

<https://github.com/ssbc/tinySSB>

- Tremola unterstützt im Moment nur Android (BLE, Wifi und WebSocket) und ein paar embedded devices
- zusätzlich nun eine Version für Chrome-Browser, mit simulierten “untere Schichten”
- *aktuell eine laufende BSc-Arbeit: simulierte “untere Schichten” in JavaScript*  
*Ziel: Einsatz der Library der Firma SocketSupply*  
*—> lauffähige Versionen für MacOS, iOS, Windows sowie Android*

# Die Tremola SW-Architektur

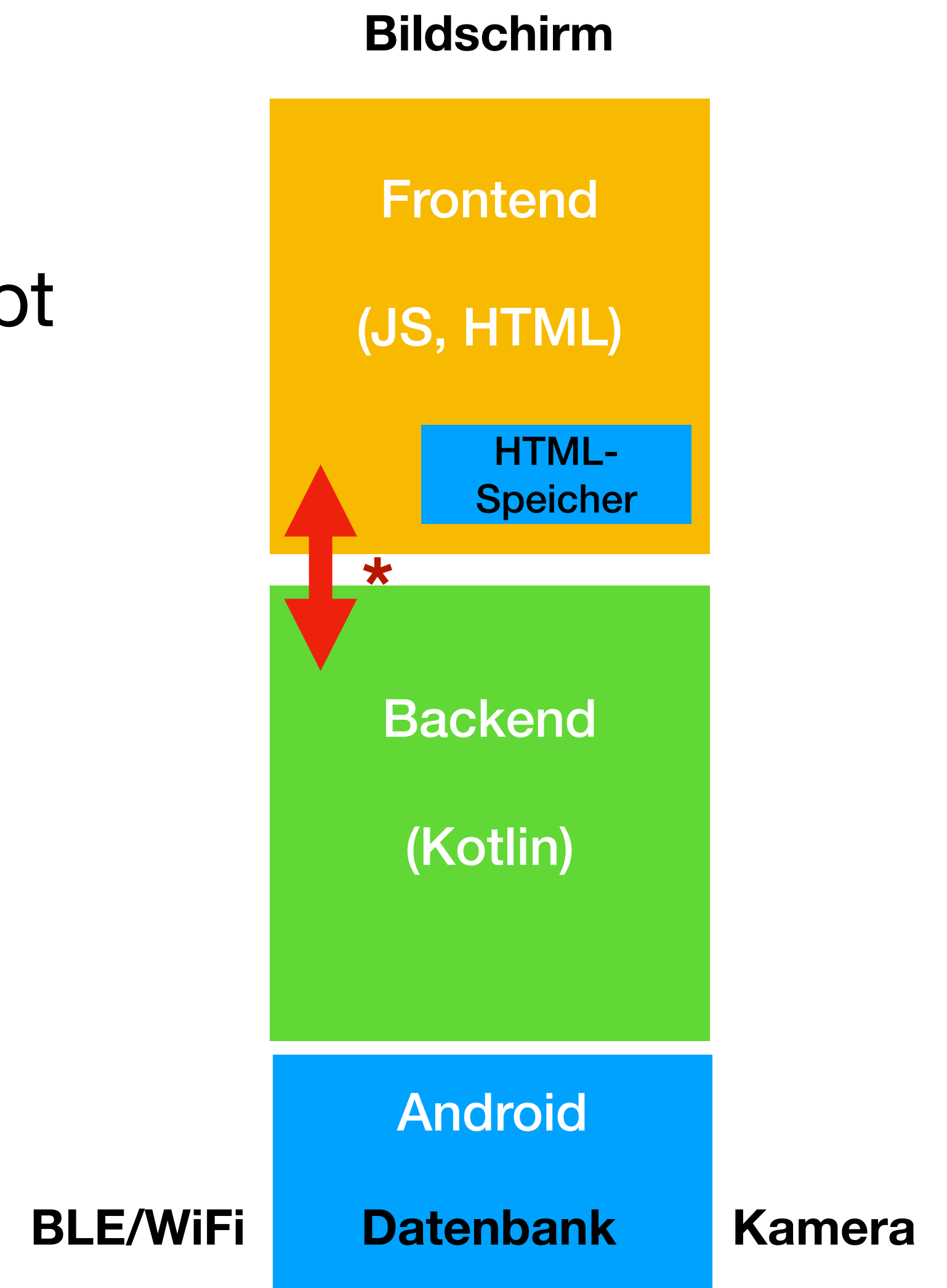
Aufteilung in zwei Komponenten:

