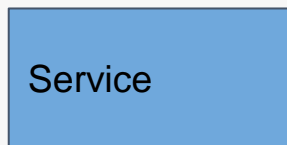
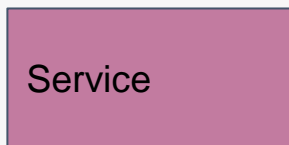
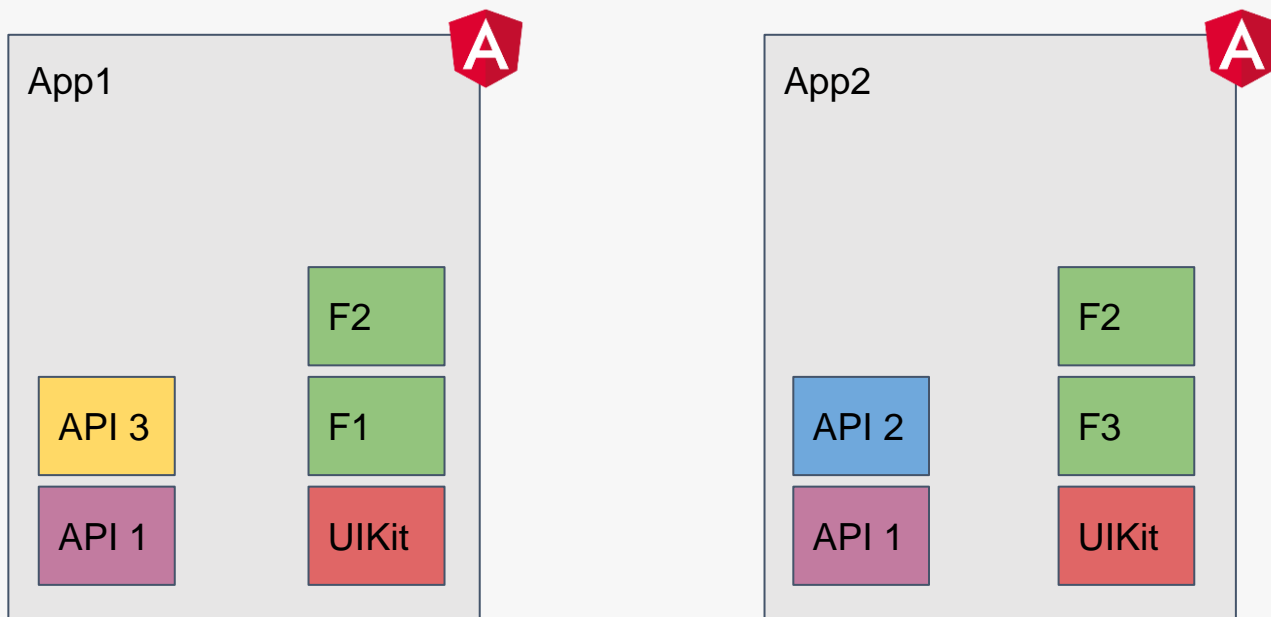
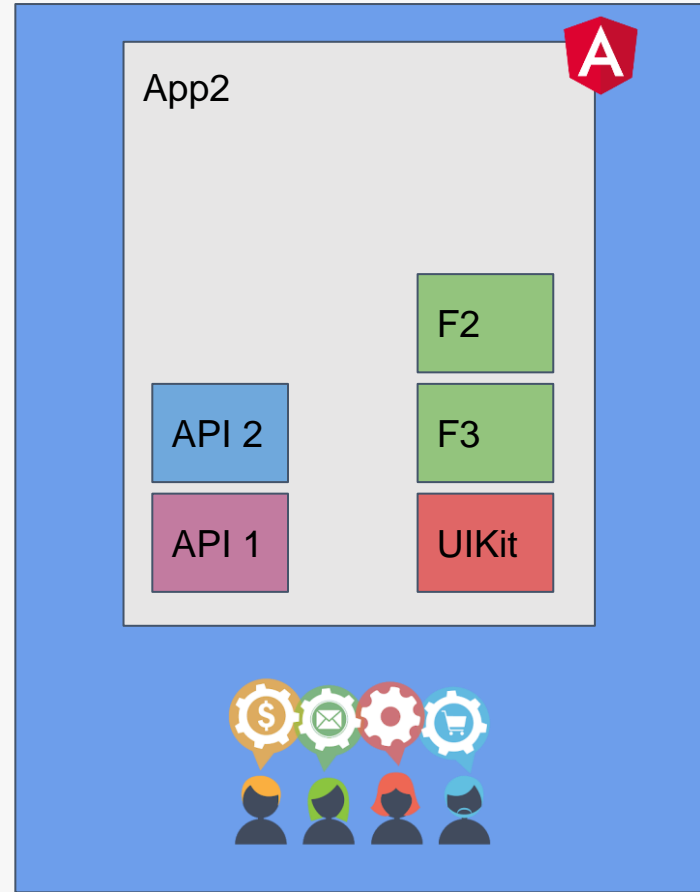
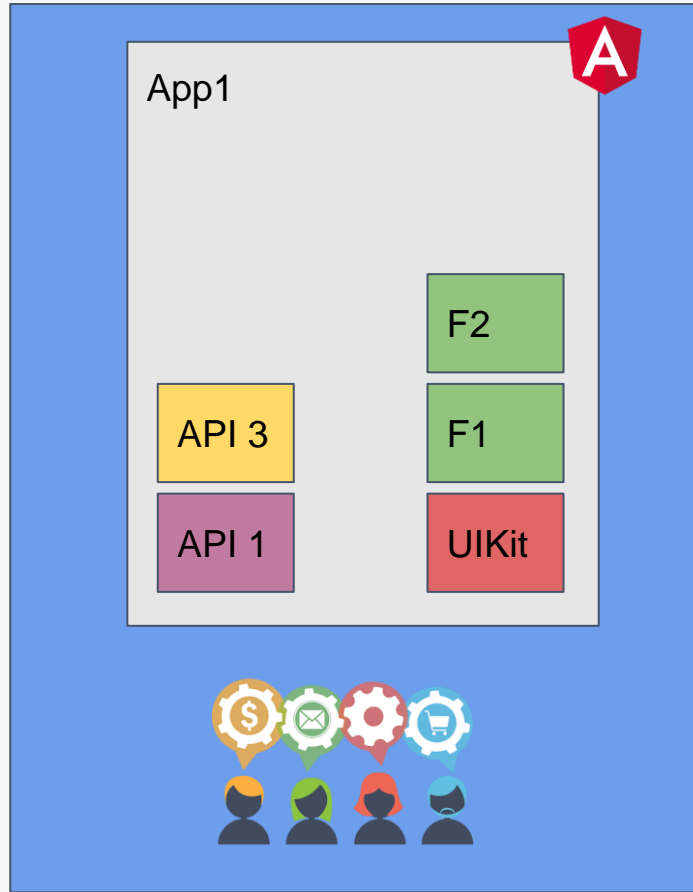


