State of the Matrix Dart SDK 2024

td
matrix: @td:technodisaster.com
email, fedi: td@technodisaster.com

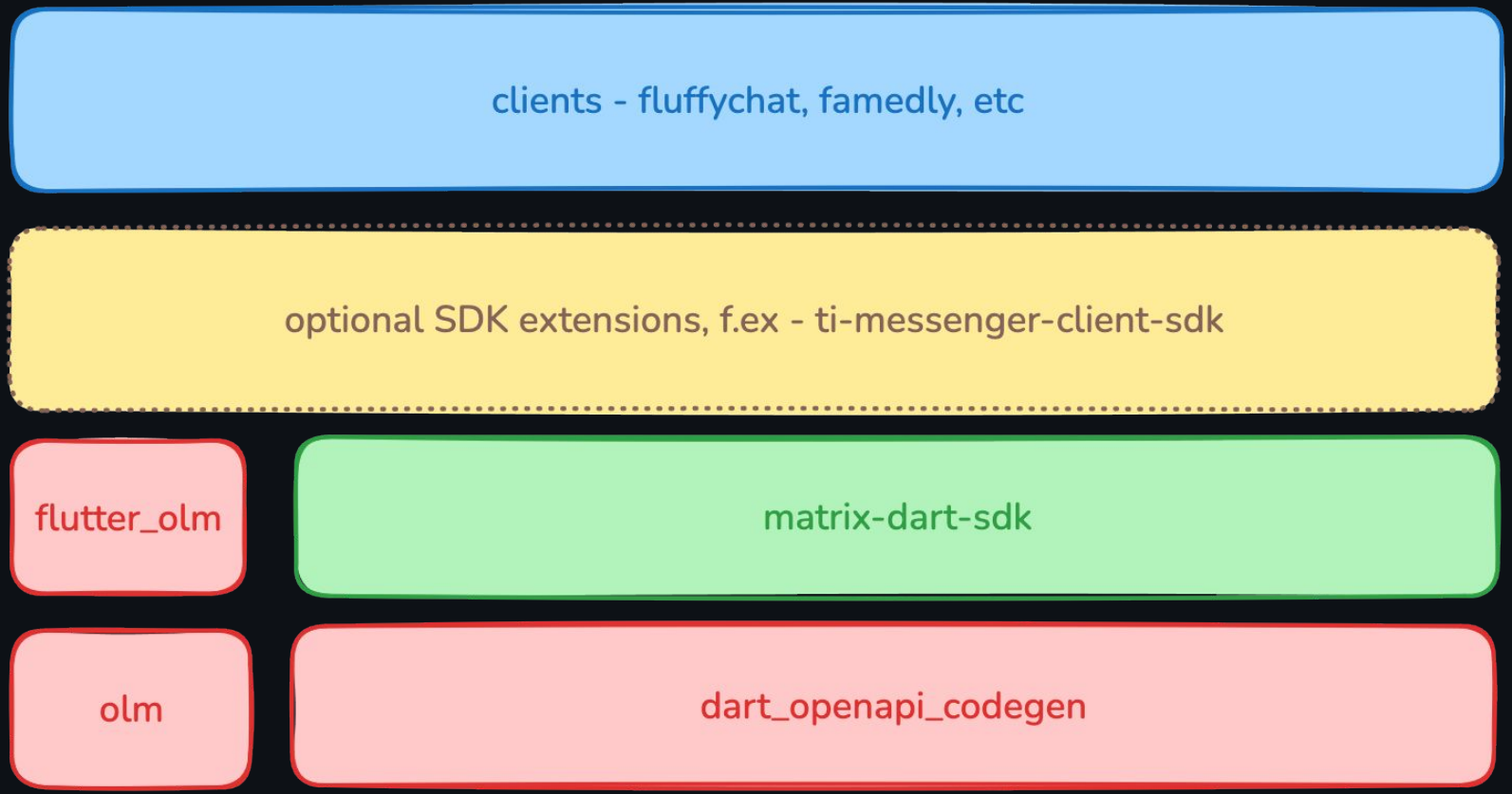
Matrix Dart SDK

- A matrix dart sdk, just released v0.33.0
- Made by the team at Famedly and external contributors
- Open source, AGPL-3.0 license
- Fairly well tested, around 60% coverage and has integration tests

Used by

- Gematik, as their reference SDK.
- Across all the pharmacies in Germany via Gedisa.
- Fluffychat, your cutest client.
- By insurance providers and consumers someday? (*maybe*)
- And ofcourse - Famedly, an approved TI-Messenger.

