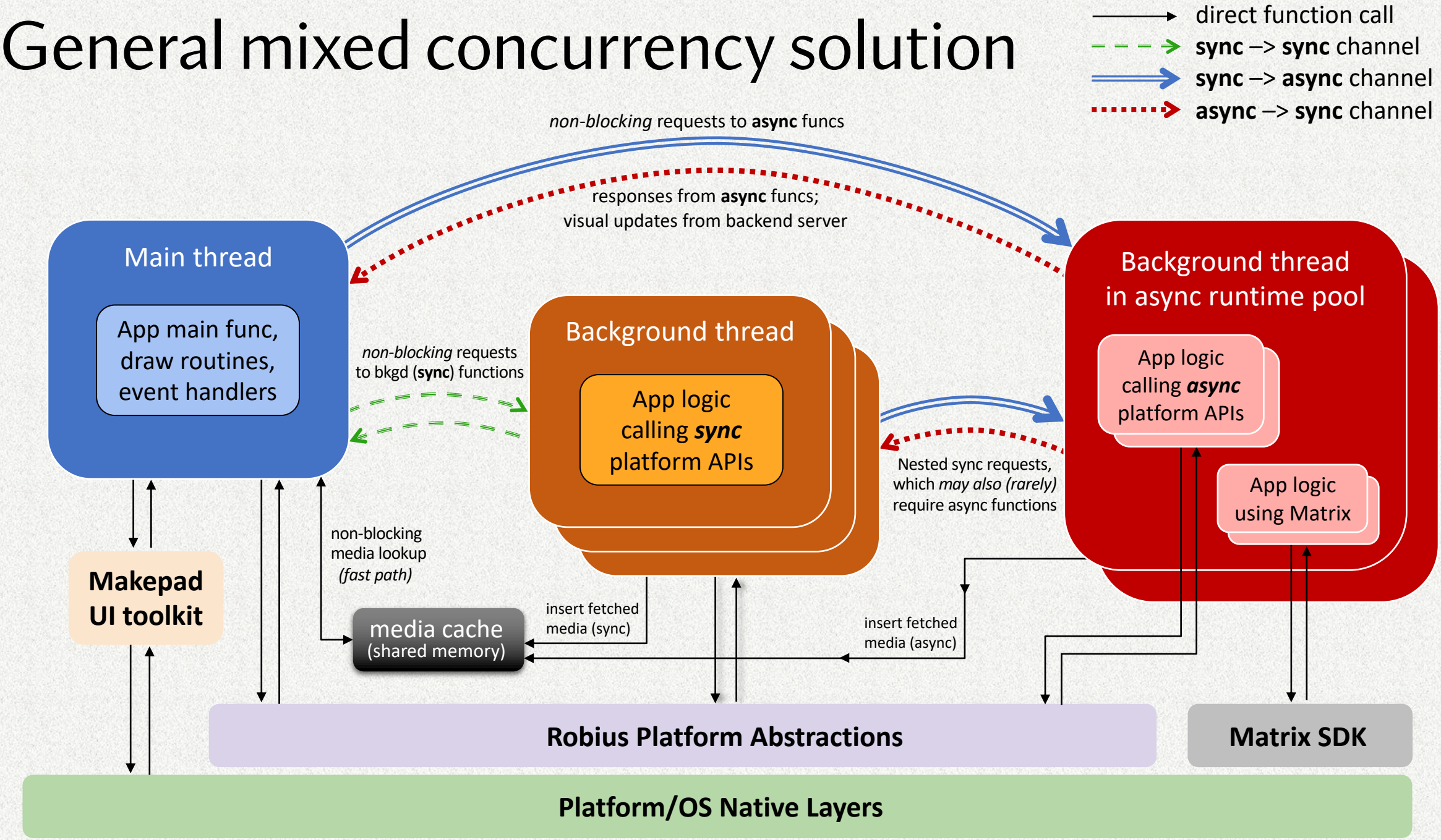
triggers async request handler

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

```rust
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

General mixed concurrency solution



Stage 1 summary, and looking forward

- Robrix (and Robius) are only just getting started!