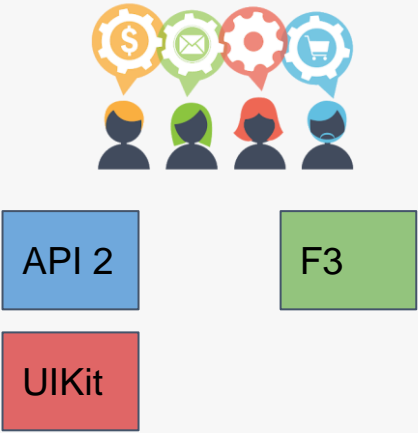
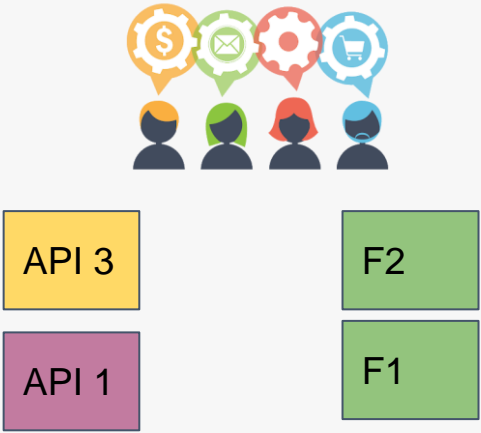