## Frontend:

- GUI und Applikationslogik, in HTML und JavaScript
- bekommt und schreibt Log-Einträge als Strings (\*)

## Backend:

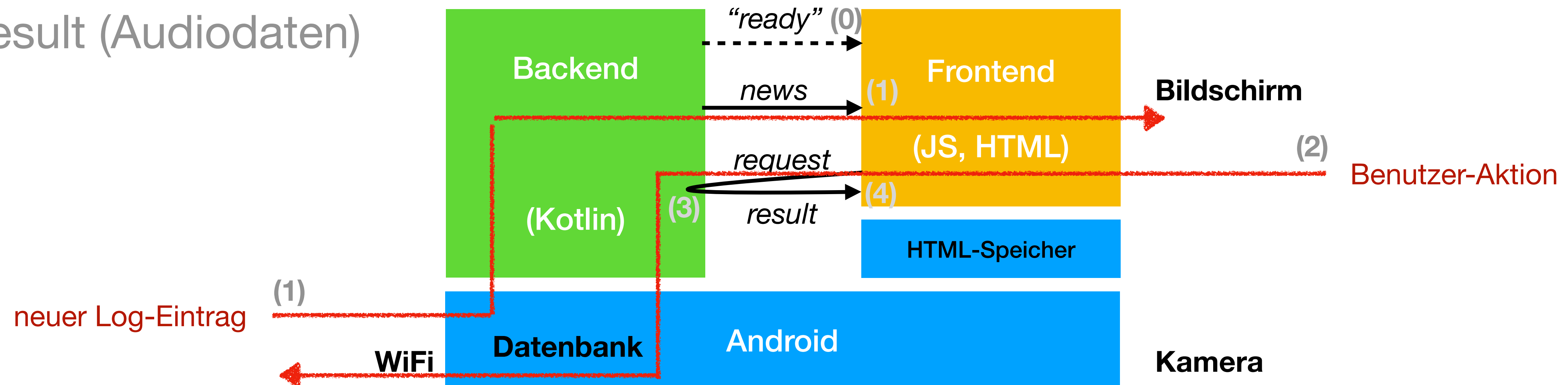
- binäre Kodierung
- digitale Signatur
- Log-Speicherung
- Replikation