  - Many challenges remain: UI, platform, build, tooling
  - Active collaboration with UI toolkits, who are improving every day

2024 Roadmap

Continue developing key **feature** & **platform abstraction** crates

Publish first “full” version of Robrix to app stores & package managers

End-to-end **guided walkthroughs**: how to create a simple app like Robrix atop Robius

Future Robrix: Stages 2-3

Stage 2 — a Matrix client for power users

A. Approx. feature parity with existing clients

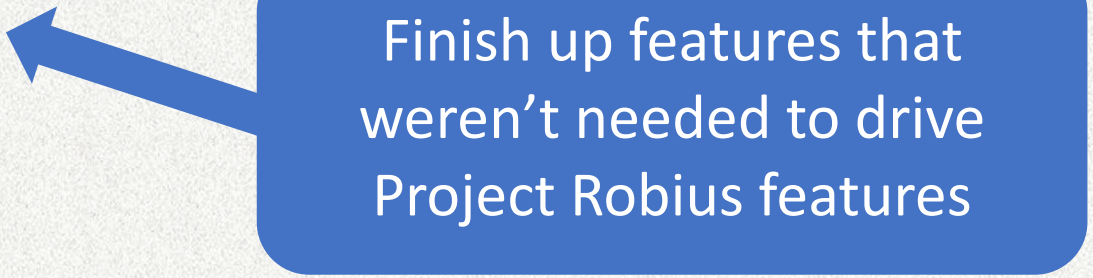
B. Responsive UI with side-by-side panes & dockable tabs
   - Visually beautiful and customizable across desktop, tablet, foldable, mobile
   - Fast navigation with keyboard-driven interaction mode

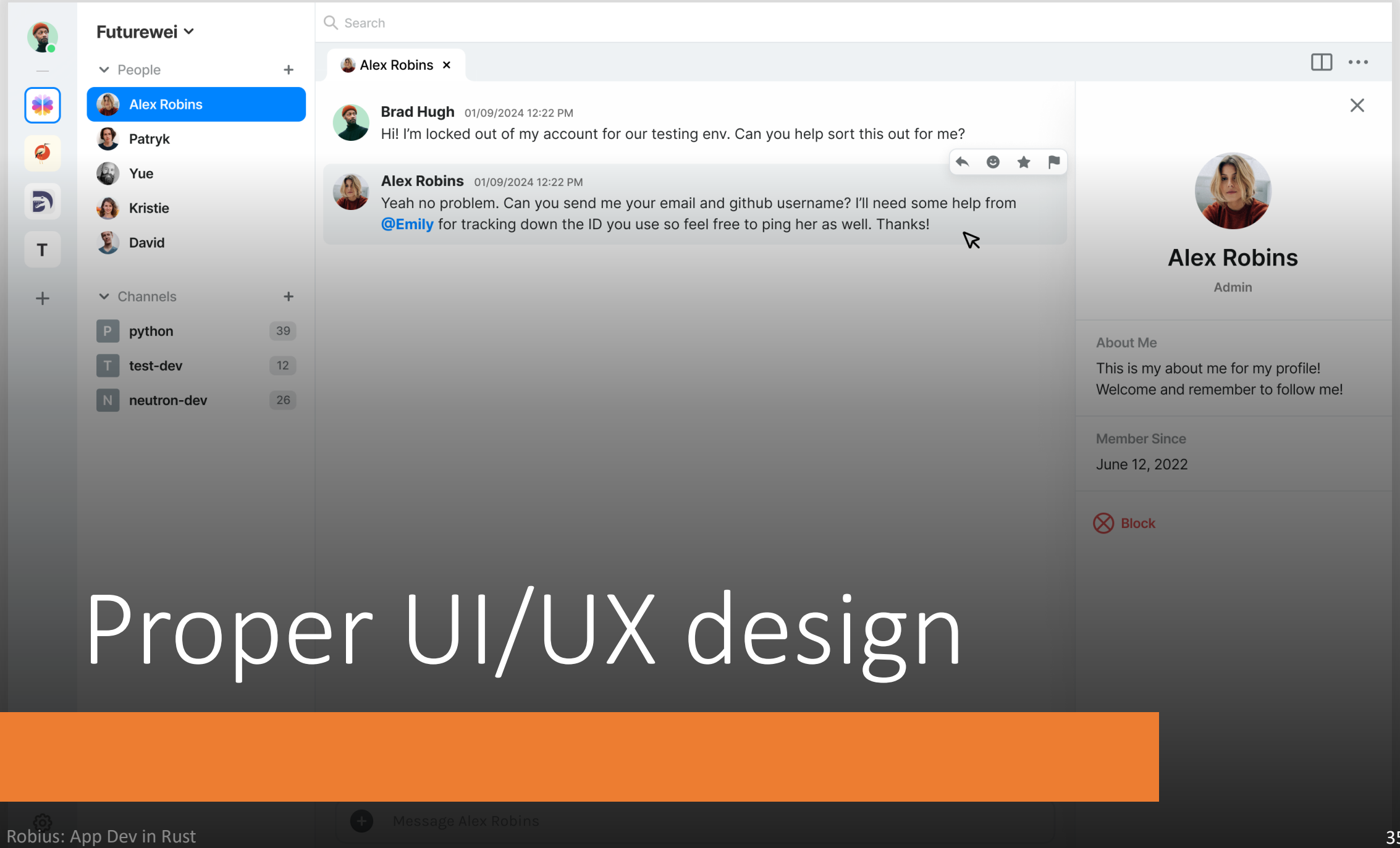
C. Integration with ***local*** LLMs to leverage AI benefits
   - Automated topic analysis, chat synopses of “what you missed”, etc.