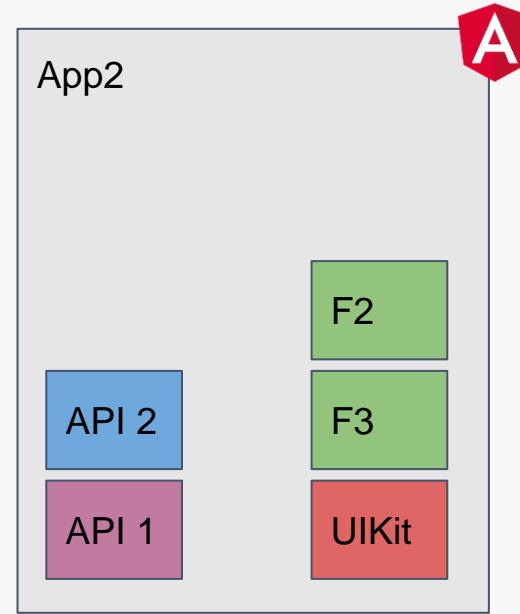
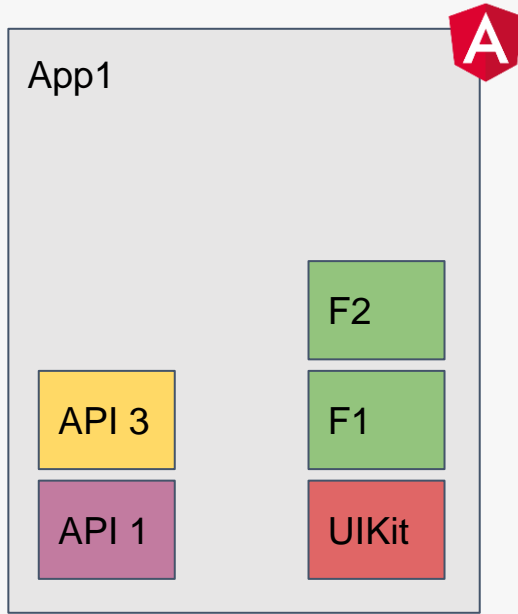
Micro Frontends

