LeanIX

Battle tested strategy for Angular Micro Frontends using web components

Learnings from 2 years of micro frontends at LeanIX

Konstantin Tieber, Staff Software Engineer

xkons.de

Resources

LeanIX

Blogs

traveling-coderman.net/code/microfrontends/why/

xkons.de/posts/ng-micro-frontends/

Code

github.com/fboeller/microfrontends-with-angular/

Demo App

ng.traveling-coderman.net/

Video Tutorial

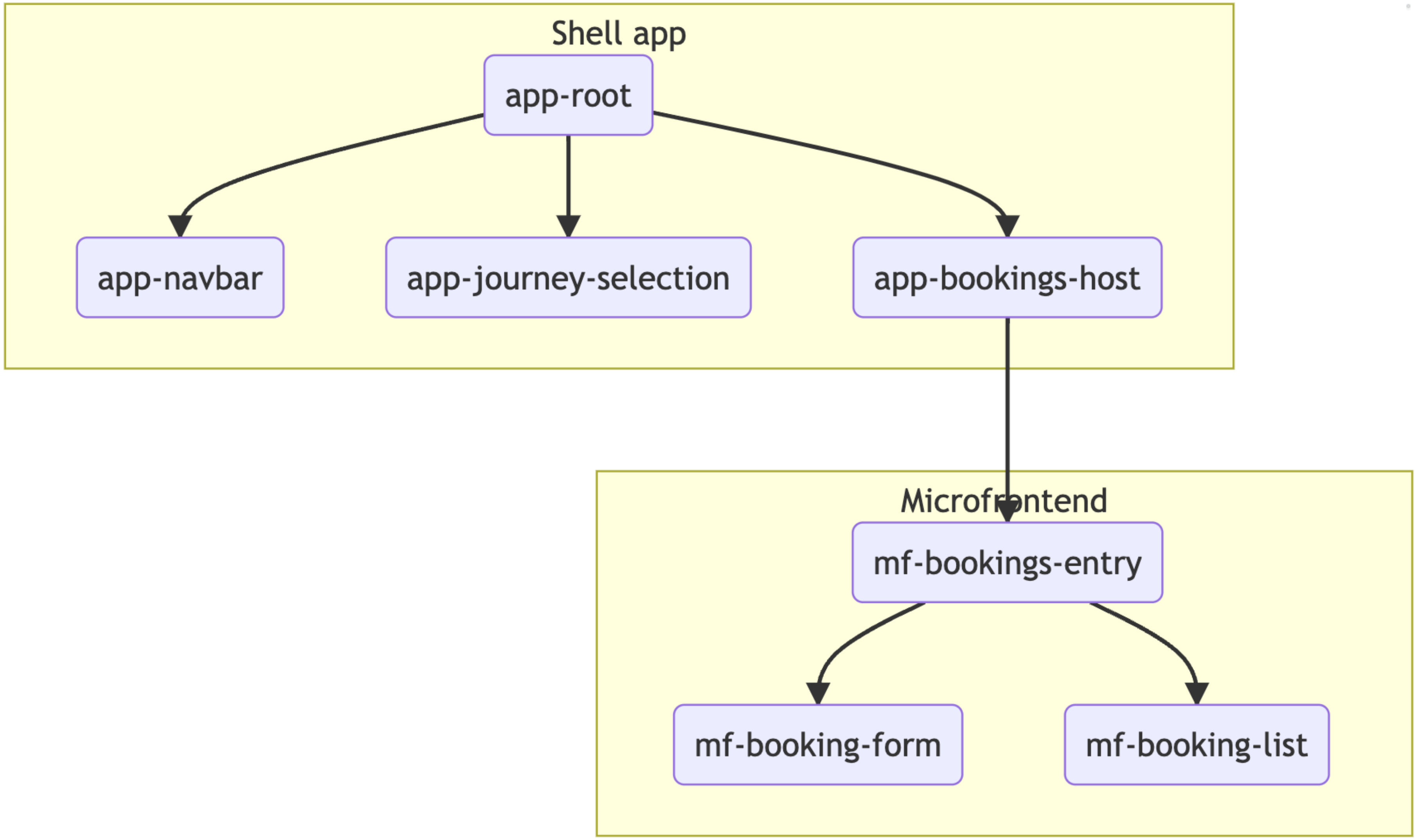
youtube.com/watch?v=ee17YczpCpU

LeanIX

Demo

ng.traveling-coderman.net/

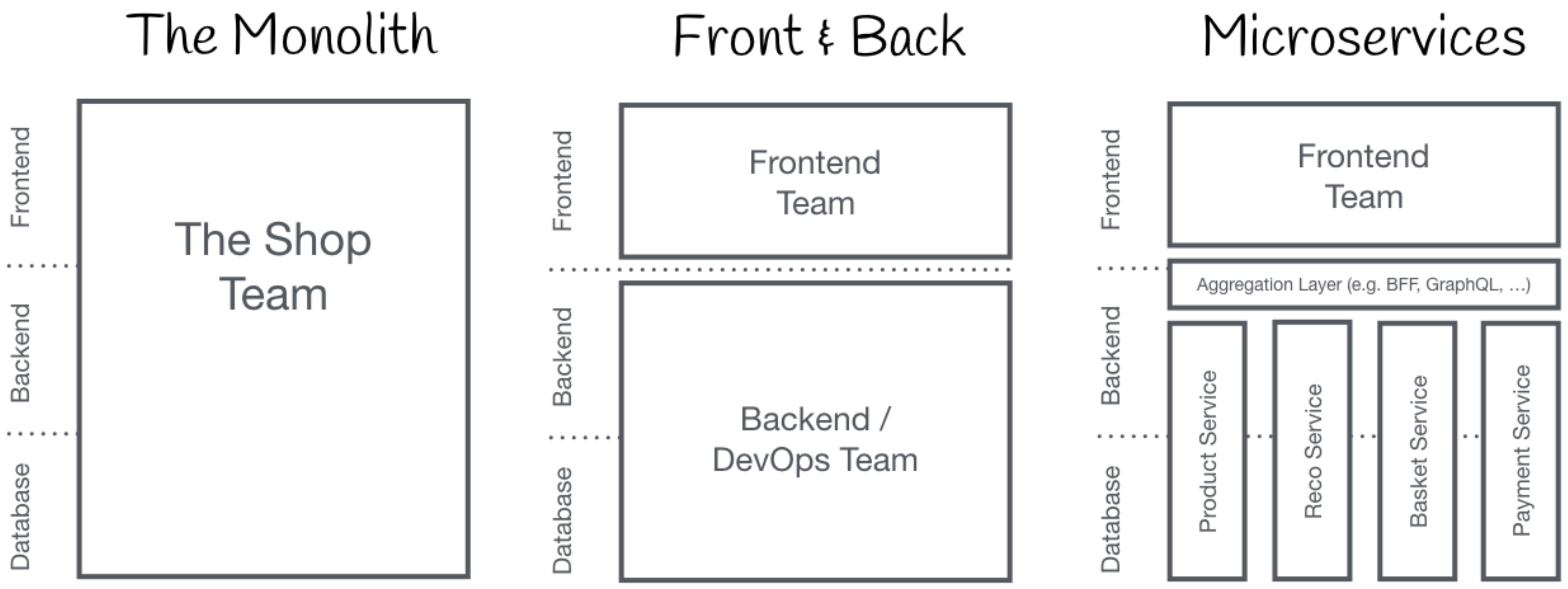
Component Hierarchy



LeanIX

What is a frontend Monolith?