Structure


clients - fluffychat, famedly, etc
optional SDK extensions, f.ex - ti-messenger-client-sdk
flutter_olm
matrix-dart-sdk
olm
dart_openapi_codegen

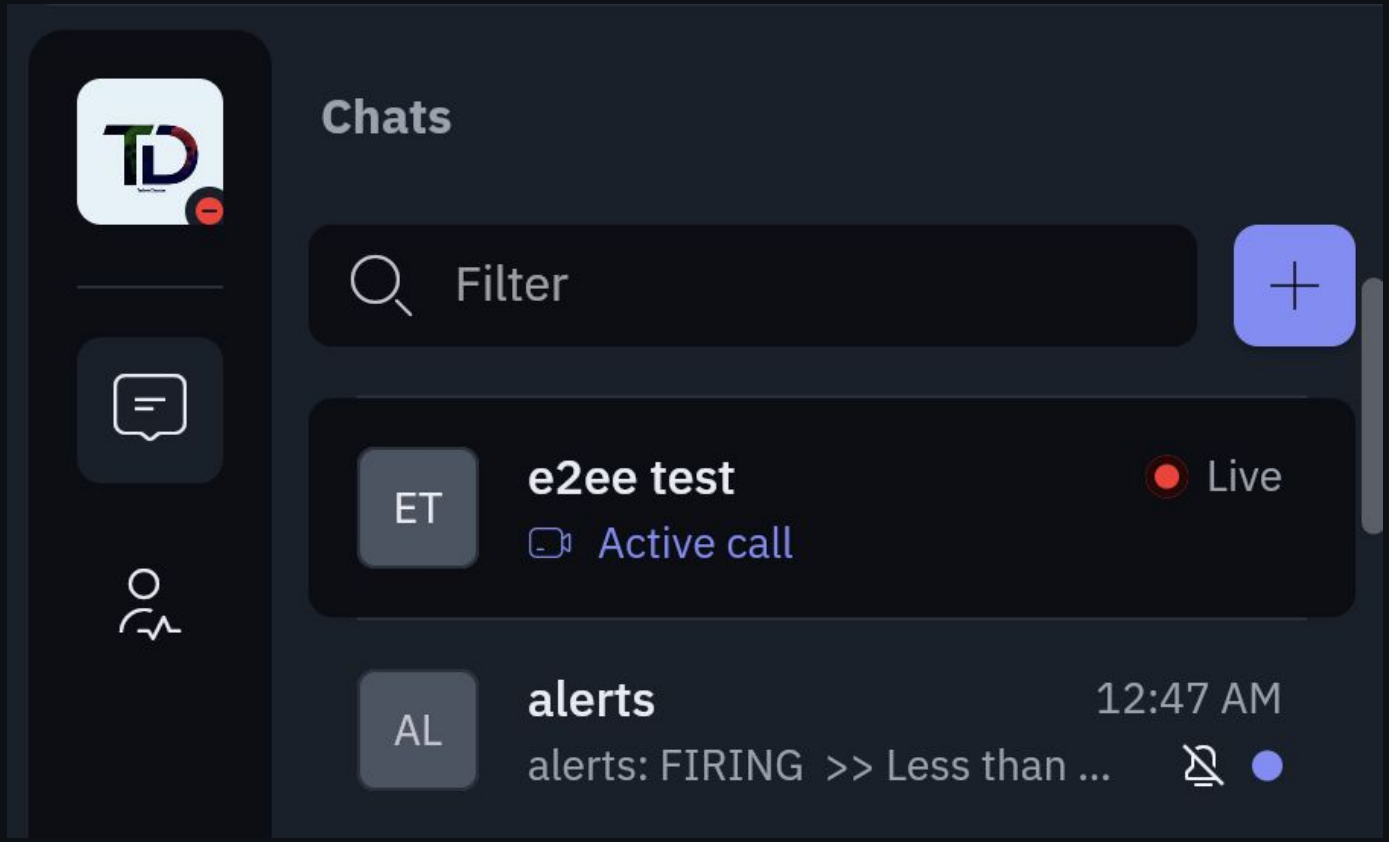
Features!