Fravezzi Mattia
m.fravezzi@almaviva.it

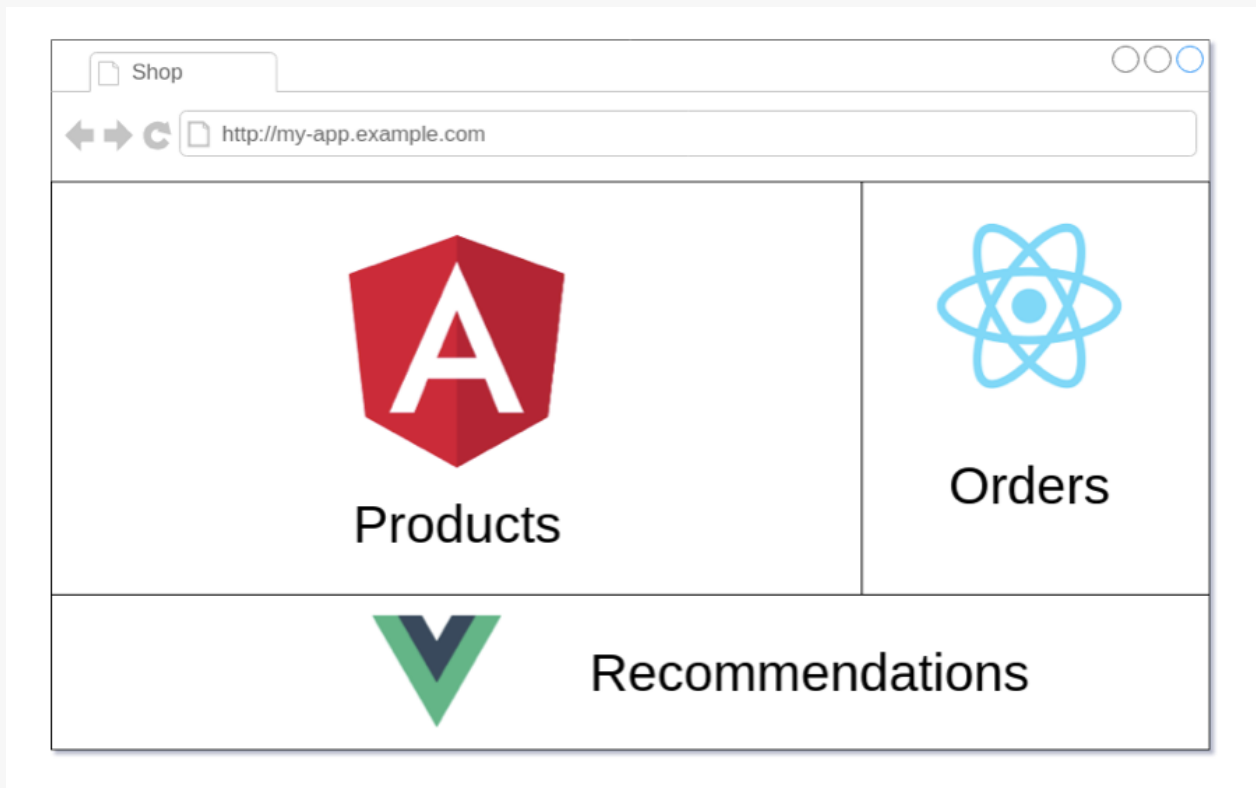








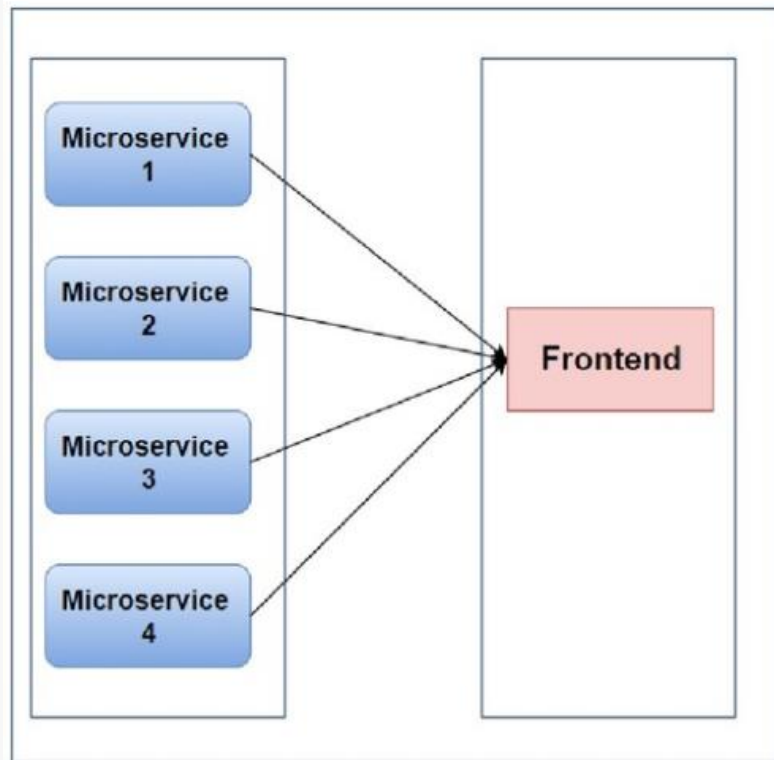
Cos'è una architettura a Micro frontends