Monolithic Frontends

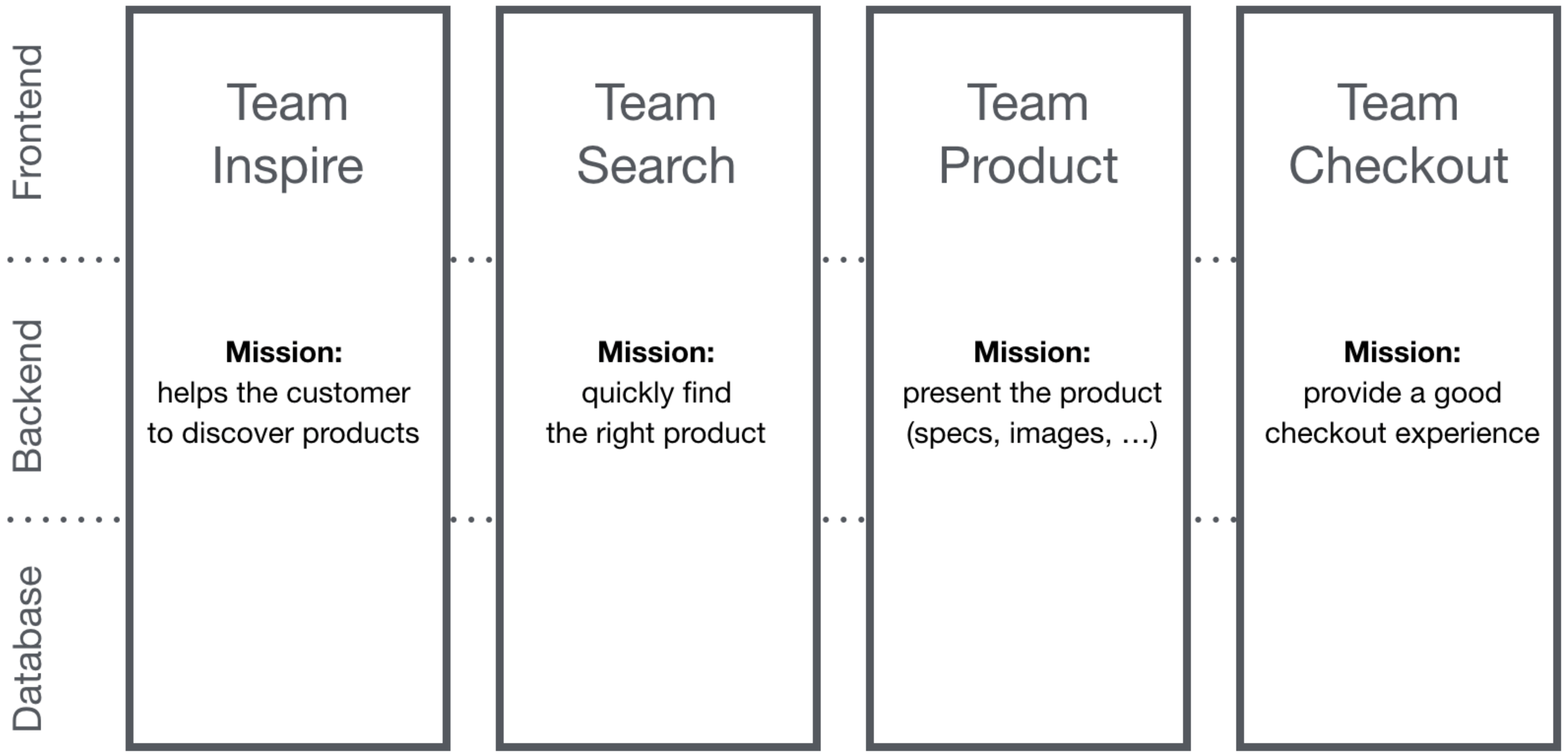


micro-frontends.org

LeanIX

Why Micro Frontends?

Micro Services throughout the whole stack

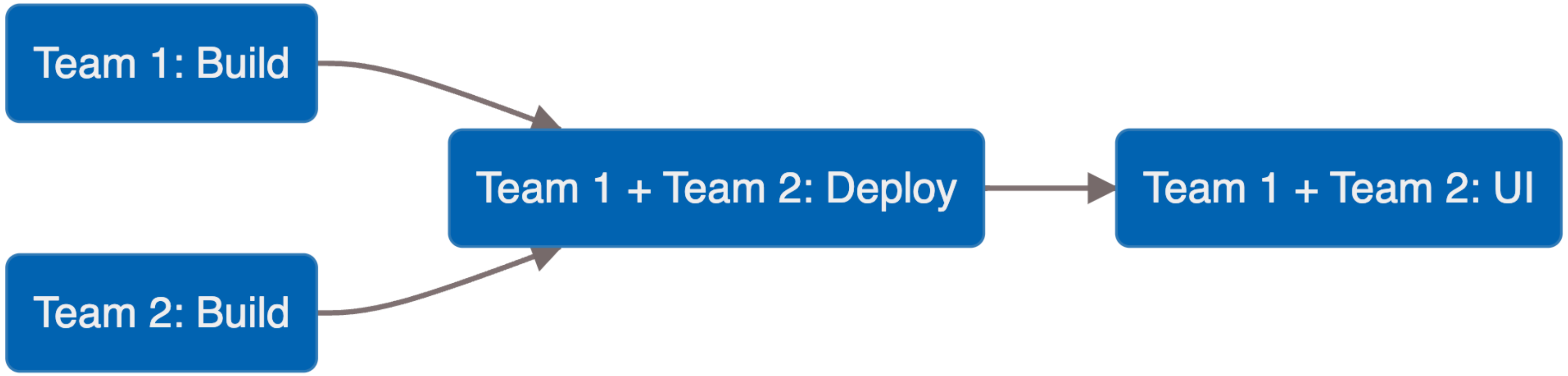


Frontend
Backend
Database
Team Inspire
Mission: helps the customer to discover products
Team Search
Mission: quickly find the right product
Team Product
Mission: present the product (specs, images, …)
Team Checkout
Mission: provide a good checkout experience

