

Scaling your project with Micro-frontends

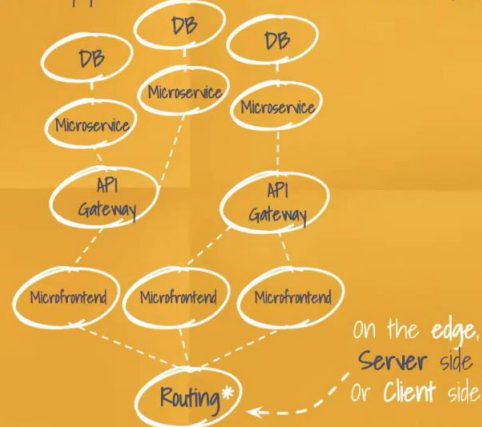
Luca Mezzalana

VP of Architecture at DAZN

Scaling our applications

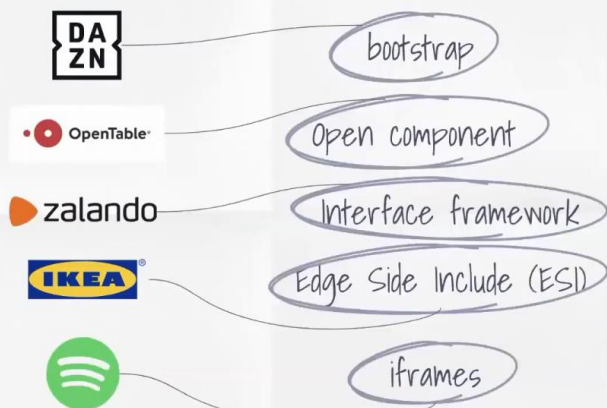


Scaling our applications with microfrontends



From Domain
Driven Design
(DDD)

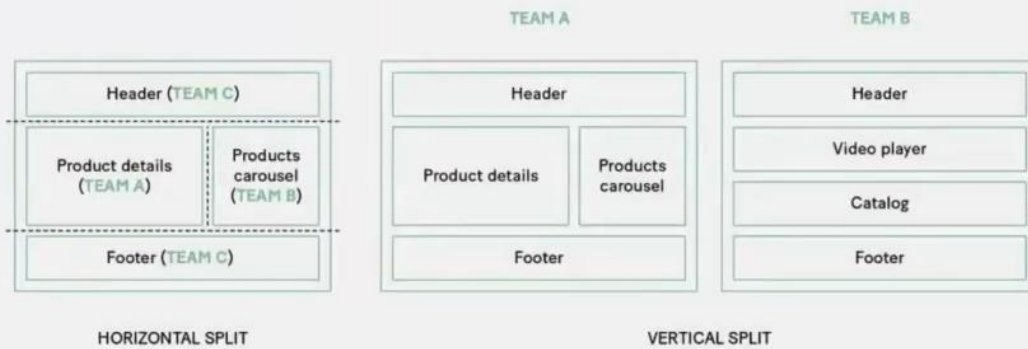
Micro-frontends are the technical representation of a business subdomain, they allow independent implementations with same or different technology choices, finally they should avoid sharing logic with other subdomains and they are own by a single team