   - Chatbot response channels to help newcomers in large open-source communities
   - Local LLM runtime preserves E2EE & data sovereignty

Reach feature parity with existing clients

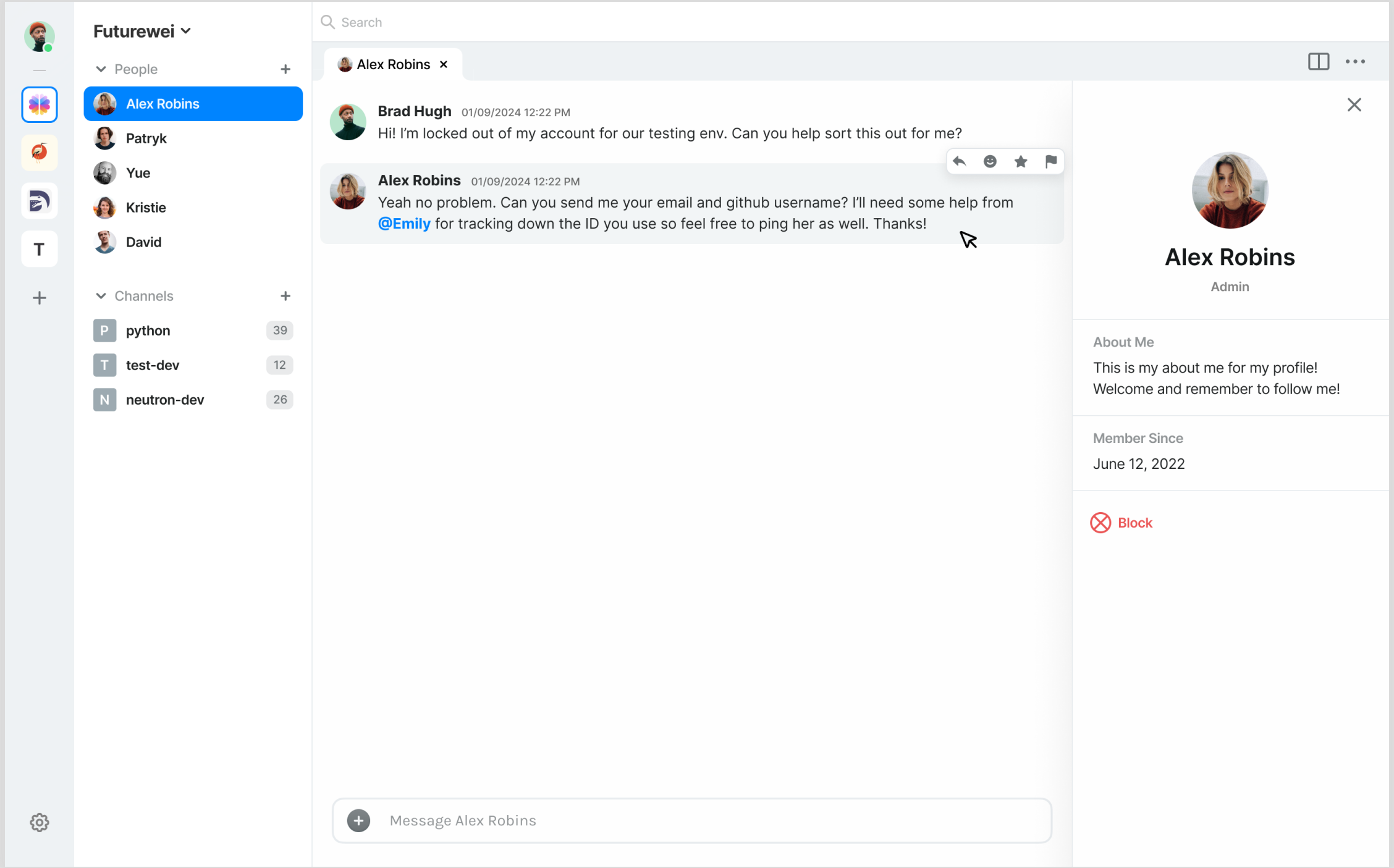
Auxiliary/admin features: login, registration, settings

- [ ] 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





Proper UI/UX design


Futurewei
People
Alex Robins
Patryk
Yue
Kristie
David
Channels
python 39
test-dev 12
neutron-dev 26
Search
Alex Robins
Brad Hugh 01/09/2024 12:22 PM
Hi! I'm locked out of my account for our testing env. Can you help sort this out for me?