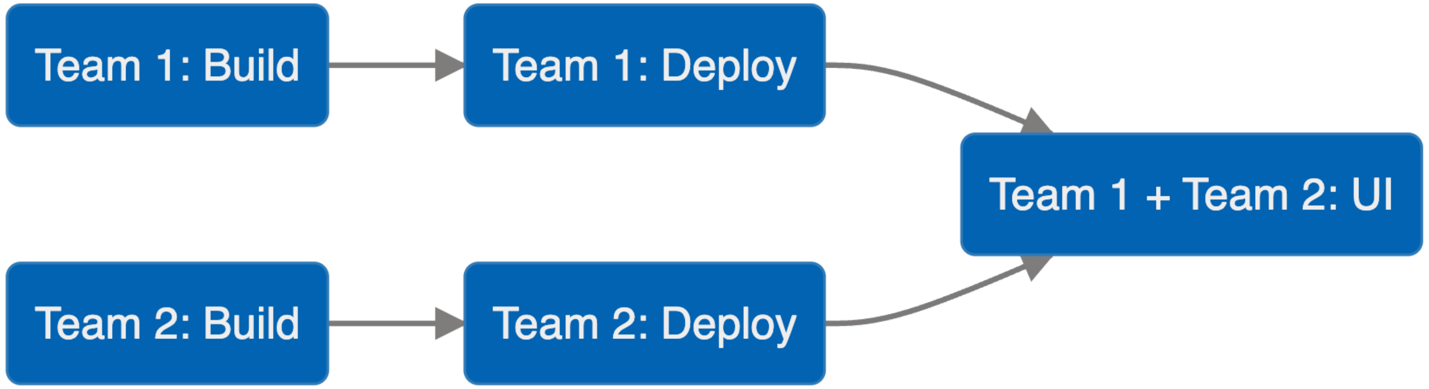
micro-frontends.org

Team Autonomy

Delivery pipeline with frontend monolith:



Delivery pipeline with micro frontends:



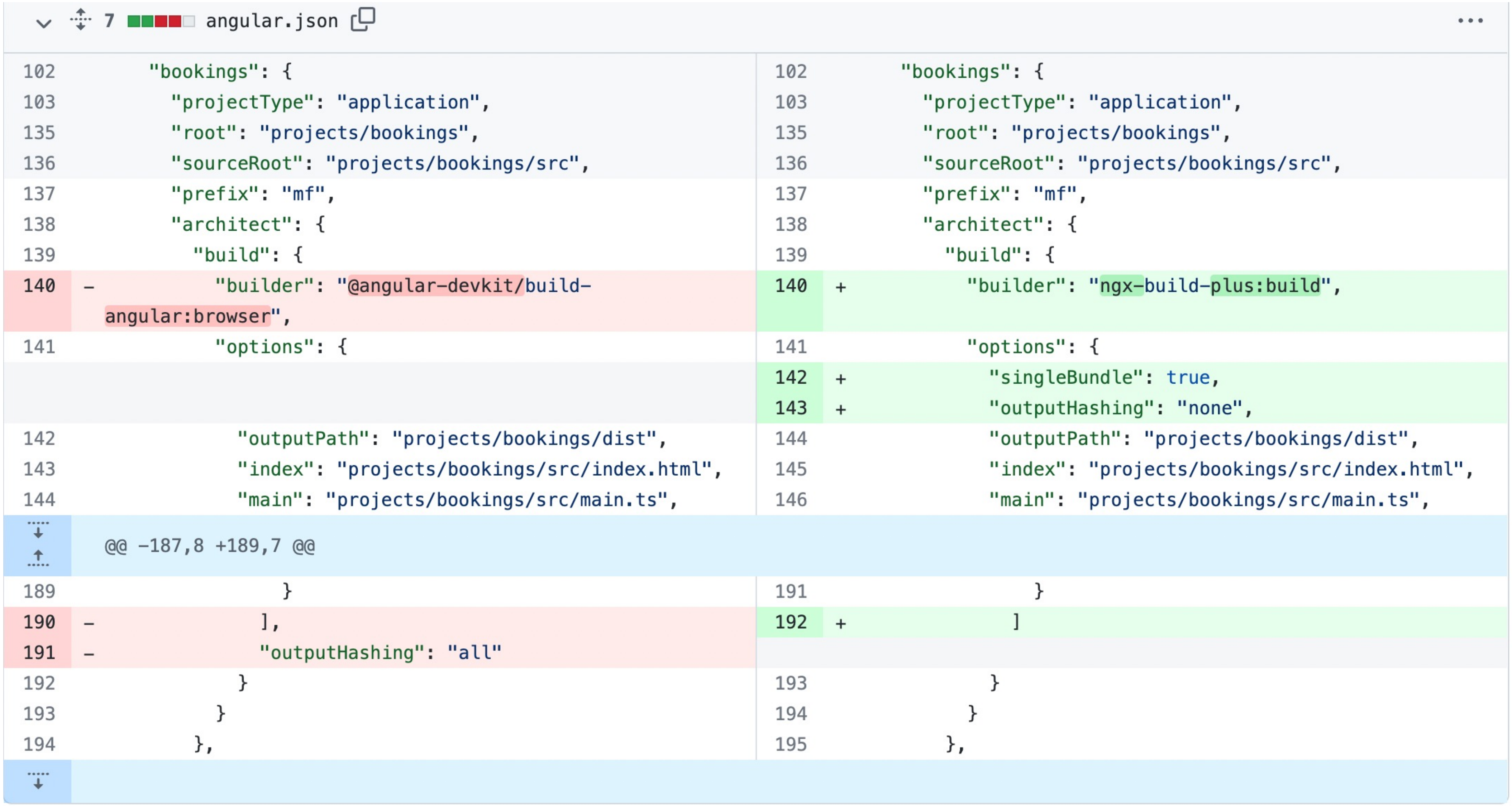
You can start using micro frontends for new routes today

- Keep serving existing routes from your monolith, build new routes and gradually migrate existing routes as micro frontends.

LeanIX

How does it work?