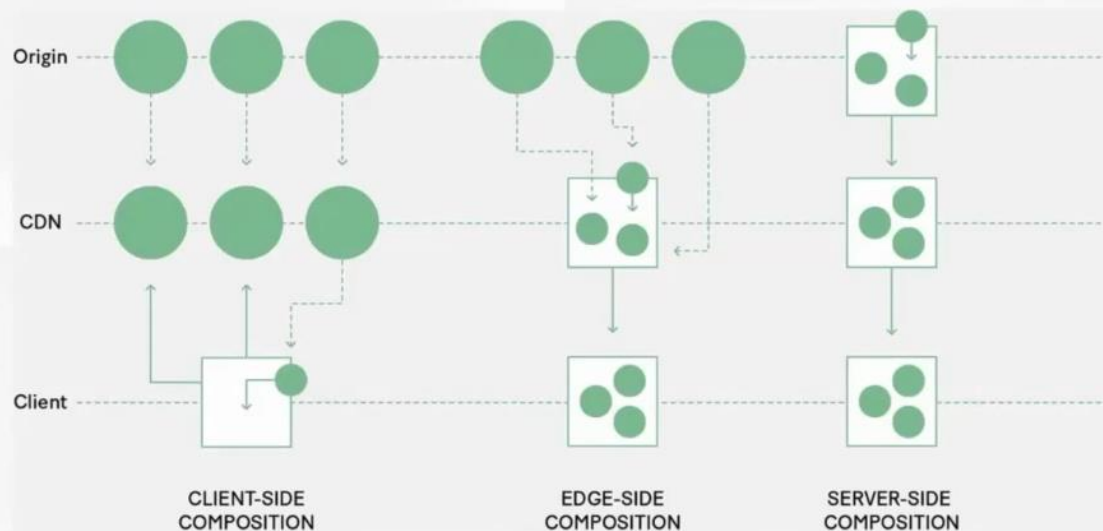
Micro-Frontends Decisions Framework

Identify, Compose, Route and Communicate

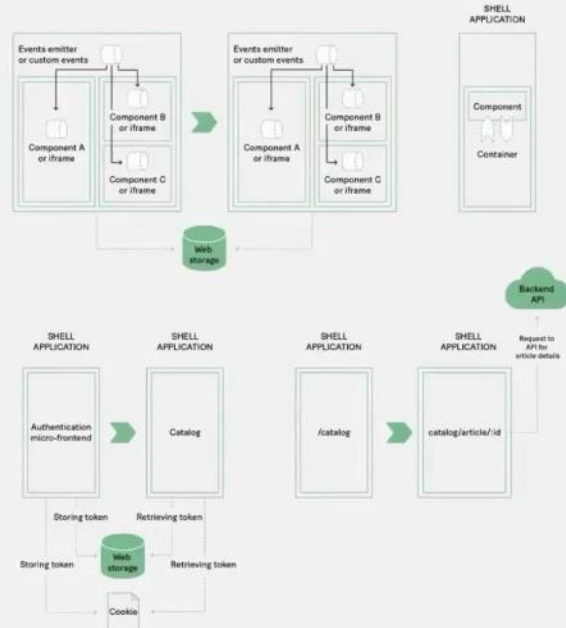
Identify a micro-frontend



Compose and Route a micro-frontend



Micro-frontends communication

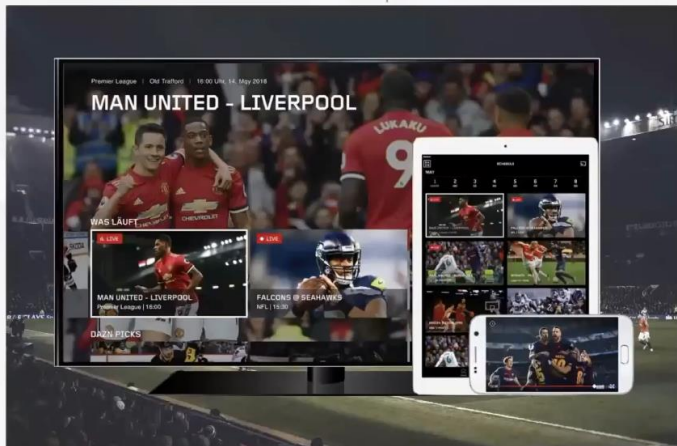


How does MFEs on the same view communicate? You can use events where each MFE has an eventEmitter (MFE-to-Container communication) or custom events (MFE-to-MFE communication). Communication across MFEs on different pages can use web storage (e.g. localStorage) or querystrings on the destination URL



bootstrap

Multi Device experience



Design for hundreds of developers

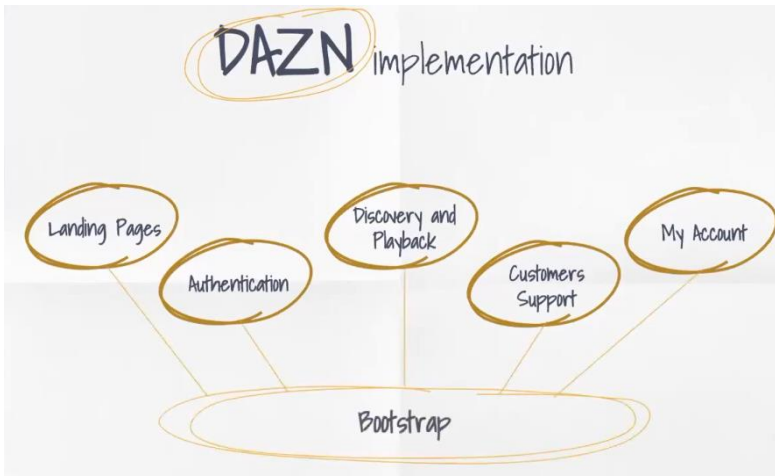


“We are going to have **hundreds of developers** working on DAZN.”

