FLEX - Frontend Integration

Simon Härer, Michael Sperber

Created: 2025-12-02 Tue 13:19

User Experience

Currently there are three separate frontends. The user should have an integrated experience, not have to browse through many frontends.

Frontend Integration

- A user should not be hindered considerably by our architecture
- A frontend should be usable as if it is one application
- Examples: Amazon, Ebay, …

JavaScript SPA vs. plain HTML

- powerful JS frameworks & other languages
- HTML composes, via server-side includes
- JavaScript composes poorly

@ctive group

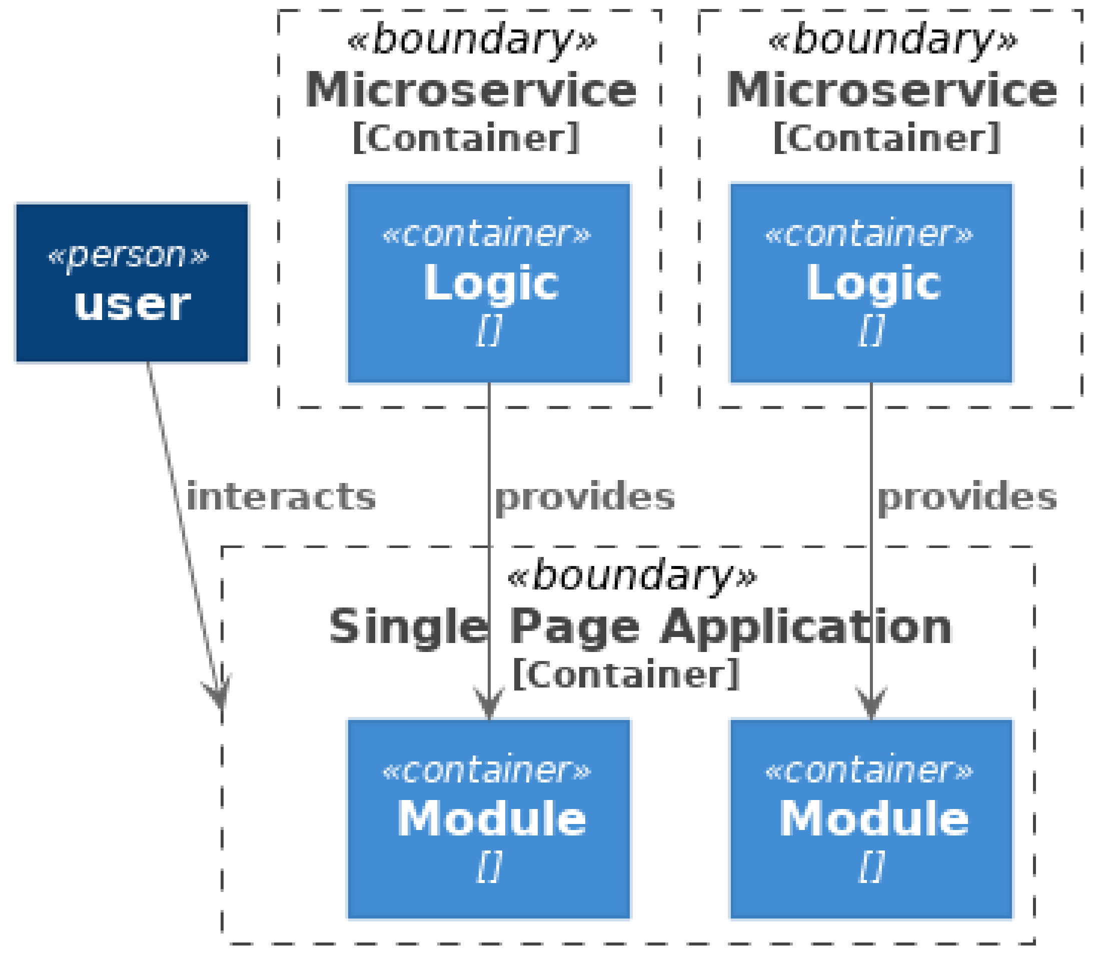
One SPA for each microservice

- inconsistent user experience
- asset server for unified look and feel
- links to switch between microservices' SPAs
- each SPA must be loaded & cached separately