Bundling your micro frontend into a single main.js



```
7 angular.json
```

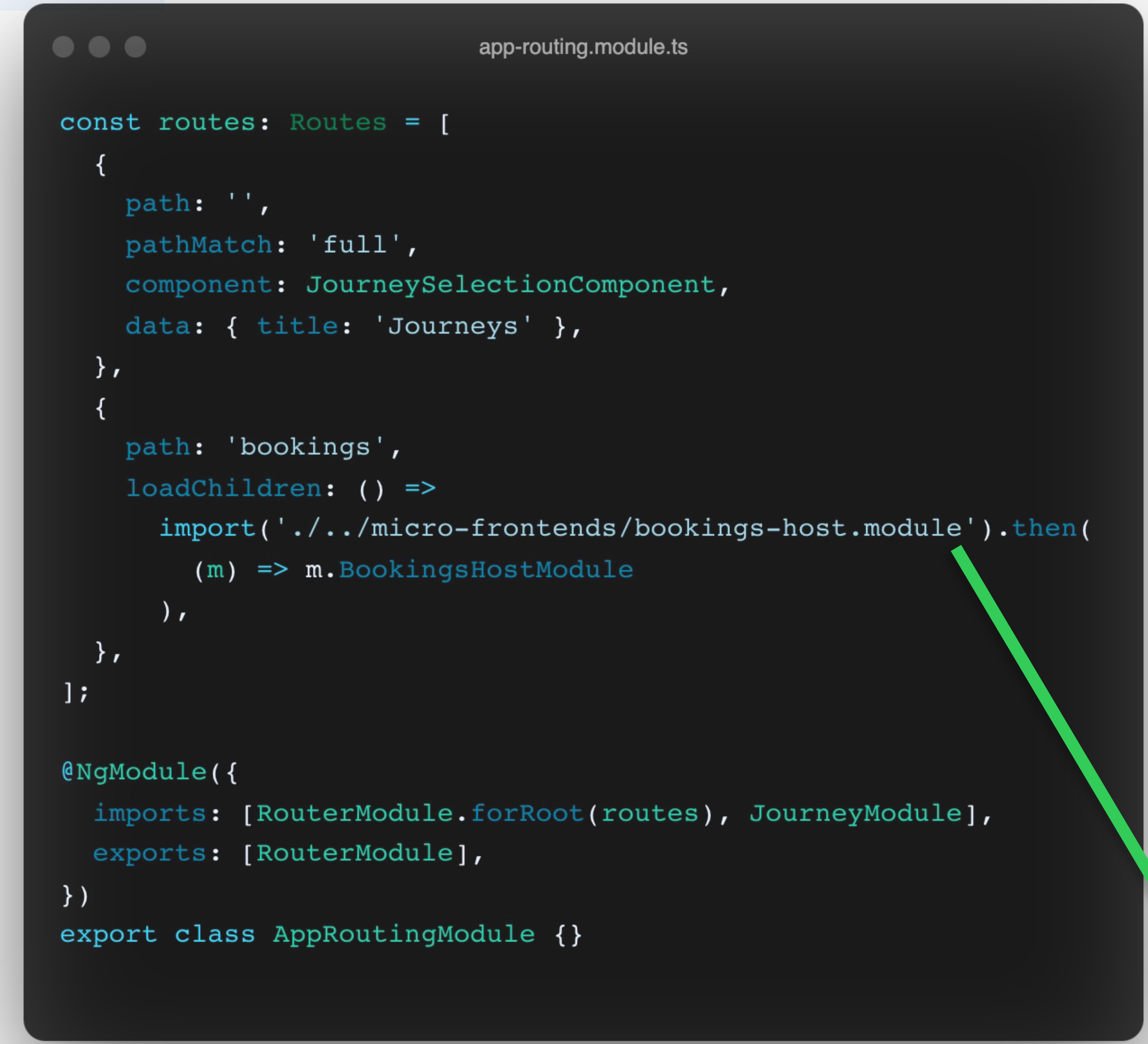
```diff
        "bookings": {
          "projectType": "application",
          "root": "projects/bookings",
          "sourceRoot": "projects/bookings/src",
          "prefix": "mf",
          "architect": {
            "build": {
-             "builder": "@angular-devkit/build-angular:browser",
+             "builder": "ngx-build-plus:build",
              "options": {
+               "singleBundle": true,
+               "outputHashing": "none",
                "outputPath": "projects/bookings/dist",
                "index": "projects/bookings/src/index.html",
                "main": "projects/bookings/src/main.ts",
@@ -187,8 +189,7 @@
                  }
-               ],
-               "outputHashing": "all"
+               ]
              }
            }
          },
```

npmjs.com/package/ngx-build-plus

LeanIX

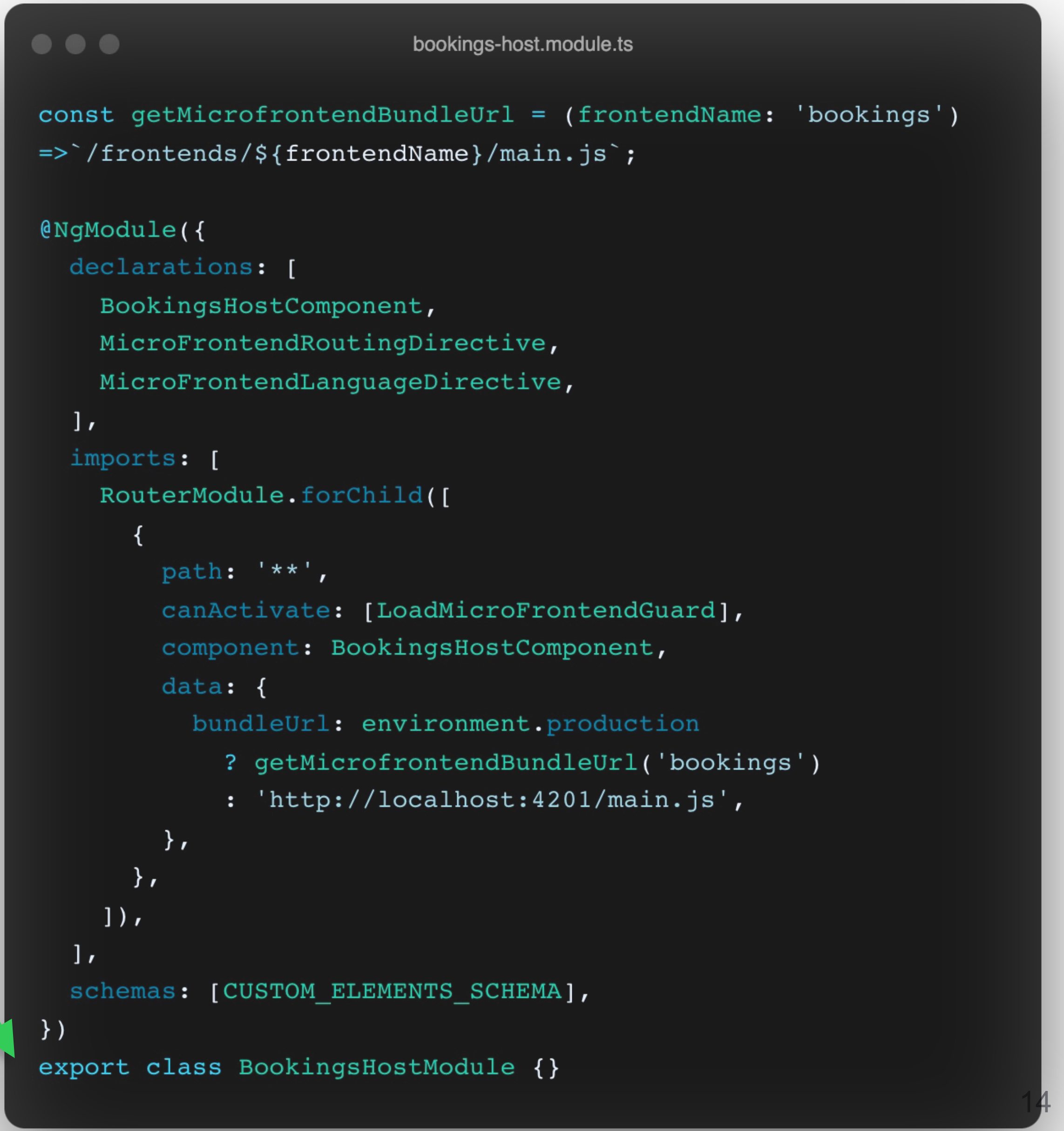
Loading via Angular Router

Router Modules



app-routing.module.ts

```typescript
const routes: Routes = [
  {
    path: '',
    pathMatch: 'full',
    component: JourneySelectionComponent,
    data: { title: 'Journeys' },
  },
  {
    path: 'bookings',
    loadChildren: () =>
      import('./../micro-frontends/bookings-host.module').then(
        (m) => m.BookingsHostModule
      ),
  },
];

@NgModule({
  imports: [RouterModule.forRoot(routes), JourneyModule],
  exports: [RouterModule],
})
export class AppRoutingModule {}
```