# Die Tremola SW-Architektur (Forts.)

*Ereignisgesteuerte Software. Ausführungspfade:*

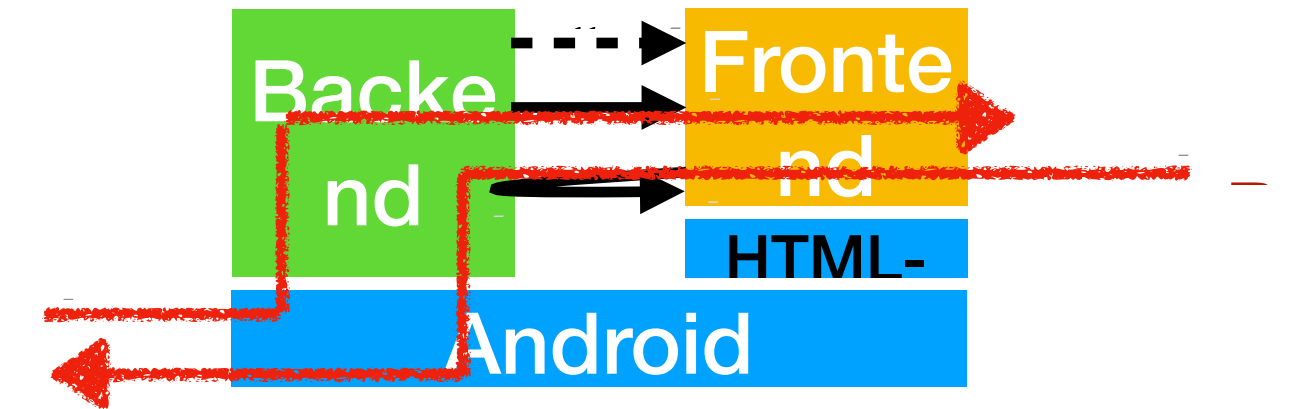
- (0) intern ("ready"-Signal bei Start)
- (1) von aussen (empfangene neue Log-Einträge)
- (2) Benutzer\*innen-Aktion
- (3) request (Start Audiorecording)
- (4) result (Audiodaten)



# Abarbeitung von User-Input

*Zum Beispiel "Senden eine Chat-Meldung"*

a) Klick auf Send-Button



b) Request von Frontend an Backend, in JavaScript:

```
backend("publ:post [] btoa(text) btoa(voice)")
```

c) Backend formatiert dies wie folgt:

```
['TAV', [], "text", bytearray]
```

kodiert dies binär mit BIPF, fügt kryptograph. Signatur dazu,  
macht daraus mehrere 120B Datenpakete, sorgt für die Replikation

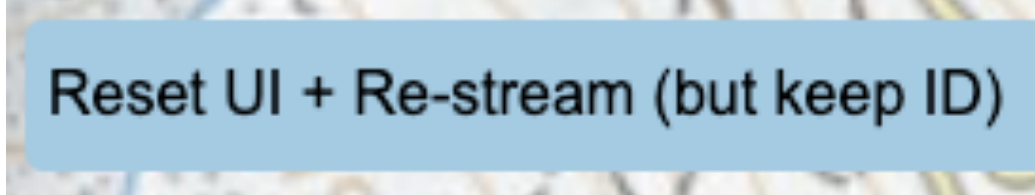
d) Backend schickt zudem diesen Log-Eintrag mittels einem Event e mittels

```
b2f_new_event(e)
```

zum Frontend: erst jetzt wird dies im Chat dargestellt (!)

# Das “Tremola”-Objekt im Frontend

Jede App verarbeitet erhaltene Log-Einträge

- legt wichtige Daten im “Tremola”-Dictionary ab (als JS-Objekte)
- periodisch wird dieses Objekt im HTML5-Speicher persistiert
- beim Neustart von Tremola lädt JavaScript aus dem HTML5-Speicher den letzten Stand
  - Darstellung Chat-Verlauf etc (ohne die Logs nochmals lesen zu müssen)
  - wartet dann auf neue Ereignisse von aussen
- Entwicklungshilfe: “Settings ->  , um das Tremola-Objekt nochmals aufzubauen

# Simulation des Backends für Chrome

*Werkzeug zur Entwicklung der Applikationslogik von Tremola-Apps*

Ansatz: “virtuelles Back-End” (siehe die nächsten Slides)

- braucht nur einen Chrome-Browser, kein Kotlin oder Android
- beinhaltet die “Kanban-Board”-Erweiterung von Hr Heisch (BSc-Arbeit)  
als Beispiel, wie eine zusätzliche Applikation eingebaut werden kann

Die Applikationslogik in JavaScript und HTML kann dadurch schnell und ohne Kompilieren getestet werden, sollte später 1:1 in der Android-Version laufen