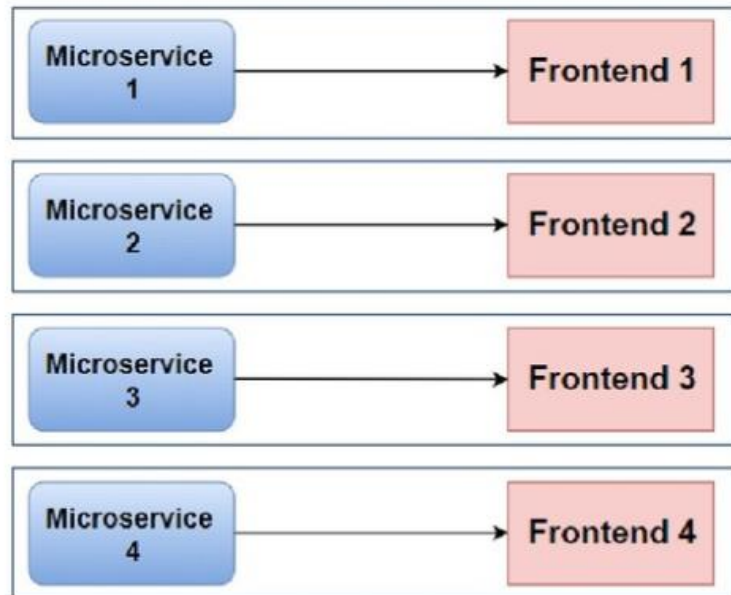
Micro Frontends Principi

- Sviluppo indipendente
- Testing indipendente
- Deployment indipendente
- Decentralizzato (Tecnologia indipendente)
- Modellato intorno al business domain
- Isolazione dei fallimenti
- Osservabile (monitoring intera applicazione)

Monolith Frontend



Micro Frontends



Monolithic

Pro

- Architettura semplice
- Facile da sviluppare
- Facile da testare
- Facile da deployare
- Facile da Scalare

Cons

- Complessità aumenta maggiore è il software
- Incrementa tempo di startup
- Diminuisce leggibilità
- Aumenta difficoltà di scrivere codice
- Modificare sezioni rischia di impattare sul software
- Aumento costo testing per integrare nuove funzionalità
- CI/CD diventa difficile
- Se down una sezione va down tutta l'applicazione

Micro frontend

Pro

- + Leggibilità
- Facile da sviluppare
- Facile da testare
- Facile CI/CD (indipendenti)
- Ogni Fe sviluppato da un team diverso
- Imprevisti su integrazione di codice sono ridotti
- - bugs e conflitti
- - costo di test e debugging
- - down time
- - costi di hosting / resource

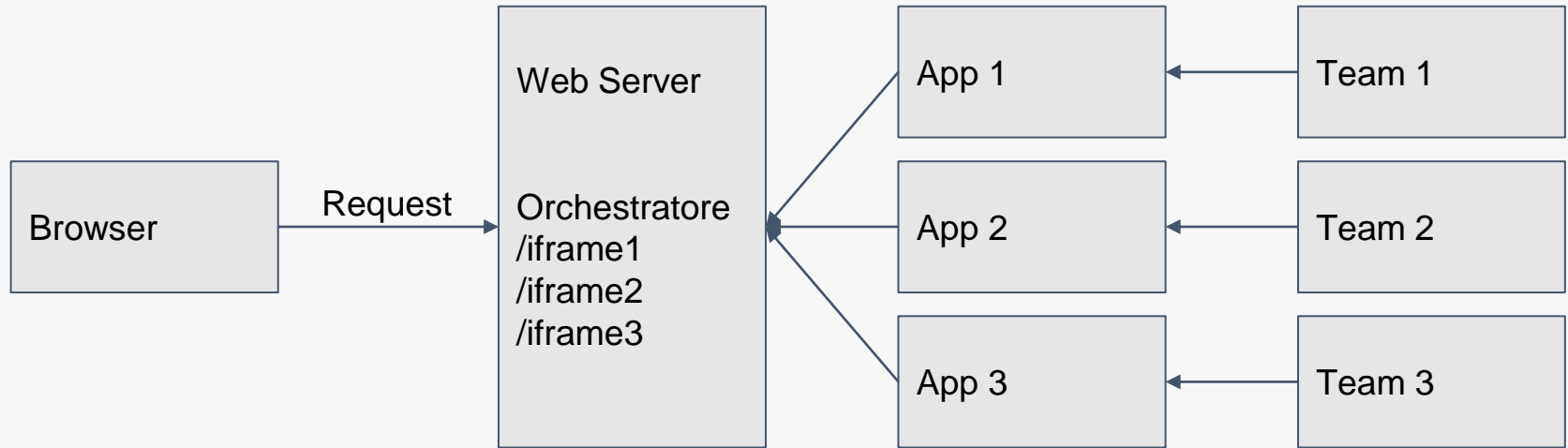
Cons

- + expertise e competenze diverse (devOps etc..)
- + complessità in sistemi distribuiti
- Implementazione di sistemi di comunicazione tra i microservizi
- Computer che riescono ad eseguire più Fe simultaneamente