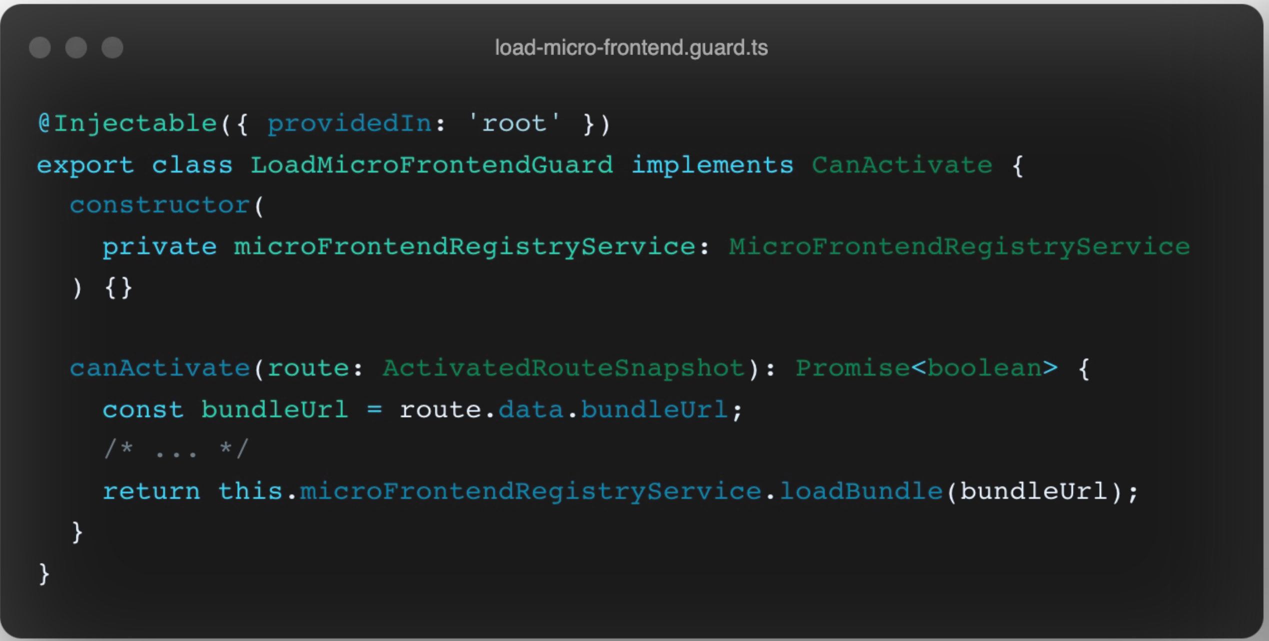


bookings-host.module.ts

```typescript
const getMicrofrontendBundleUrl = (frontendName: 'bookings')
=>`/frontends/${frontendName}/main.js`;

@NgModule({
  declarations: [
    BookingsHostComponent,
    MicroFrontendRoutingDirective,
    MicroFrontendLanguageDirective,
  ],
  imports: [
    RouterModule.forChild([
      {
        path: '**',
        canActivate: [LoadMicroFrontendGuard],
        component: BookingsHostComponent,
        data: {
          bundleUrl: environment.production
            ? getMicrofrontendBundleUrl('bookings')
            : 'http://localhost:4201/main.js',
        },
      },
    ]),
  ],
  schemas: [CUSTOM_ELEMENTS_SCHEMA],
})
export class BookingsHostModule {}
```

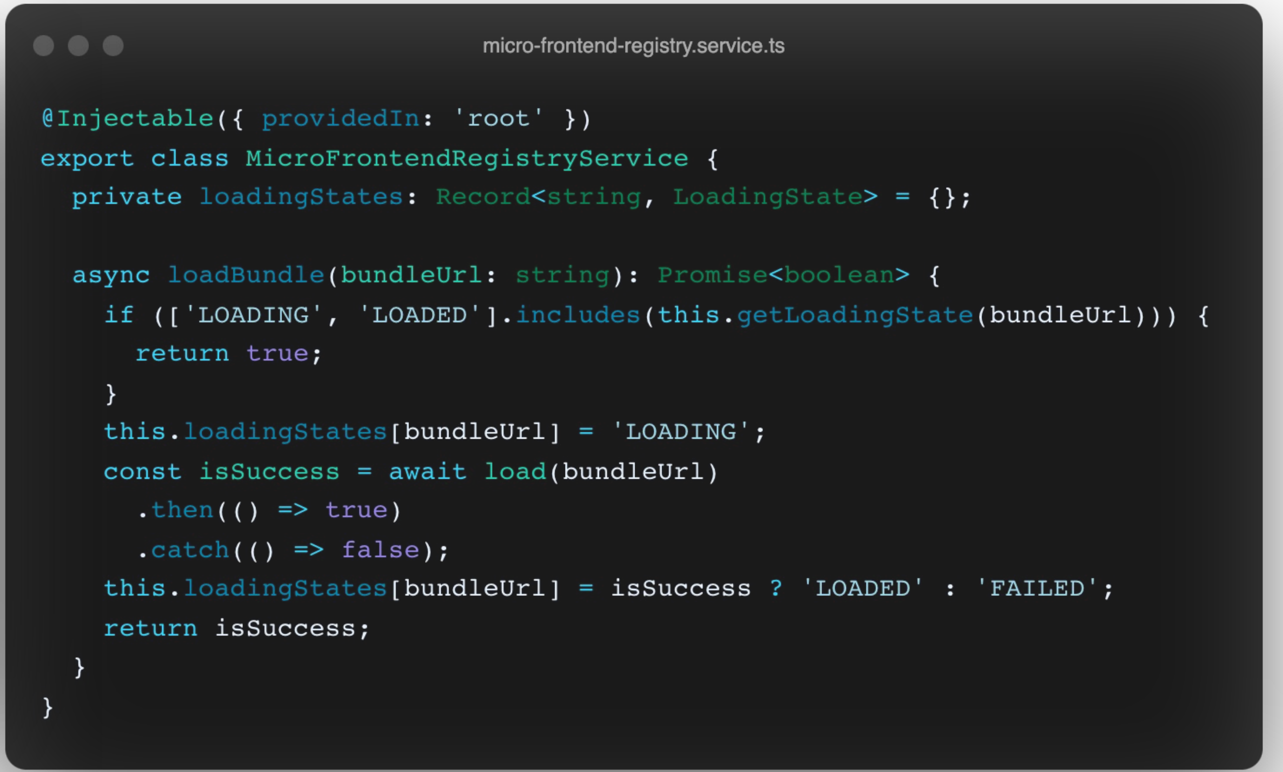
LoadMicroFrontendGuard

LeanIX



load-micro-frontend.guard.ts

```
@Injectable({ providedIn: 'root' })
export class LoadMicroFrontendGuard implements CanActivate {
  constructor(
    private microFrontendRegistryService: MicroFrontendRegistryService
  ) {}

  canActivate(route: ActivatedRouteSnapshot): Promise<boolean> {
    const bundleUrl = route.data.bundleUrl;
    /* ... */
    return this.microFrontendRegistryService.loadBundle(bundleUrl);
  }
}
```