One SPA for all microservices

- integrate microservice frontends as modules
- specification of interfaces between modules necessary, e.g. events
- when interfaces change, teams must coordinate the changes
- deployment of SPA depends on multiple teams again

One SPA per microservice



Resource Oriented Client Architecture (ROCA)

- Based on resource internetworking in a RESTful setup
- HTML with conventional hyperlinks is used as a basis
- Application logic must be implemented on the server only to avoid redundancy

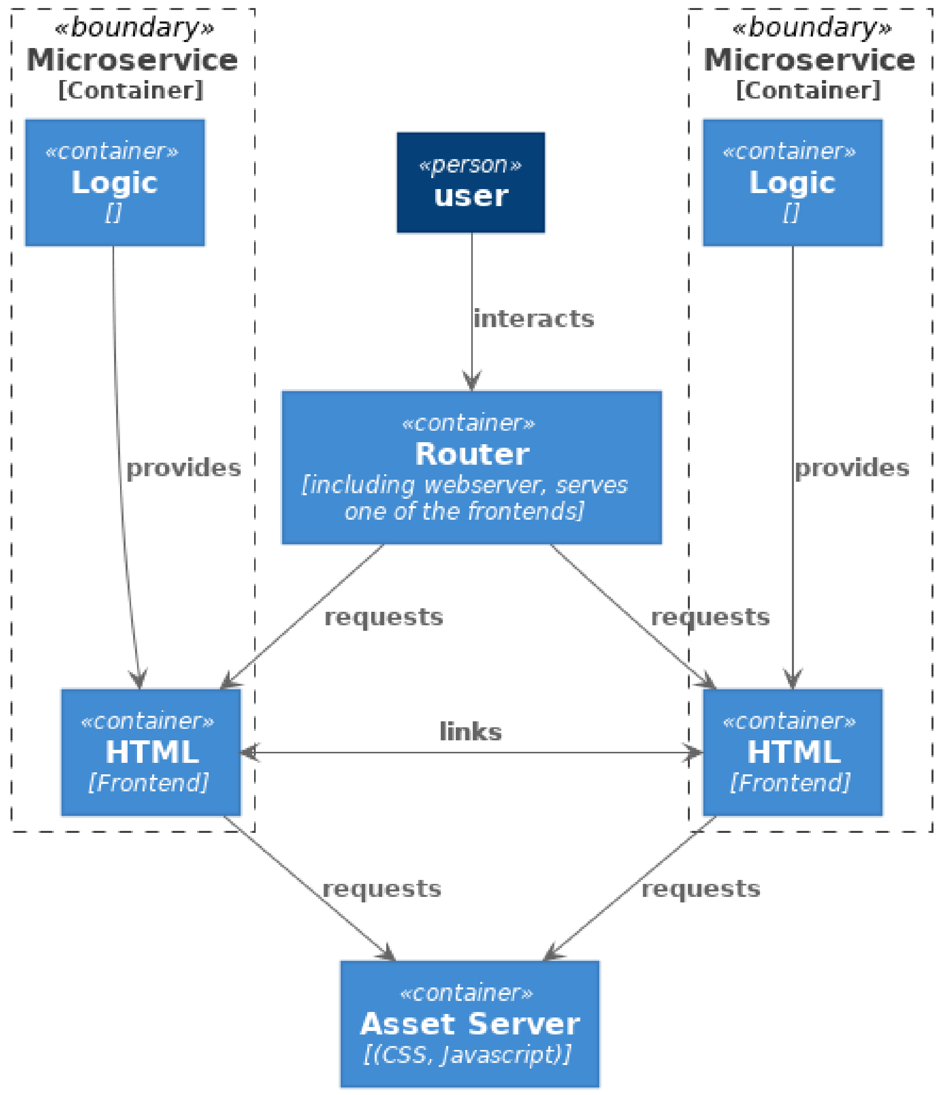
- Javascript is then only used as a incremental enhancement. That is, not using Javascript as a functional requirement, but as an enrichment.