Tecniche per i micro frontends

- IFrames
- Shell as a Proxy
- Web Components (Angular Elements)
- Reverse proxy
- Using API Gateways
- Moduli Federati

IFrame Architecture



IFrame

Pro

- Team e Progetto completamente indipendenti
- Applicazione interamente incapsulata nel iframe, non interagisce con l'host

Cons

- più iframe nella stessa pagina causa problemi di performance
- codice duplicato
- problemi di sicurezza (chrome!)
- UX/UI diventa complessa da gestire (dimensionamento)
- Scaling

IFrame

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My App</title>
  <base href="/">
</head>
<body>
  <iframe src="http://myapp.com/products"></iframe>
  <iframe src="http://myapp.com/orders"></iframe>
  <iframe src="http://myapp.com/recommendations"></iframe>
</body>
</html>
```

IFrame - Come comunicano?

```
window.parent.postMessage({ type: 'ADD_PRODUCT', data: myData }, '*');
```

```
windows.addEventListener('message', receiveMessage, false);
```

```
function receiveMessage(msg): void {
```

```
    switch (event.data.type) {
```

```
        case: 'ADD_PRODUCT':
```

```
            //to something
```

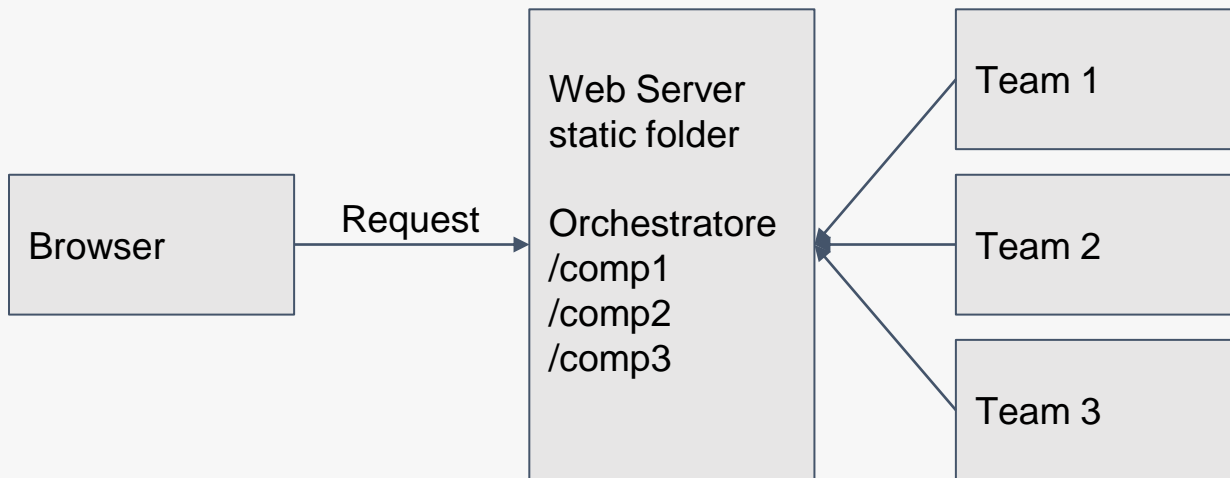
```
    }
```

```
}
```

```
postMessage$ = fromEvent(window, 'message').map(res => res.data);
```

```
addProduct$ = postMessage$.filter(item => item.type === 'ADD_PRODUCT');
```

Angular Elements Architecture



Angular Elements

Pro

- La comunicazione tra componenti avviene come fosse un normale componente angular (input e output)
- Usando lo shadowDom è possibile avere una completa indipendenza dall'host
- Indipendente dagli altri components su tecnologie e rilasci
- Sono facilmente riutilizzabili
- Basso effort nel trasformare componenti esistenti


Cons

- Performance ridotte
- Codice duplicato
- Non è supportato da tutti i browser
- Difficile creare componenti che siano consapevoli del contesto in cui vengono usati.
- SEO problems

Angular Elements