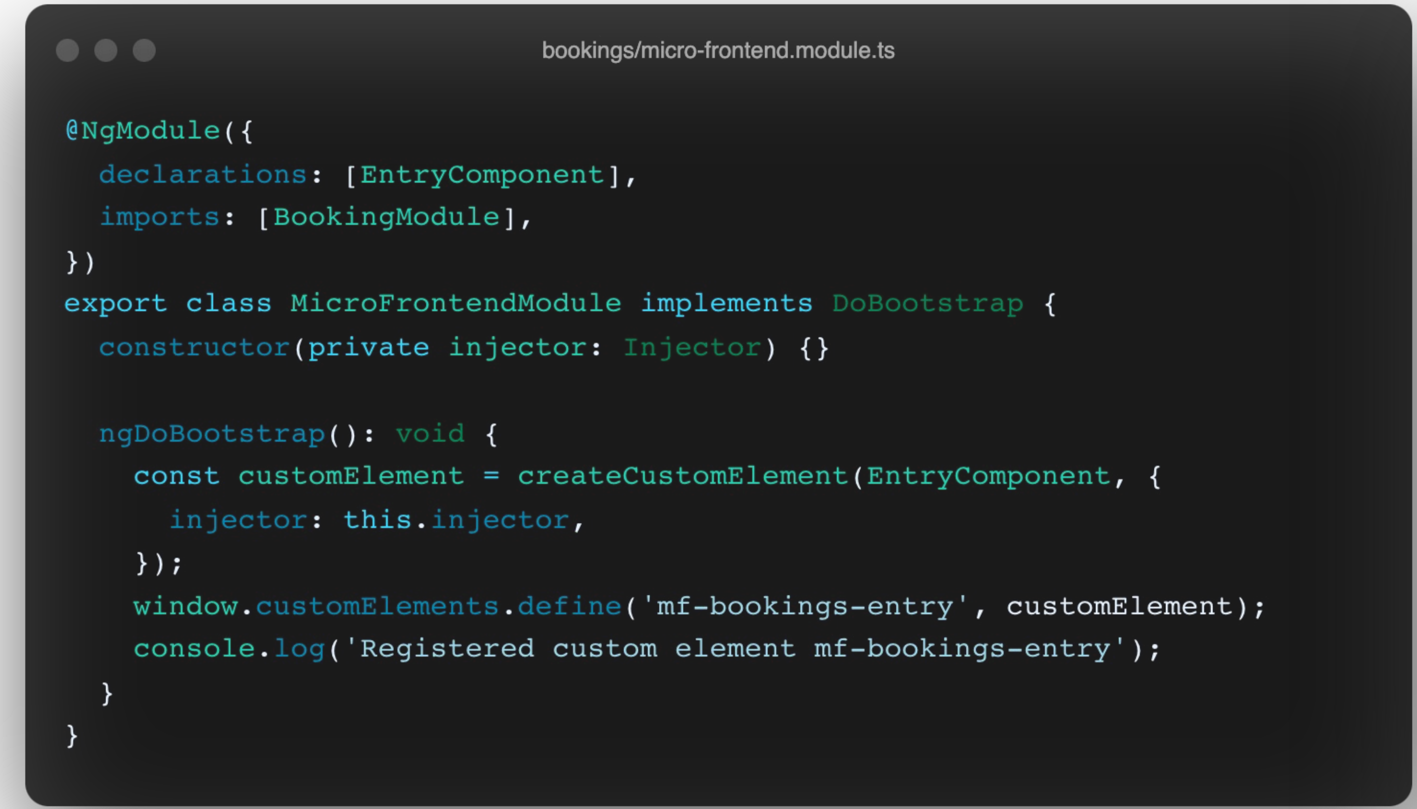


micro-frontend-registry.service.ts

```
@Injectable({ providedIn: 'root' })
export class MicroFrontendRegistryService {
  private loadingStates: Record<string, LoadingState> = {};

  async loadBundle(bundleUrl: string): Promise<boolean> {
    if (['LOADING', 'LOADED'].includes(this.getLoadingState(bundleUrl))) {
      return true;
    }
    this.loadingStates[bundleUrl] = 'LOADING';
    const isSuccess = await load(bundleUrl)
      .then(() => true)
      .catch(() => false);
    this.loadingStates[bundleUrl] = isSuccess ? 'LOADED' : 'FAILED';
    return isSuccess;
  }
}
```

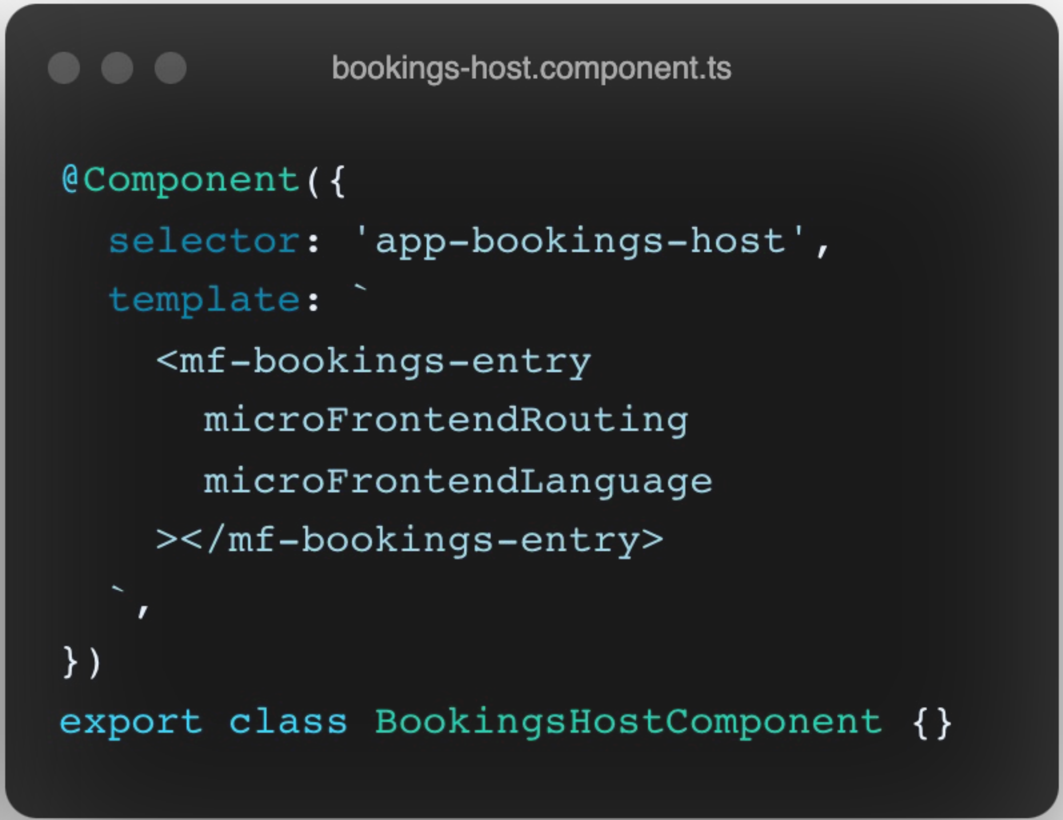
Rendering your web component Micro Frontend