Alex Robins 01/09/2024 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!
Message Alex Robins
Alex Robins
Admin
About Me
This is my about me for my profile! Welcome and remember to follow me!
Member Since
June 12, 2022
Block

Multi-pane dockable tabs

Futurewei

Search

People

- Alex Robins
- Patryk
- Yue
- Kristie
- David

Channels

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

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.

Message Alex Robins

test-dev ×

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.

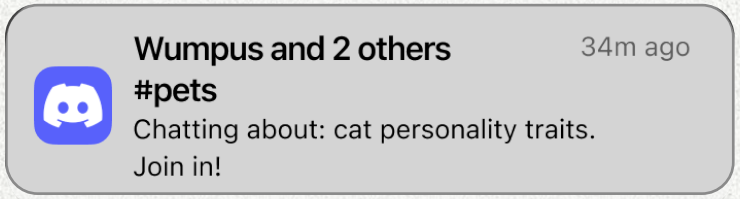
Today

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.

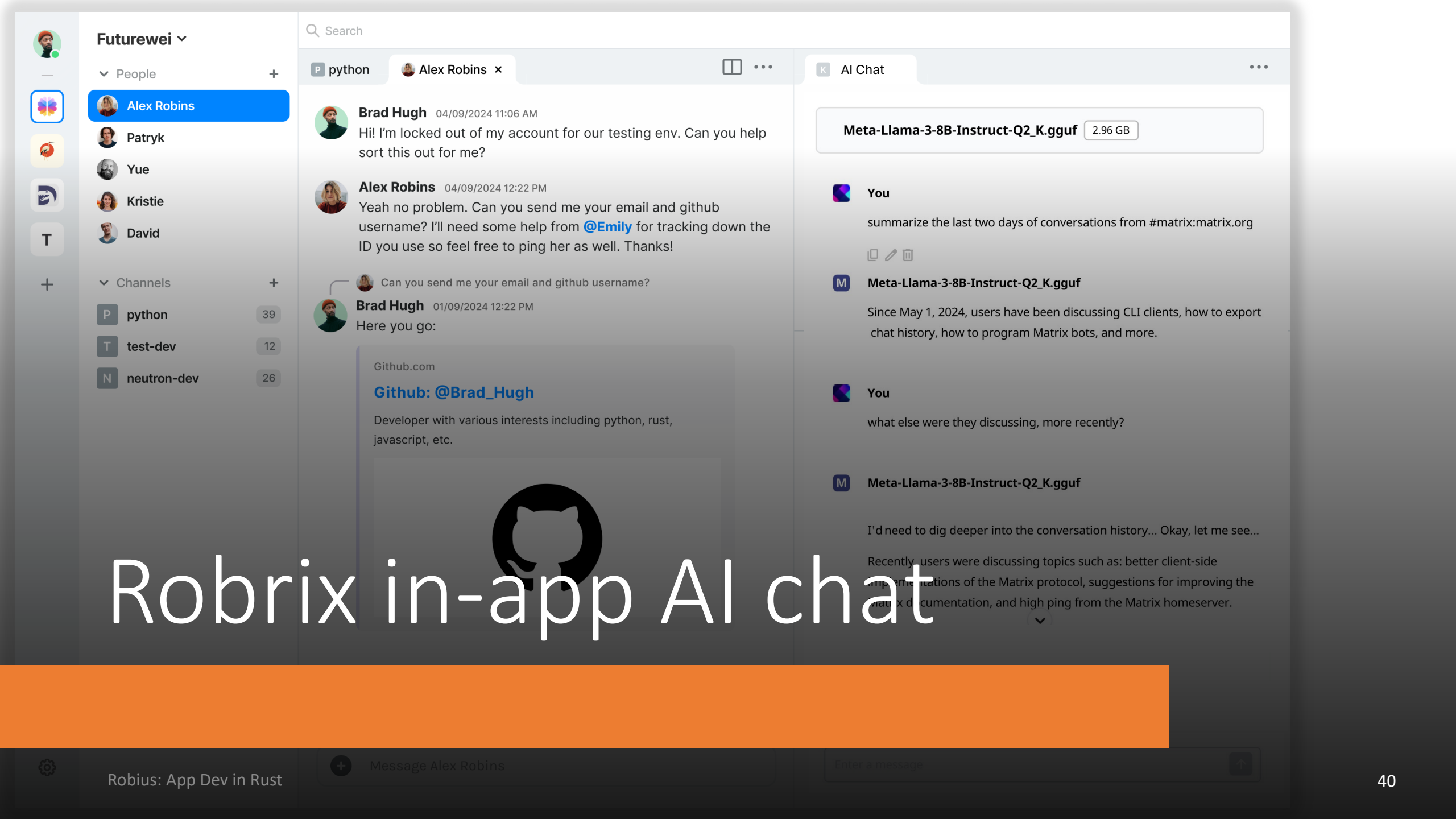
Message test-dev

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

Search

People

- Alex Robins
- Patryk
- Yue
- Kristie
- David

Channels

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

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.