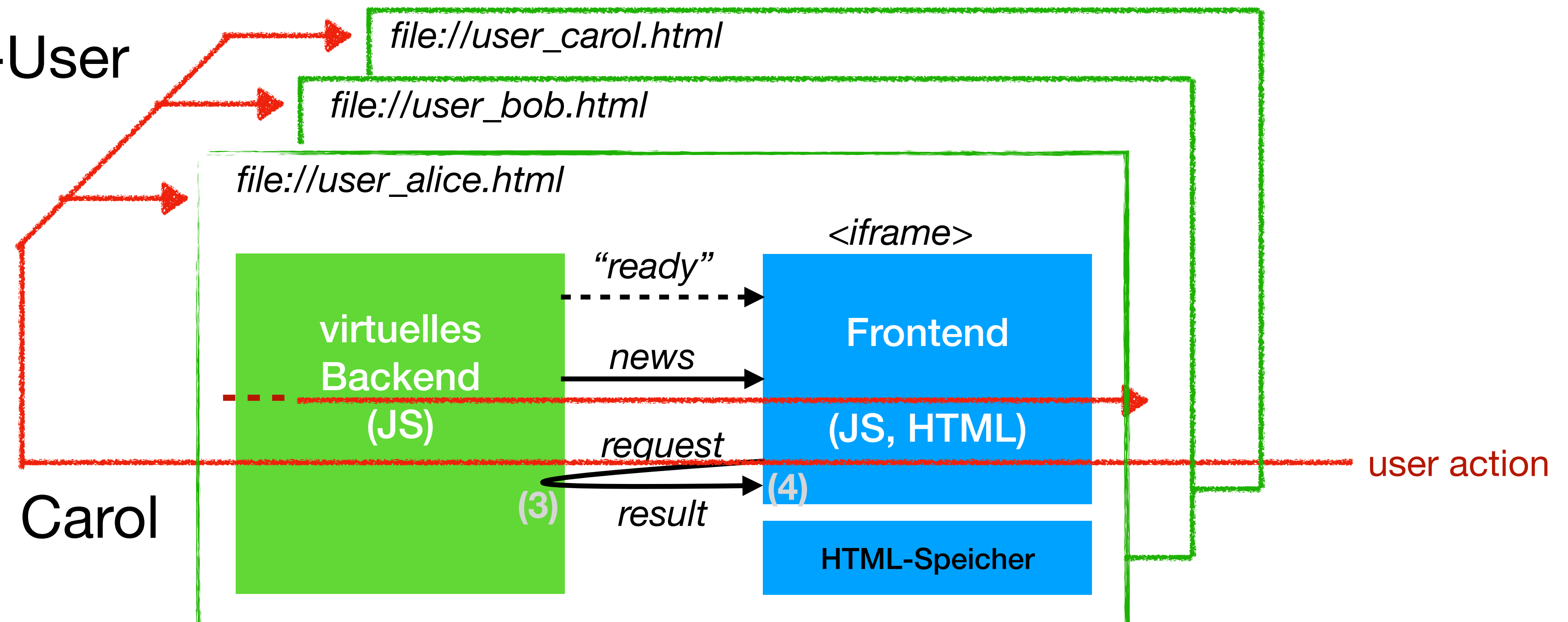


# virtuelles Back-End (in JS+HTML)

Entwicklung der App-Logik ist nun ausserhalb von Android möglich:  
“**virtuelles Backend**” ersetzt den Kotlin- und Android-Teil.

Wie? Im Chrome-Browser!

- ein Tab pro Tremola-User
- Kommunikation erfolgt intern im Browser
- drei vorinstallierte User Alice, Bob und Carol





# erlaubt Simulation von On/Offline

Stand der Warteschlangen  
für ein- und ausgehende Log-Einträge

Carol ist Offline (toggle)

Reset-Knopf: löscht den  
Speicher aller drei User ...

.. Reload der Software muss  
zusätzlich in jedem Tab einzeln  
ausgelöst werden!