Bear in mind when you design our **new architecture!**”



We are currently targeting about 40 different devices with mostly HTML5 and JS.

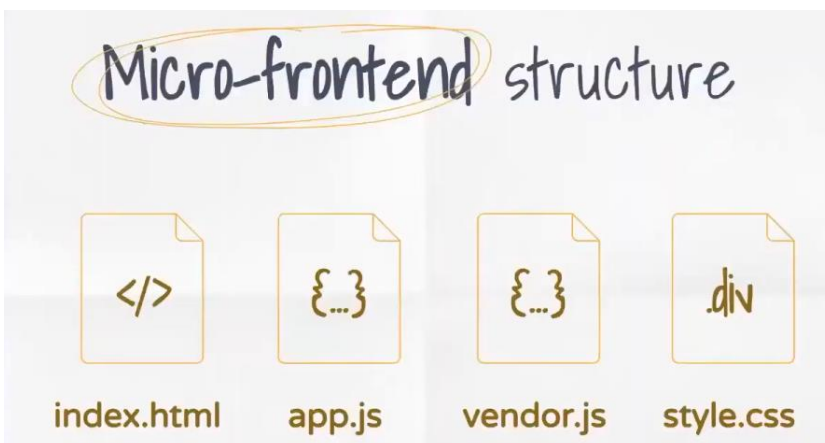


We checked to see patterns of how the users were using our applications. We don't have to load all the functionalities as a SPA if the users are going to only a few pages or sections only during a visit. We can separate functionalities into MFEs that are loaded as needed.

DAZN implementation

Micro-frontends

- Each **Micro-frontend** represents a **subdomain** matching the business structure
- It's **technology/framework agnostic**
- A Micro-frontend is **AUTONOMOUS**
- Inside a Micro-frontend the team can share components, code, styles or any other asset
- Independent building systems
- **1 Micro-frontend loaded per time**
- **1 team own 1 Micro-frontend**



How bootstrap works



www.dazn.com



bootstrap

Bootstrap is responsible for:

- . application startup
- . I/O operations
- . routing between micro-frontends
- . sharing configurations across multiple micro-frontends

The bootstrap is vanilla JS that is embedded inside the initial index.html page and does things like understanding the country the user is, the device and configuration (has localStorage or not?) they are using, are they authenticated or not (does valid JWT exist in localStorage), and expose an API for the MFE to get data and persist data.

How bootstrap works

```
<html>
  <head>
    <script src="/bootstrap.js" />
  </head>
  <body>
    ...
  </body>
</html>
```

- Bootstrap loaded as first element and always available
- Tiny JS layer responsible to load micro-frontends
- Exposes APIs for different micro-frontends

How bootstrap works

```
<html>
  <head>
    <script src="/bootstrap.js" />
    <style type="text/css">...</style>
  </head>
  <body>
    <div>
      ...
    </div>
    <script src="/catalogue.js" />
    <script src="/cat-vendor.js" />
  </body>
</html>
```



The bootstrap works and then appends the correct styles and MFEs into the page.

How bootstrap works

```
Window.DAZN = {  
  Lifecycle: {  
    onLoad: function(){}  
    onunload: function(){}  
  },  
  LocalStorage: {},  
  ...  
}
```

- DAZN object exposes methods and properties for all the micro-frontends
- Each micro-frontend has lifecycle callbacks available
- This object abstracts the platform exposing common APIs

To remove MFEs on the page, the bootstrap will unload (removing the listeners, etc) the MFE as needed.

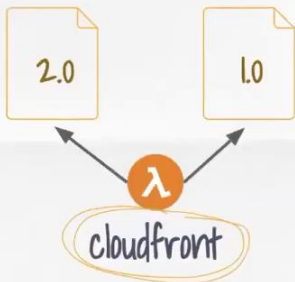
Components



- Components available on NPM private repos
- Components need to work with any framework
- They expose a contract for the micro-frontend to interact with
- Components are own by specific teams
- They can be shared within same team

We publish certain complex things like the video player as NPM packages in our private NPM repository. Components are loaded at compile time, but can also be loaded at run time.

Deployment



- Based on some scenarios I can redirect the user to a version or another
- Don't need to do a big bang deployment
- Canary releases or Blue Green deployment on Frontend! ([Amplify-like](#))

We use serverless Lambda-at-the-edge with CloudFront as CDN to serve data from the closest location to the user.

5 teams



Onboarded on the same projects in 3 weeks

Over 400 techies

Working on the same platform

De-risk our releases

2 years without S1 incidents, multiple releases per day, doubling the user base every year