```
export class AppModule {
  constructor(private injector: Injector) { }

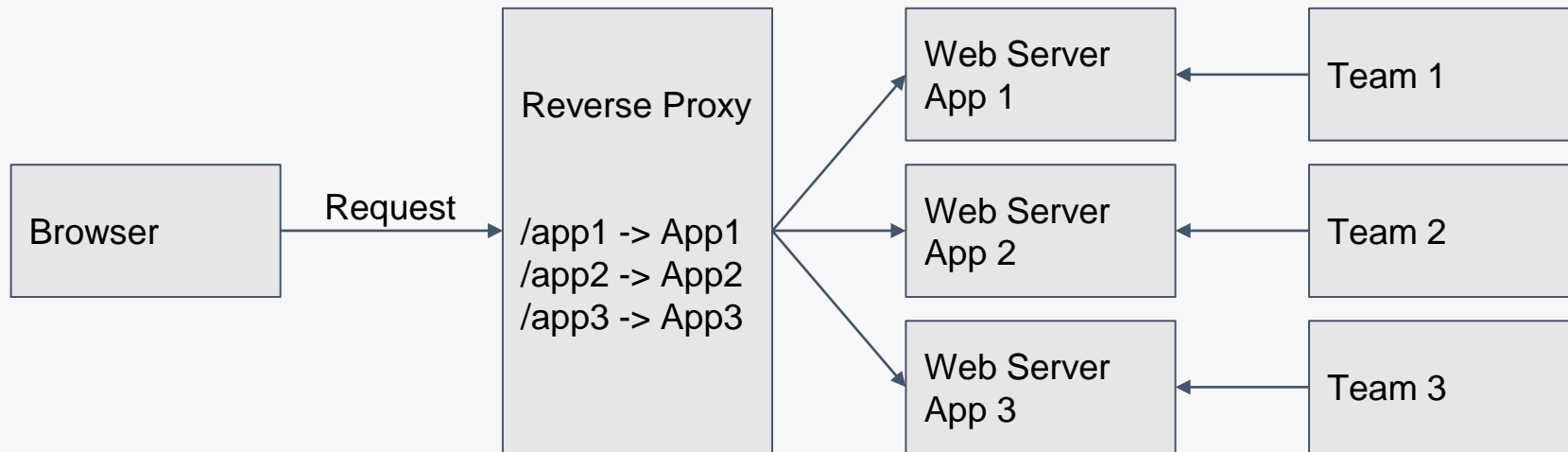
  ngDoBootstrap() {
    var customElement = createCustomElement(AppComponent, { injector: this.injector });
    customElements.define("almaviva-planimetria-widget", customElement);
  }
}
```



```
<script defer type="text/javascript" src="assets/web-components/almaviva-planimetria-widget.js"></script>