Vector-Clock: höchste bisher  
gesehene Sequenznummern

Liste der empfangenen  
Log-Einträge

**user\_alice.html**

This is Alice **ONLINE** (0/0 queued)

Alle Drei  
[me, Bob, Carla]

Hi Bob and Carla Wed Jul 13 2022 07:08

wie geht es Euch? Wed Jul 13 2022 07:09

Tap here to type

vectorClock = {"Alice":2,"Bob":0,"Carla":0}

- 7:09:21 {"from":"Alice","seq":2,"type":"post"}
- 7:08:53 {"from":"Alice","seq":1,"type":"post"}

**user\_bob.html**

This is Bob **ONLINE** (0/0 queued)

TREMOLA

local notes (for my eyes only)  
[me]

Unnamed conversation [Alice, me, Carla] 2

vectorClock = {"Alice":2,"Bob":0,"Carla":0}

- 7:09:21 {"from":"Alice","seq":2,"type":"post"}
- 7:08:53 {"from":"Alice","seq":1,"type":"post"}

**user\_carla.html**

This is Carla **OFFLINE** (1/0 queued)

TREMOLA

local notes (for my eyes only)  
[me]

Unnamed conversation [Alice, Bob, me] 1

vectorClock = {"Alice":1,"Bob":0,"Carla":0}

- 7:08:53 {"from":"Alice","seq":1,"type":"post"}

deshalb hat sie erst  
einen Log-Eintrag erhalten



# Log-Eintrag vom Backend zum Frontend

Wird als JavaScript-Objekt ausgeliefert

```
▼ 2:  
  ▶ header: {tst: 1716416562050, ref: 107305, fid: '@AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=.ed25519', seq: 10}  
  ▶ public: (4) ['TAV', 'hello world', null, 1716416562]
```

Header: Metadaten des Log-Eintrags

(Timestamp, Hashwert, Absender, seq-Nr)

Public: Inhalt des Log-Eintrags, eine Liste

Log-Eintrag ist Liste mit:

'TAV', 'KAN' - bestimmt für welche Applikation dieser Eintrag bestimmt ist,  
gefolgt von beliebig vielen Argumenten

Beispiel 2:

```
▼ 2:  
  ▶ header: {tst: 1716416928065, ref: 643024, fid: '@AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=.ed25519', seq: 11}  
  ▶ public: (5) ['KAN', 'null', 'null', 'board/create', 'new Kanban board']
```

# Walk Through für IAM (“I am”)

Die IAM-App: beim Definieren des eigenen Display Names wird der neue Wert der ganzen Welt mitgeteilt.

Format Log-Eintrag: [“IAM”, “Alice”] (der öffentliche Schlüssel ist die FeedID selbst)

Die beiden Event Handlers:

*on user input:*

if my display name changed then

- Frontend sends "iam QWxpY2U=" to Backend (base64-encoded display name)
- Backend creates log entry with content [“IAM”, “Alice”]

*on incoming event* [“IAM”, “Bob Jr.”] in Bob’s log:

if Bob’s display name not set manually then  
set it to received value

# Eigenheiten des virtuellen Back-Ends

Drei Browser-Tabs ersetzen keine richtigen, unabhängigen Geräte:

- alle drei Tabs müssen geöffnet sein, *bevor* eine App-Aktion erfolgt
- Grund: ein Nachzügler-Tab kann alte Log-Einträge nicht anfordern

Ablauf bei Änderung der Software:

- bei den drei Tabs einzeln ein Reload auslösen, **und**
- den “Reset”-Button drücken (re-initialisiert alle drei Clients)

Wenn ein User offline gestellt wird, werden Log-Einträge für ihn/ihr gesammelt und beim Online-Gehen diesem User eingespielt.



# In Chrome: “Developer Tools” nutzen

Menu:

View ->

Developer ->

Developer Tools