/frontends/bookings/main.js
-> bootstrapModule(MicroFrontendModule)


bookings/micro-frontend.module.ts

```
@NgModule({
  declarations: [EntryComponent],
  imports: [BookingModule],
})
export class MicroFrontendModule implements DoBootstrap {
  constructor(private injector: Injector) {}

  ngDoBootstrap(): void {
    const customElement = createCustomElement(EntryComponent, {
      injector: this.injector,
    });
    window.customElements.define('mf-bookings-entry', customElement);
    console.log('Registered custom element mf-bookings-entry');
  }
}
```

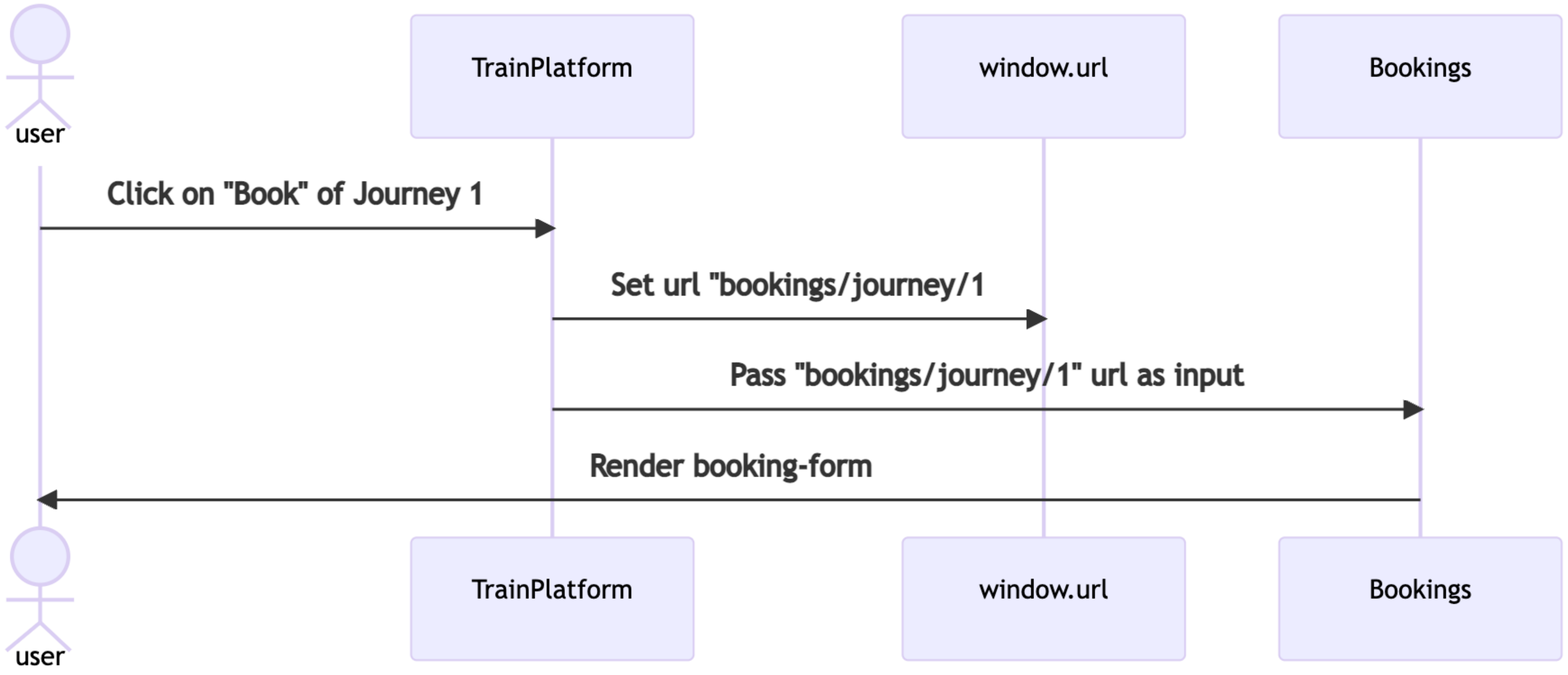
Host component in shell application

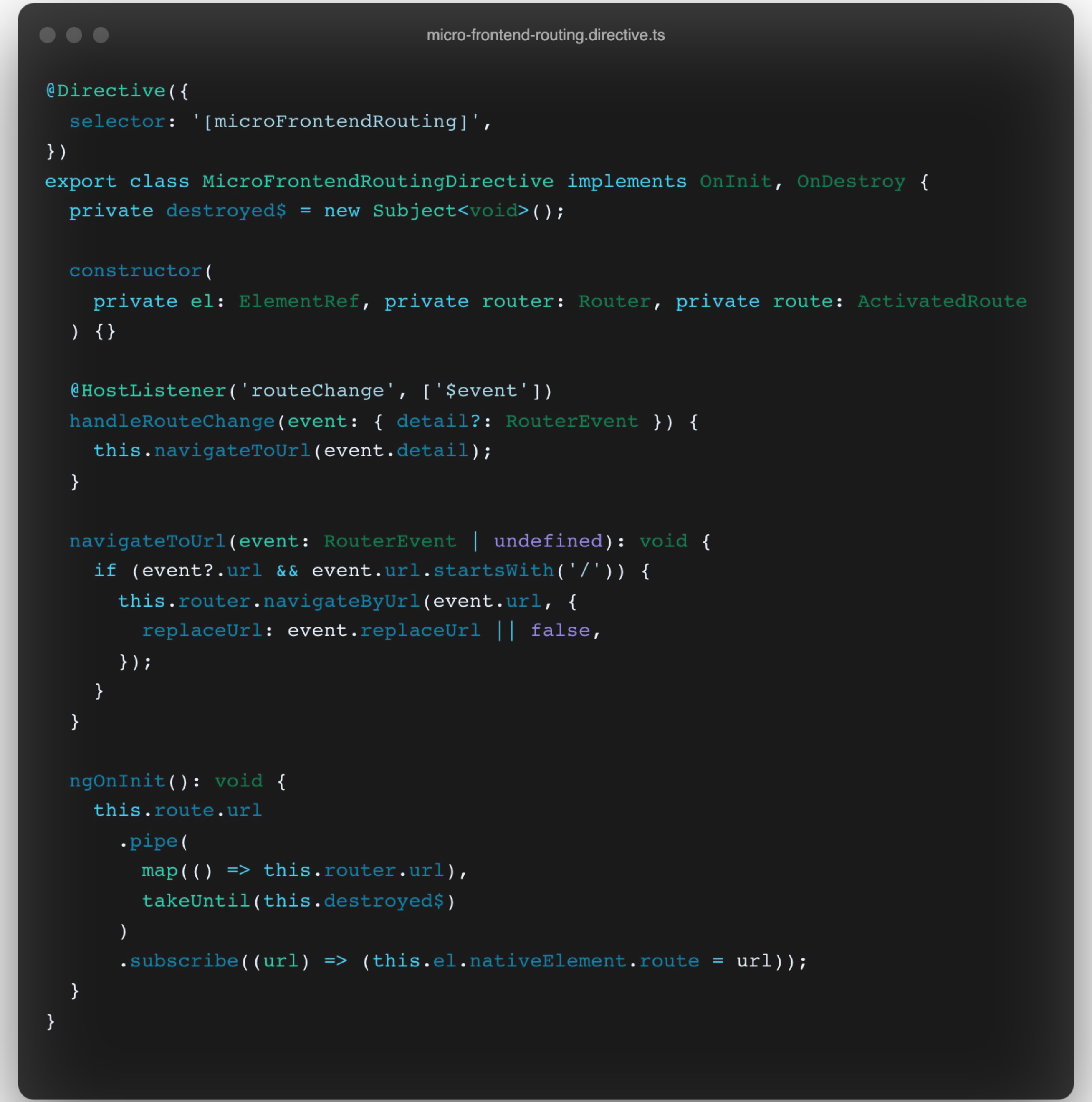

bookings-host.component.ts

```
@Component({
  selector: 'app-bookings-host',
  template: `
    <mf-bookings-entry
      microFrontendRouting
      microFrontendLanguage
    ></mf-bookings-entry>
  `,
})
export class BookingsHostComponent {}
```

