# Motivation

- SPAs → single page application

- Simulate pages → routes

- URL should direct to the routed component
  - Menus & bookmarks (today → Google ☺)
  - Sharing (social platforms)
  - **The Back button!**
  - **For development**

# Contents

- Basics & Parameters

- Child Routes

- Aux Routes

- Guards (new in NG15!)

- Resolver

- Lazy Loading

# Angular Router

# Routing in Angular
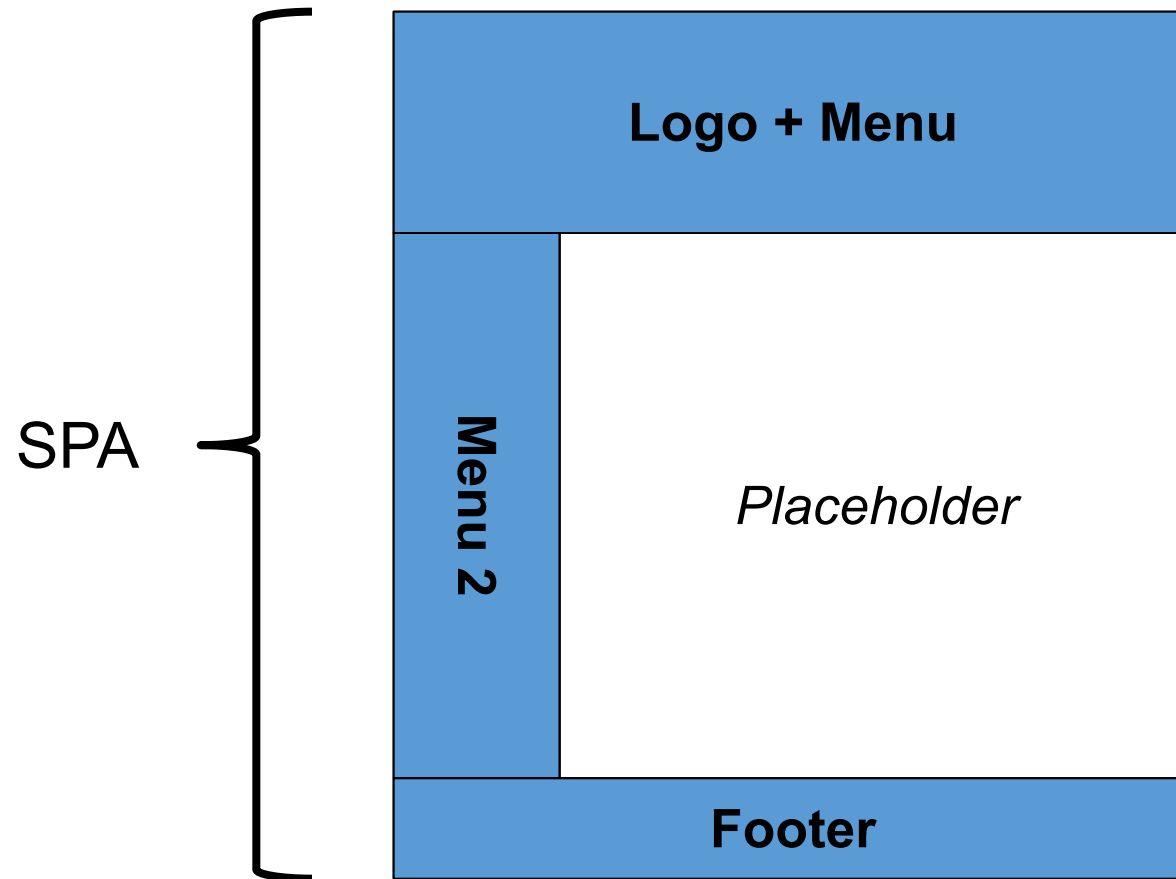
SPA

**Logo + Menu**

**Menu 2**

*Placeholder*

**Footer**

ANGULAR **ARCHITECTS** INSIDE KNOWLEDGE   SOFTWARE **ARCHITECT**

# Routing in Angular

/FlightApp**passenger**

SPA

| Logo + Menu |
|---|

| Menu 2 | **Passenger-Component** |
|---|---|

| Footer |
|---|

ANGULAR
**ARCHITECTS**
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT
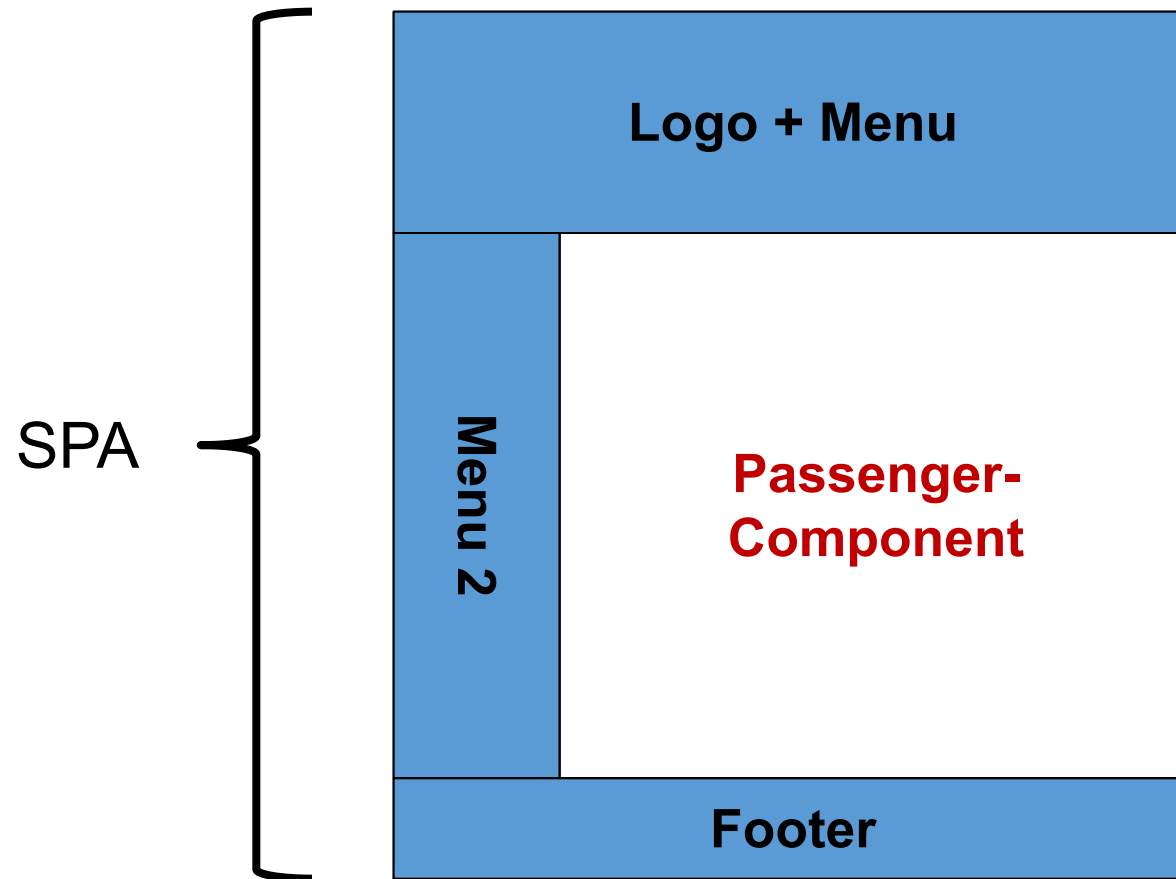
# Configuration

```
const APP_ROUTES: Routes = [
    {
        path: 'home',
        component: HomeComponent
    },
    {
        path: 'flight-search',
        component: FlightSearchComponent
    },
    {
        path: '**',
        redirectTo: 'home'
    }
]
```

# Configuration

```typescript
// app.module.ts
@NgModule({
    imports: [
        BrowserModule,
        HttpModule,
        FormsModule,
        RouterModule.forRoot(ROUTE_CONFIG)
    ],
    […]
})
export class AppModule {
}
```

**For Root-Module**

**For Feature-Module: forChild**

# AppComponent

```html
<a routerLink="/home">Home</a>
<a [routerLink]="'/flight-search'">Flight Search</a>

<div>
    <router-outlet></router-outlet>
</div>
```

# Parameters

- passenger

- passenger/7

- passenger/7/flights

- passenger/flights

- ~~passenger/7?details=true&page=7~~

- passenger/7;details=true;page=7

- passenger/7;details=true;page=7/flights

# Parameters

```
const APP_ROUTES: Routes = [

    […]
    {
        path: 'flight-search',
        component: FlightSearchComponent
    },
    {

        path: 'flight-edit/:id',
        component: FlightEditComponent
    }
}
```