  https://roca-style.org/

Resource Oriented Client Architecture (ROCA)

- microservice frontends are integrated using links
- asset server for unified look and feel
- router can unify URIs
- Technologies based on ROCA: Zuul (Netflix), various reverse proxies

ROCA



@ctive group

Server Side Includes

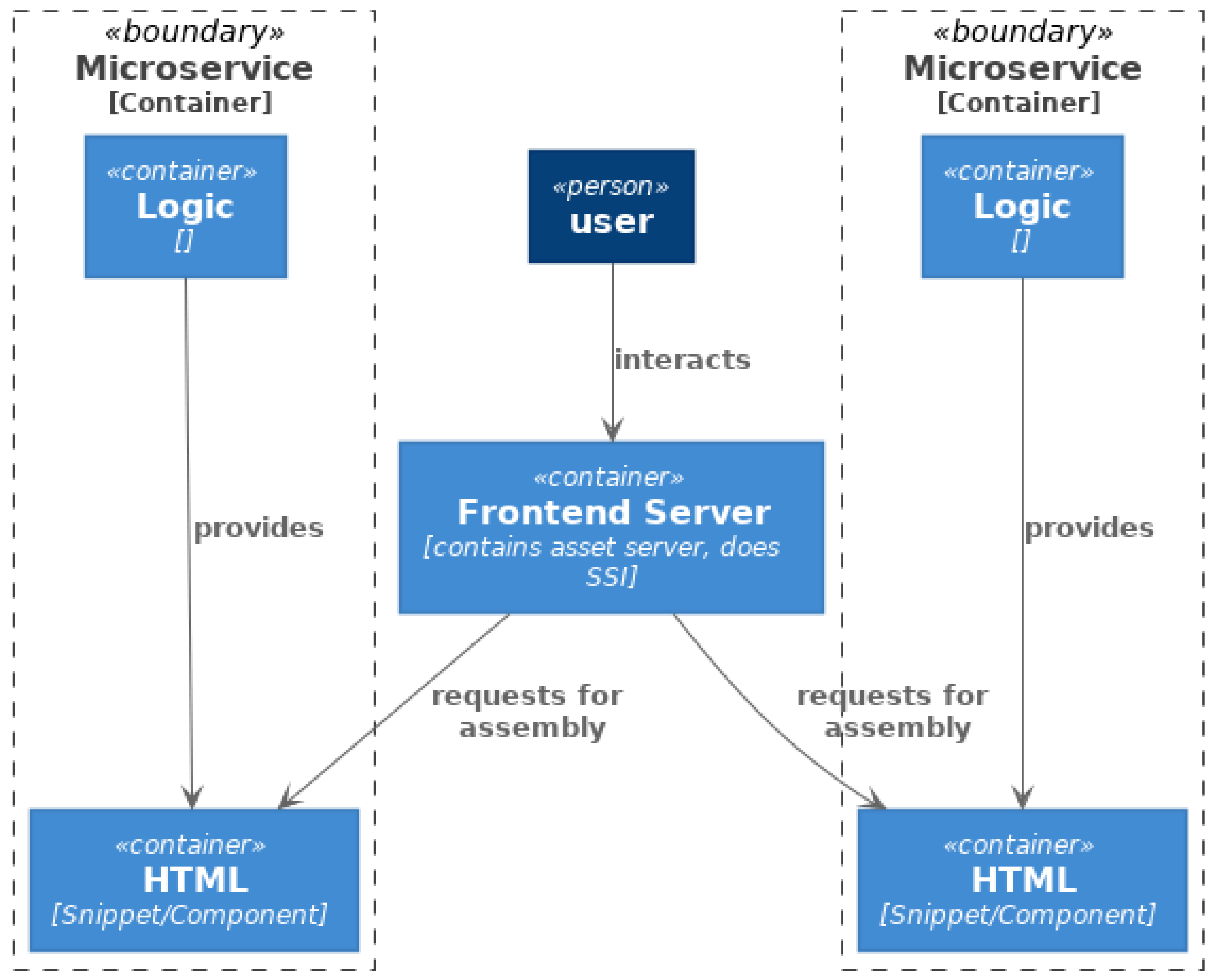
With server side includes, webpages are assembled on the webserver using simple directives. A HTML template can be defined that puts together frontend components from various microservices.

```
<!DOCTYPE html>
<html>
    <body>
        <!-- ... some content -->

        <!-- Embed the content of another page here -->
        <!--#include virtual="https://..." -->

        <!-- ... more content -->
    </body>
</html>
```