Message Alex Robins

AI Chat

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

AI model is user's choice

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?

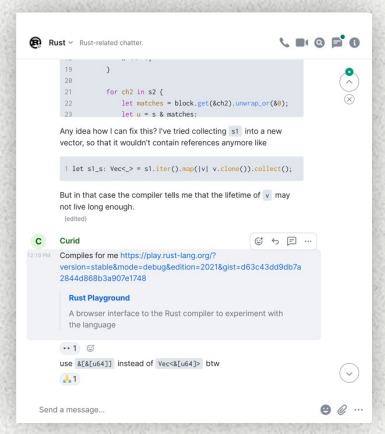
Meta-Llama-3-8B-Instruct-Q2_K.gguf

I'd need to dig deeper into the conversation history... Okay, let me see...

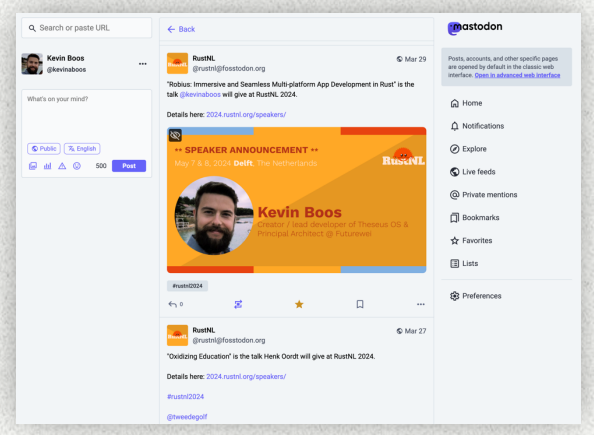
Recently, users were discussing topics such as: better client-side implementations of the Matrix protocol, suggestions for improving the Matrix documentation, and high ping from the Matrix homeserver.