microFrontendRoutingDirective

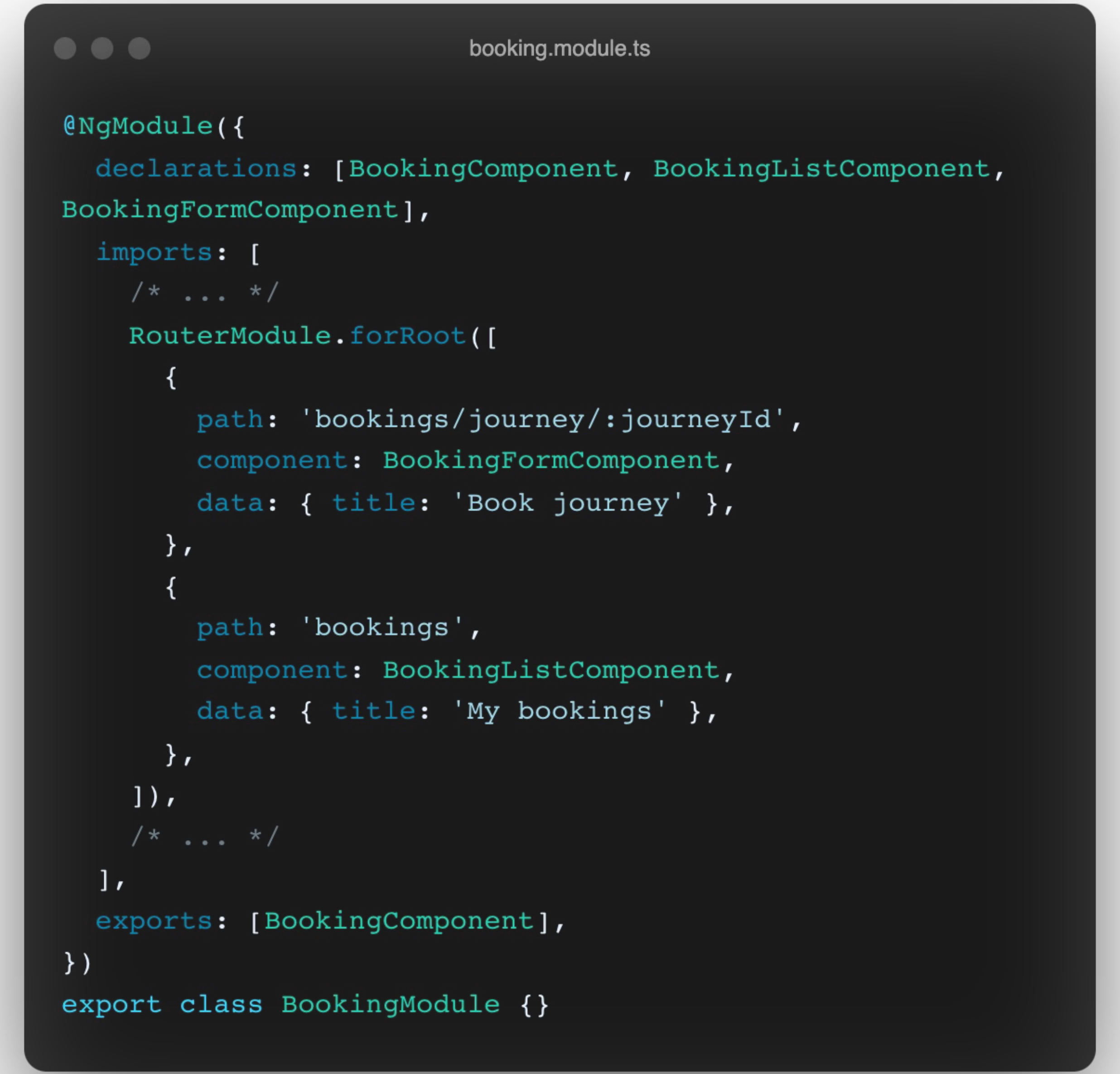
LeanIX


user
TrainPlatform
window.url
Bookings
Click on "Book" of Journey 1
Set url "bookings/journey/1
Pass "bookings/journey/1" url as input
Render booking-form
user
TrainPlatform
window.url
Bookings


micro-frontend-routing.directive.ts

```typescript
@Directive({
  selector: '[microFrontendRouting]',
})
export class MicroFrontendRoutingDirective implements OnInit, OnDestroy {
  private destroyed$ = new Subject<void>();

  constructor(
    private el: ElementRef, private router: Router, private route: ActivatedRoute
  ) {}

  @HostListener('routeChange', ['$event'])
  handleRouteChange(event: { detail?: RouterEvent }) {
    this.navigateToUrl(event.detail);
  }

  navigateToUrl(event: RouterEvent | undefined): void {
    if (event?.url && event.url.startsWith('/')) {
      this.router.navigateByUrl(event.url, {
        replaceUrl: event.replaceUrl || false,
      });
    }
  }

  ngOnInit(): void {
    this.route.url
      .pipe(
        map(() => this.router.url),
        takeUntil(this.destroyed$)
      )
      .subscribe((url) => (this.el.nativeElement.route = url));
  }
}
```

Your microfrontend can have multiple child routes


booking.module.ts

```
@NgModule({
  declarations: [BookingComponent, BookingListComponent,
BookingFormComponent],
  imports: [
    /* ... */
    RouterModule.forRoot([
      {
        path: 'bookings/journey/:journeyId',
        component: BookingFormComponent,
        data: { title: 'Book journey' },
      },
      {
        path: 'bookings',
        component: BookingListComponent,
        data: { title: 'My bookings' },
      },
    ]),
    /* ... */
  ],
  exports: [BookingComponent],
})
export class BookingModule {}
```

LeanIX

Why not iframes?

Disadvantages of iframes

LeanIX

Overlays

Modals and toast messages are only displayed within the element of the iframe itself and cannot "escape" it. This is why with iframes you would need to use some mechanisms like Window.postMessage to let the outer application know that it should display a modal or toast message with some specific content that is passed along.

Theming

The DOM of the iframe is separate from the DOM of the embedding application. This makes it harder to use CSS variables for example.

LeanIX

What about Module Federation?

Resources

LeanIX

Blogs

traveling-coderman.net/code/microfrontends/why/

xkons.de/posts/ng-micro-frontends/

Code

github.com/fboeller/microfrontends-with-angular/

Demo App

ng.traveling-coderman.net/

Video Tutorial

youtube.com/watch?v=ee17YczpCpU