Server Side Includes (SSI)

- frontend is assembled server side, the customer does not notice
- the assembled parts can be cached separately
- asset server for unified look and feel
- Technologies: E.g. Nginx supports SSI

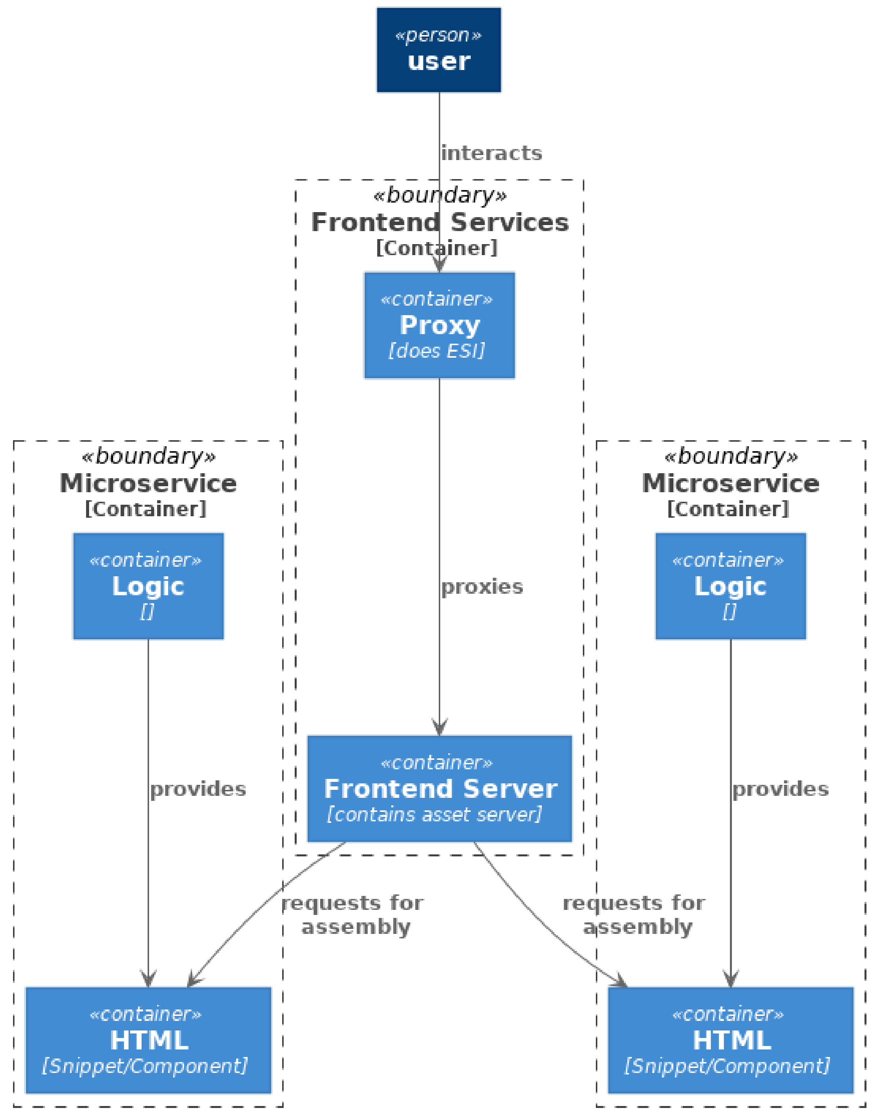
Server Side Includes (SSI)



Edge Side Includes (ESI)

- Similar to server side includes, but not by the web server, but a specialized middleware. These can be proxies for example.
- Different components can be cached individually, usually allows more cashing control
- Technologies: Mostly proxies, as Varnish, Squid, ...