Enter a message

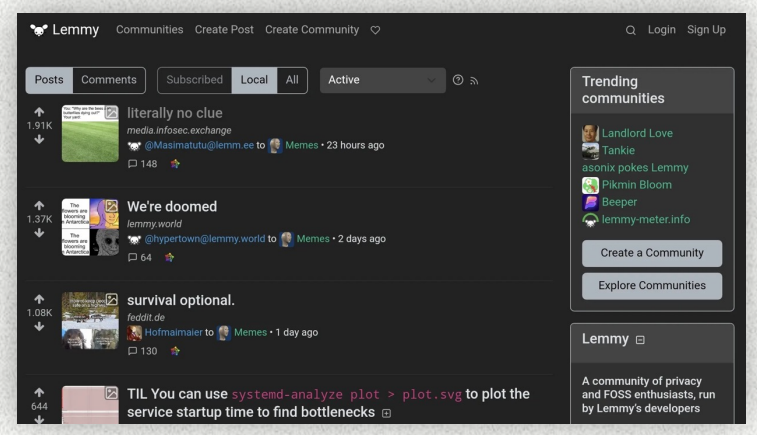
Stage 3 — a community hub (for open-source devs)



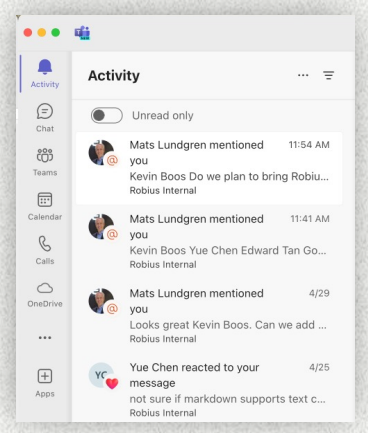
Chat (Matrix)



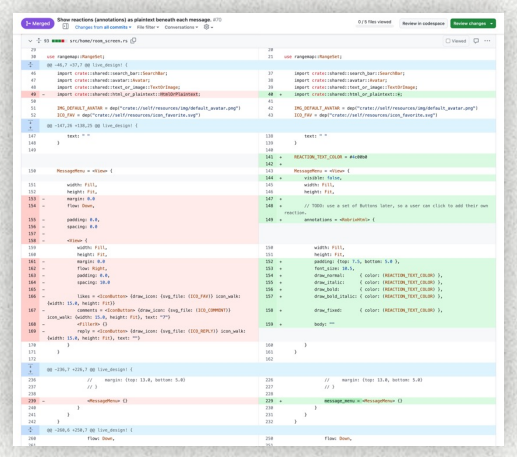
Microblogs (Mastodon)



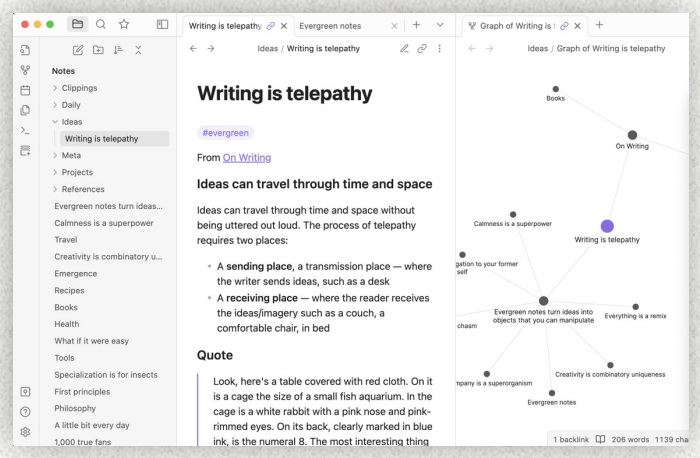
Community Forums (Lemmy)



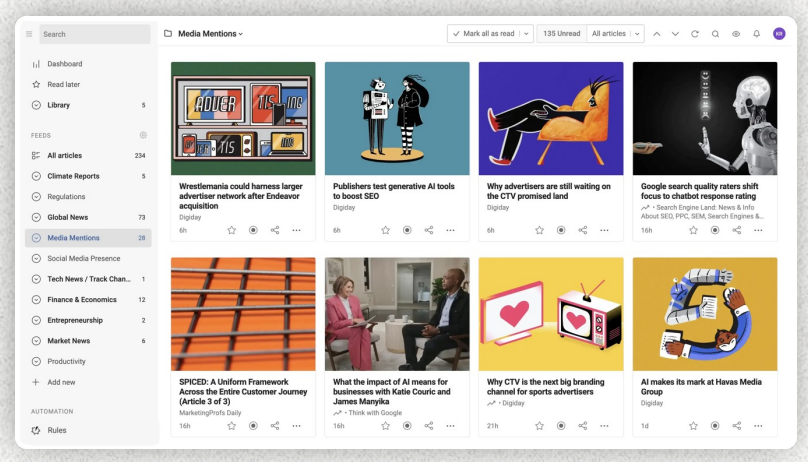
Notifications, activity pane



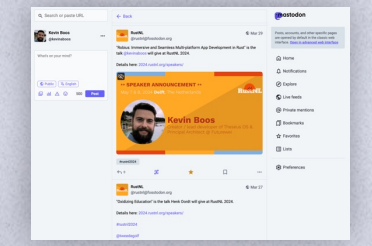
Code [re]view (git)