<almaviva-planimetria-widget></almaviva-planimetria-widget>
```

Reverse Proxy Architecture



Moduli Federati

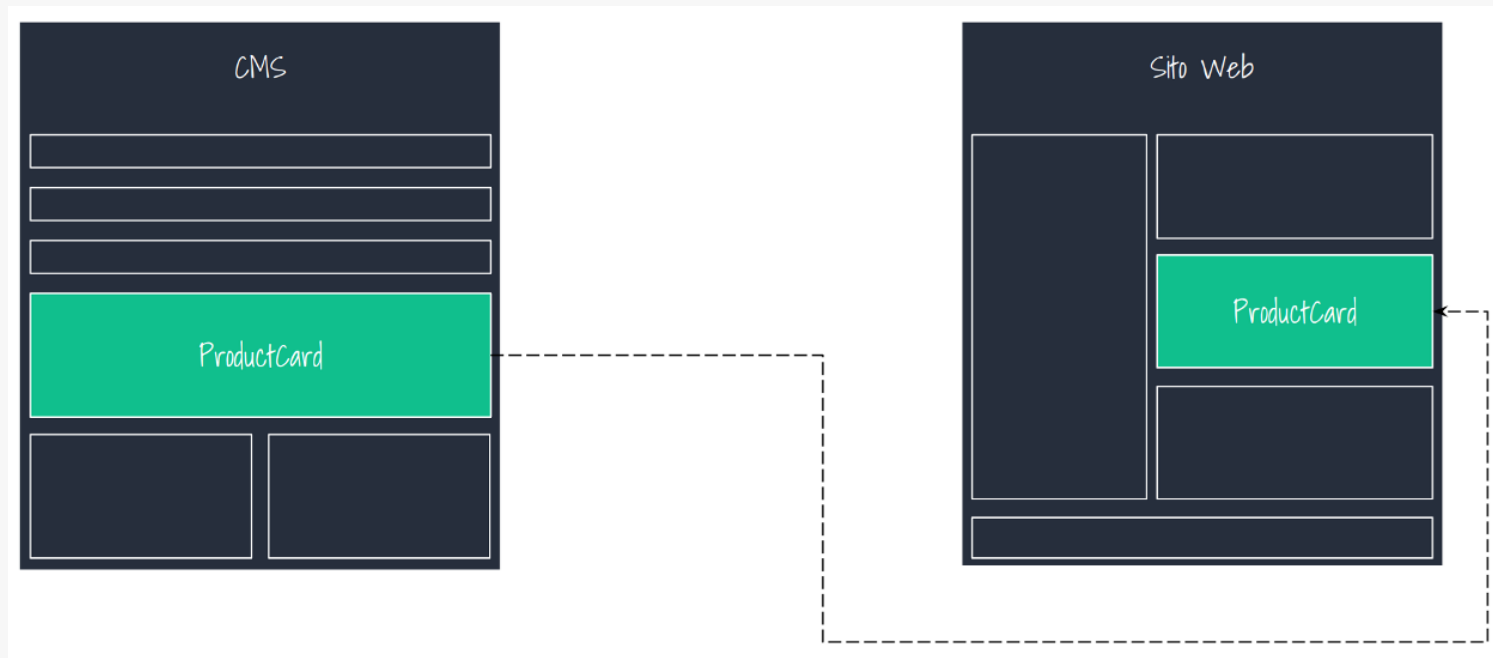


Webpack



Module Federation

Moduli Federati



Moduli Federati

Pro

- Codice sorgente del widget rimane nel posto in cui dovrebbe essere.
- Non abbiamo la necessità di utilizzare framework per micro-frontends
- Non abbiamo necessità di loader personalizzati.

Cons

- Esecuzione dei moduli federati a runtime
- Bug nel modulo causa malfunzionamenti in tutti i moduli che lo usano

```
const ModuleFederationPlugin = require("webpack/lib/container/ModuleFederationPlugin");
module.exports = {
  output: {
    uniqueName: "lottery"
  },
  plugins: [
    new ModuleFederationPlugin({
      name: "lottery",
      filename: "remoteEntry.js",
      exposes: {
        './Module': './apps/lottery/src/app/app.module.ts',
      },
      shared: {
        "@angular/core": { singleton: true, strictVersion: true },
        "@angular/common": { singleton: true, strictVersion: true },
        "@angular/router": { singleton: true, strictVersion: true },
      }
    })
  ]
}
```

```
const ModuleFederationPlugin = require("webpack/lib/container/ModuleFederationPlugin");

module.exports = {
  output: {
    uniqueName: "community-software"
  },
  plugins: [
    new ModuleFederationPlugin({
      remotes: {
        "lottery": "lottery@http://localhost:5001/remoteEntry.js",
      },
      shared: {
        "@angular/core": { singleton: true, strictVersion: true },
        "@angular/common": { singleton: true, strictVersion: true },
        "@angular/router": { singleton: true, strictVersion: true },
      }
    })
  ]
}
```



```
{
  path: 'lottery',
  loadChildren: () => import('lottery/Module').then(m => m.LotteryModule)
}
```



```
declare module 'lottery/Module'
```

Monorepo



DEMO



almaviva.it