Edge Side Includes (ESI)



Web Components / Micro Frontends

```html
<my-alert type="warn">
  <h4>Achtung</h4>
  <p>Etwas schlimmes ist passiert!</p>
</my-alert>
<my-alert type="success">
  <h4>Gl&uuml;ckwunsch</h4>
  <p>Alles im gr&uuml;nen Bereich!</p>
</my-alert>
```

```javascript
customElements.define('my-alert', MyAlert);
```

Challenges with Web Components

- duplicate definitions
- initialization order
- non-composable frameworks / versions
- but see "module federation"

Exercise: Drawbacks of Frontend Integration

What disadvantages can the participants identify with the mentioned methodologies and technologies?

Discuss specifically security, response time, and latency

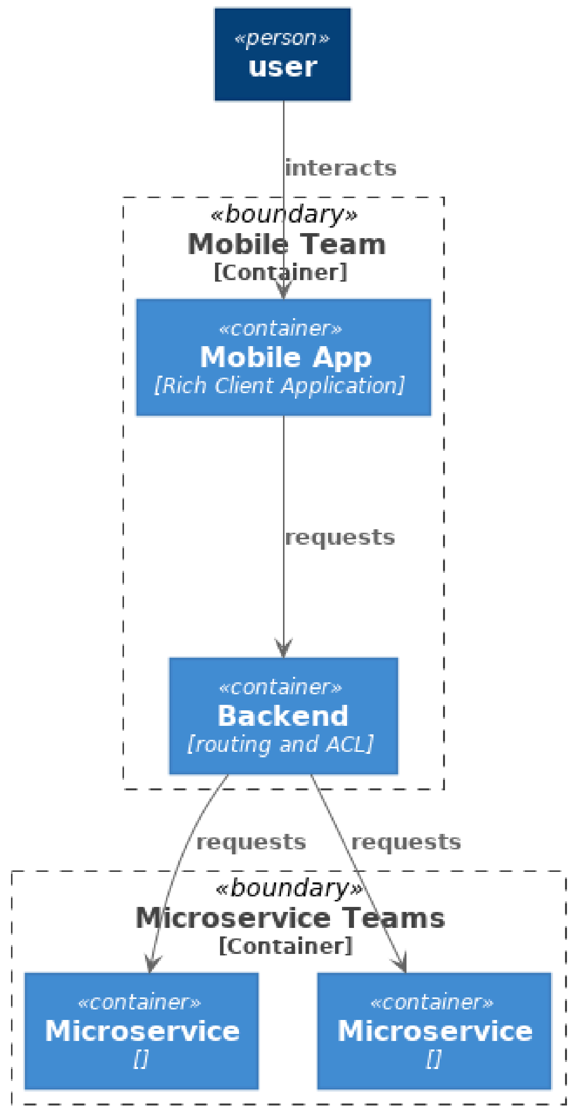
Exercise: What can non-web frontend integration look like?

Discuss and find possible other methods for non-web frontend technologies. Think about native apps, windows applications, …