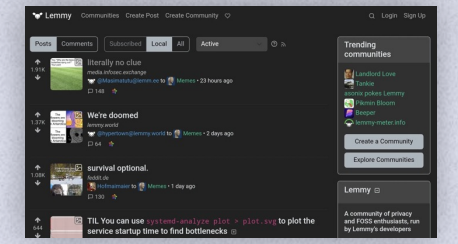
Notes



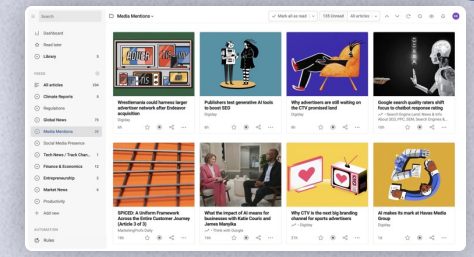
News feeds (RSS)



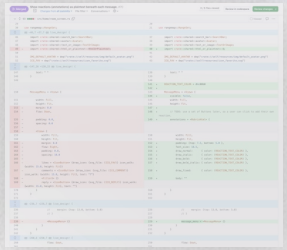
Microblogs
(Mastodon)



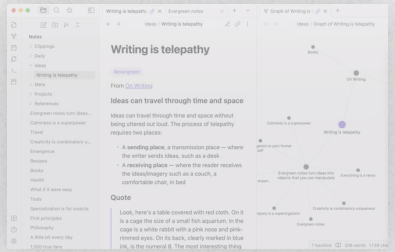
Community Forums
(Lemmy)



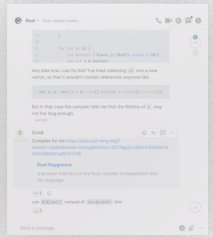
News feeds (RSS)



Code [re]view
(git)



Notes

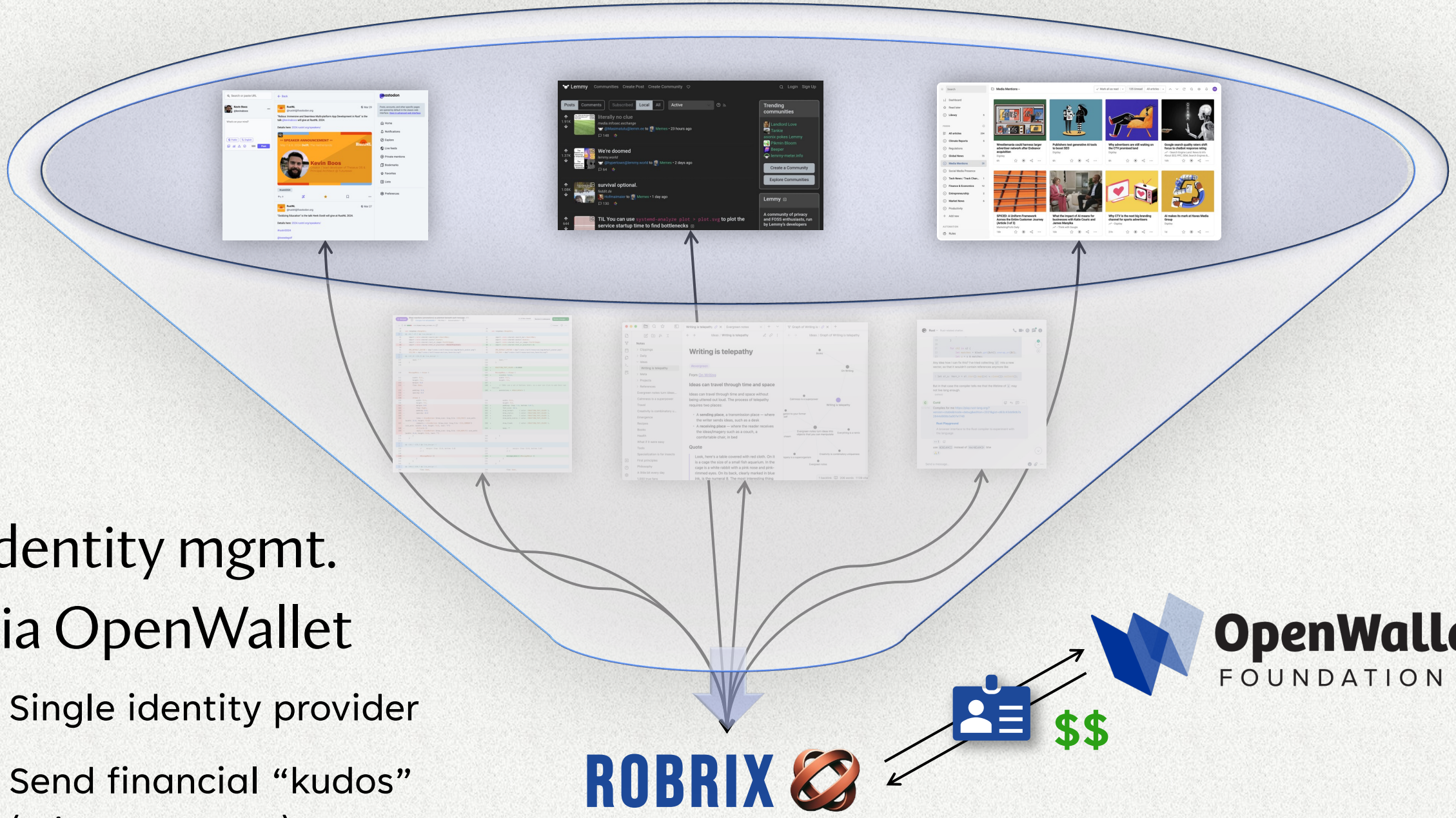


Chat
(Matrix)

Collect multiple federated services in a single experience

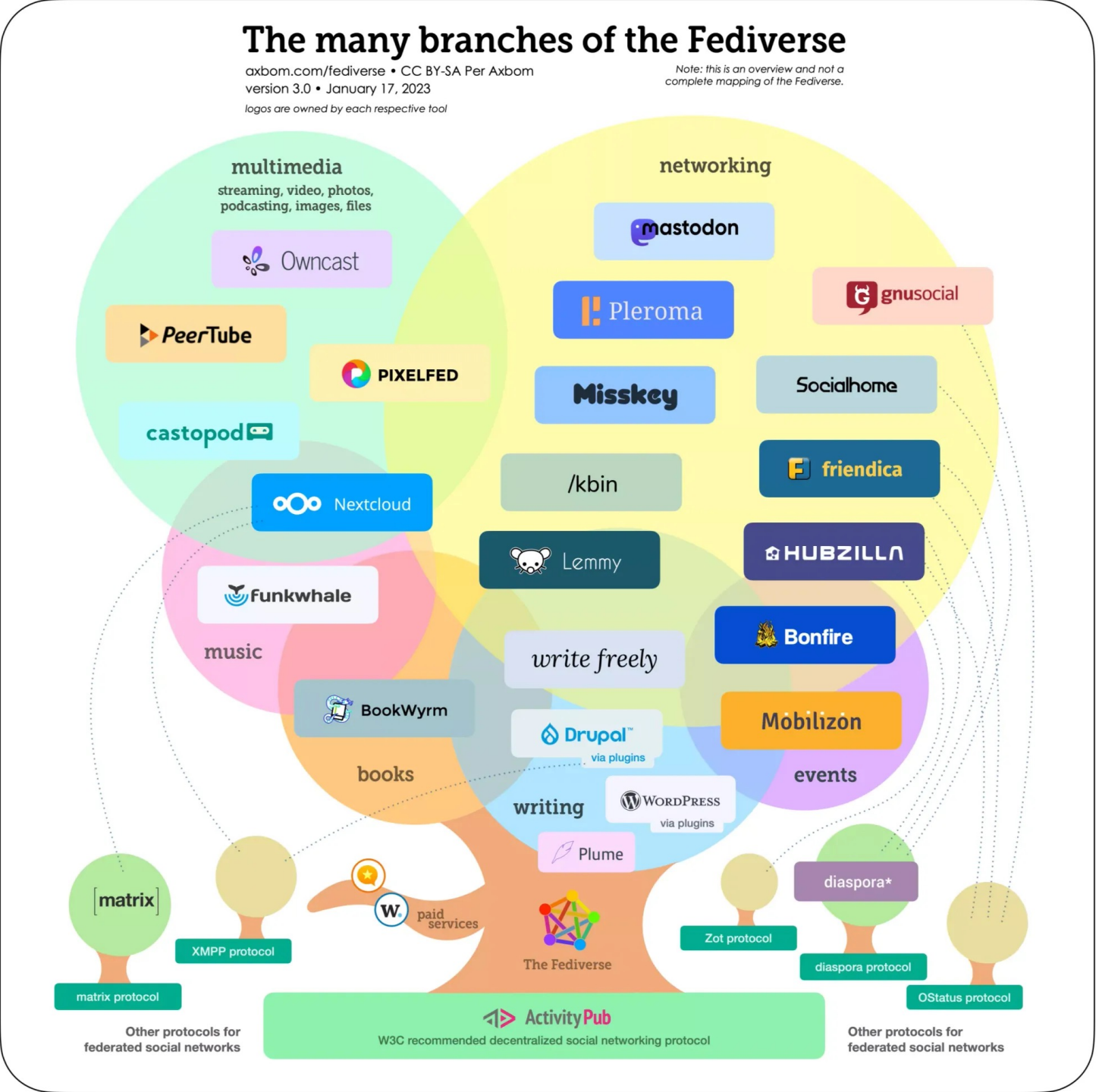
ROBRIX

Potential for powerful combo features & actions



Identity mgmt. via OpenWallet

- Single identity provider
- Send financial “kudos” (micropayments)



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
and discover the fediverse

46

Acknowledgments

Rik Arends

Eddy Bruël

Edward Tan

David Rheinsberg

Klim Tsoutsman

Sebastian Michailidis

Jorge Bejar

Julian Montes de Oca

THANK YOU

Interested? Please reach out:

@project-robius/robrix

@kevinaboos

#robius@matrix.org

GOSIM 2024 EUROPE