Database

- By default it uses sqlite for native platforms and indexedDB on web
- Provides you with a thin API if you want to use your own database
- Caching layer to make database queries faster

Chat application focused

- Store the last event in a room separately to make database queries more efficient
- Complete offline mode support
- Timeline has helpers which allow you to granularly update your UI
- Important state events preload for fast room list view



Group call state events when added as an important state event

Crypto

- Currently on olm, minimum version 3.2.7 (fallback key functions)
- Planned to move to vodozemac
    - Already existing private experimental repository

Compute API

- Hook time consuming arithmetic stuff into a different compute zone like a thread or a dart isolate.
  - `generateUploadKeys`
  - `keyFromPassphrase`
  - `decryptFile`
  - `shrinkImage`
  - `calcImageMetadata`
- Auto fallback on main thread

Everything is reactive and observable

- Login state change
```
client.onLoginStateChanged.stream.listen(
    (state) async => await _handleLoginStateChanged,
);
```

- Key verification request
```
client.onKeyVerificationRequest.stream.listen ...
```

- Participants change in a call
```
_groupCall.matrixRTCEventStream.stream.listen((event) async {
    if (event is ParticipantsChangeEvent)
playAudioOnParticipantChange(event);
}
```

Native RTC

- Customizable RTC backends allow you to easily hook your own SFU
- Support native peer to peer webrtc calls
- Supports livekit and mesh backends for group calls
- (even has experimental cloudflare SFU support!)

... to our sponsor!

It actually makes stuff easier for you!

UIA

Easy to handle UIA requests.

- `client.onUiaRequest.stream.listen ...`

```
future: () => client.uiaRequestBackground(
  (auth) => client.deleteDevices(
    deviceIds,
    auth: auth,
  ),
```

```
let auth = client.matrix_auth();
let mut try_login = true;
if let Err(resp) = auth.register(RegistrationRequest::new()).await {
    // FIXME: do actually check the registration types...
    if let Some(_response) = resp.as_uiaa_response() {
        let request = assign!(RegistrationRequest::new(), {
            username: Some(self.username.clone()),
            password: Some(self.username.clone()),

            auth: Some(uiaa::AuthData::Dummy(uiaa::Dummy::new())),
        });
        // if this failed, we will attempt to login after.
        try_login = auth.register(request).await.is_err();
    }
}
```

SSSS

BootstrapState makes it clear what's going on in the ssss key process.

```
-   client.encryption!.bootstrap(onUpdate ...
```

```
switch (bootstrap.state) {
  case BootstrapState.loading:
    break;
  case BootstrapState.askWipeSsss:
```

```
enum BootstrapState {
  /// Existing SSSS found, should we wipe it?
  askWipeSsss,
  /// Ask if an existing SSSS should be userDeviceKeys
  askUseExistingSsss,
  /// Ask to unlock all the SSSS keys
  askUnlockSsss,
  /// SSSS is in a bad state, continue with potential dataloss?
  askBadSsss,
  /// Ask for new SSSS key / passphrase
  askNewSsss,
  /// Open an existing SSSS key
  openExistingSsss,
  /// Ask if cross signing should be wiped
  askWipeCrossSigning,
  /// Ask if cross signing should be set up
  askSetupCrossSigning,
  /// Ask if online key backup should be wiped
  askWipeOnlineKeyBackup,
  /// Ask if the online key backup should be set up
  askSetupOnlineKeyBackup,
}
```

Key verifications

Easy to handle key verifications with KeyVerificationState

```
- client.onKeyVerificationRequest.stream.listen ...
- client.userDeviceKeys[client.userID].startVerification();
```

```
switch (request.state) {
  case KeyVerificationState.askAccept:
    // your UI
    request.acceptVerification()
```

```
enum KeyVerificationState {
  askChoice,
  askAccept,
  askSSSS,
  waitingAccept,
  askSas,
  showQRSuccess, // scanner a
  confirmQRScan, // shower at
  waitingSas,
  done,
  error
}
```

Easily pluggable call backends

```
final List<CallBackend> backupBackends = [];
if (ChatConfigs.livekitGroupCallsEnabled) {
  backupBackends.add(
    LiveKitBackend(
      livekitServiceUrl: ChatConfigs.features!.livekitJwtServiceUrl!,
      livekitAlias: widget.roomId,
    ),
  );
}
backupBackends.add(MeshBackend());
```

Next steps:

- Sliding sync support, maybe even client side
- Migration to vodozemac
- New MAS flow
- Better codegen from the openapi matrix spec
- Database tweaks
- Complement-Crypto?

Thank you!
any questions?

td
matrix: @td:technodisaster.com
email, fedi: td@technodisaster.com