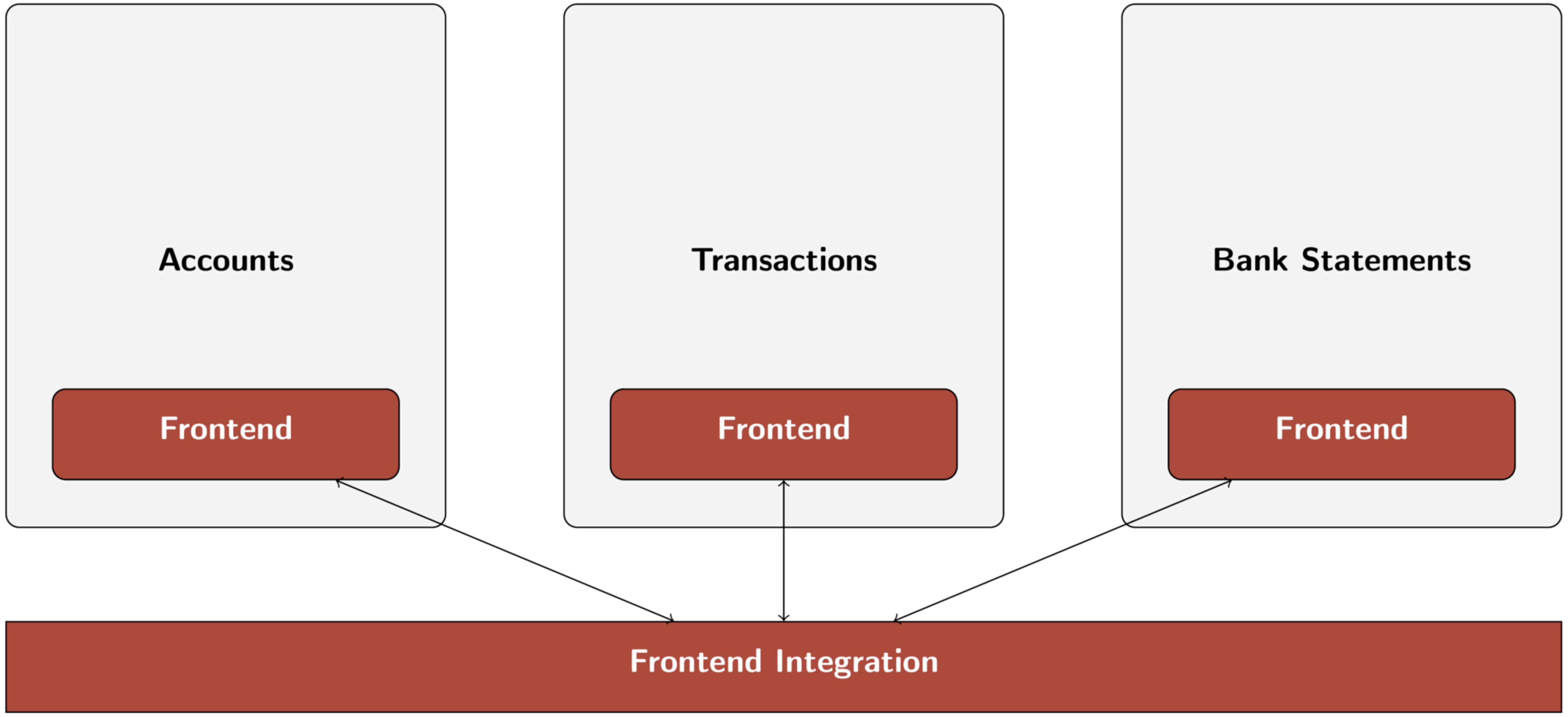
Rich Client Applications in a Microservice Landscape

- separate frontend and backend development
- client as a deployment monolith developed by a special team
- offer API for microservices
- ... or add backend layer to abstract over microservice interfaces

Rich Client Applications with Backend Layer



Erlbank


Accounts
Transactions
Bank Statements
Frontend
Frontend
Frontend
Frontend Integration

Exercise: Erlbank - Which Frontend Integration fits?

Discuss which of the mentioned front-end integration methods fit.

Frontend Integration: SSI in Nginx Config

```
server {
  location / {
    ssi on;
    ...
  }

  location /accounts/ {
    ssi on;
    proxy_set_header Accept-Encoding "";
    proxy_pass http://${ACCOUNTS_HOST}:8000$request_uri;}

    location /transfers/ {
      ssi on;
      proxy_set_header Accept-Encoding "";
      proxy_pass http://${TRANSFERS_HOST}:8001$request_uri;
      ...
    }
```

@ctive group

Frontend Integration: SSI in HTML

```html
<!DOCTYPE HTML>
<html>
  <head>
    <title> ERLBANK </title>
  </head>
  <body>
    <h1> ERLBANK </h1>

    <!--# include virtual="/accounts/" -->
    <!--# include virtual="/transfers/" -->
    <!--# include virtual="/statements/" -->

    <p> Now:  <!--#echo var="DATE_LOCAL" -->  </p>
  </body>
</html>
```

@ctive group