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Reading Parameters

```
export class FlightEditComponent {

    id: string;

    constructor(
        private route: ActivatedRoute) {

        route.params.subscribe(
            params => {
                this.id = params['id'];
                […]
            }
        );
    }
    […]
}
```

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Reading Parameters

```typescript
export class FlightEditComponent {

    id: string;

    constructor(
        private route: ActivatedRoute) {

        route.paramMap.subscribe(
            paramMap => {
                this.id = paramMap.get('id');
                [...]
            }
        );
    }
    [...]
}
```

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Reading Parameters

```
export class FlightEditComponent {

    id: number;

    constructor(
        private route: ActivatedRoute) {

        route.paramMap.subscribe(
            paramMap => {
                this.id = +paramMap.get('id'); // okay
                this.id = Number(paramMap.get('id')); // better
                […]
            }
        );
    }
    […]
}
```

# Links for Routes with Parameters

```html
<a [routerLink]="['/flight-edit', flight.id, { showDetails: "true" }]">Edit</a>
```
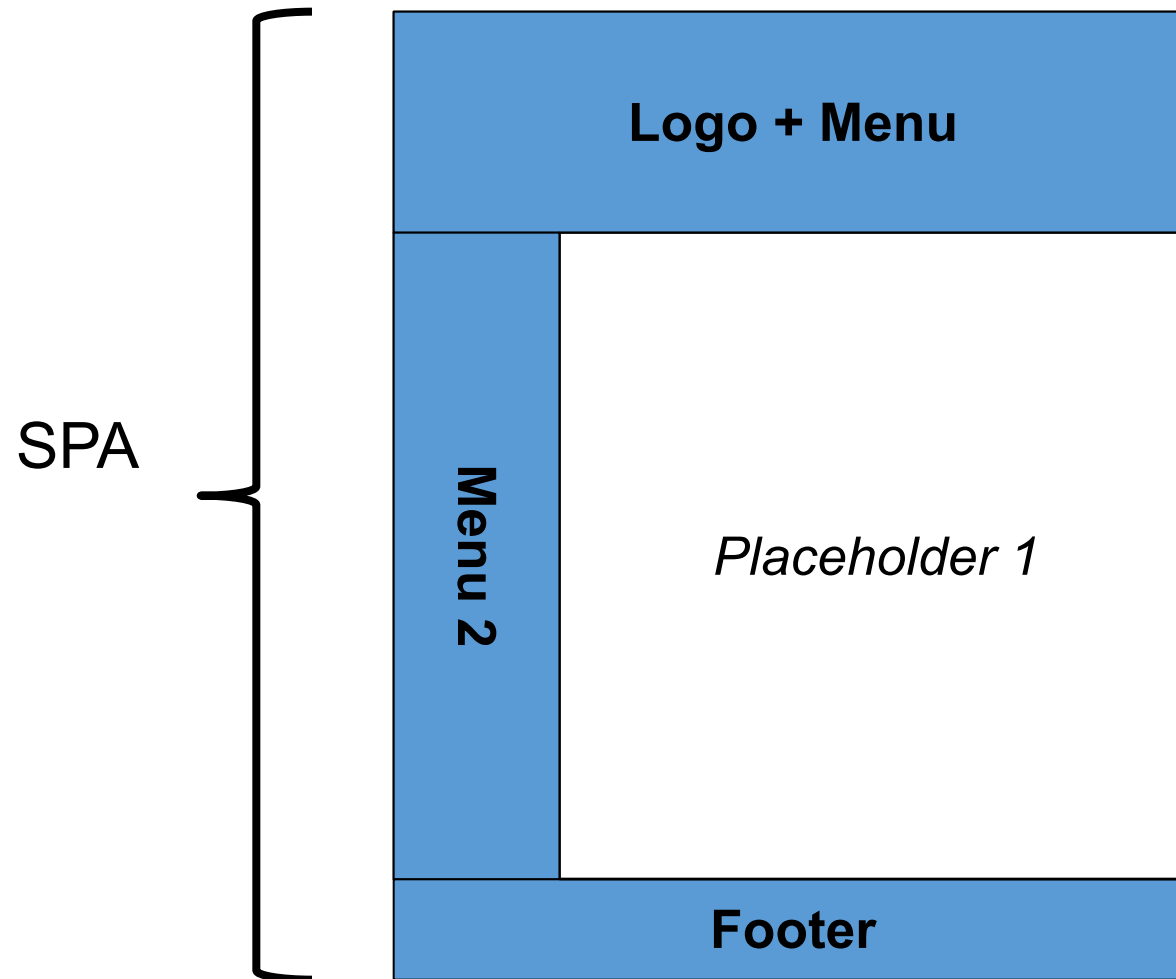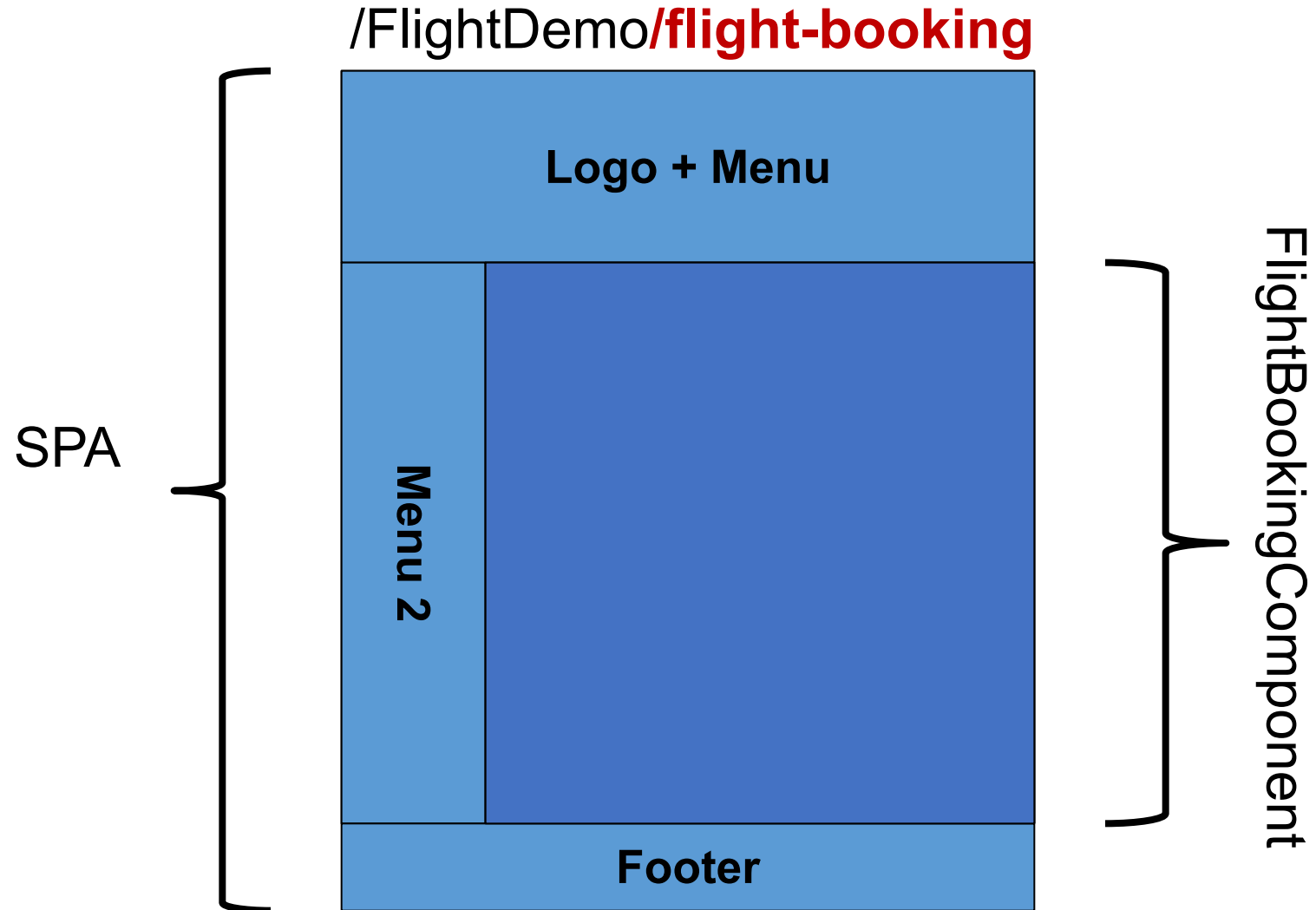
# DEMO

# Hierarchical Routing

# Hierarchical Routing



**Logo + Menu**

**Menu 2**

*Placeholder 1*

**Footer**

SPA

ANGULAR **ARCHITECTS** INSIDE KNOWLEDGE    SOFTWARE ARCHITECT

# Hierarchical Routing



/FlightDemo**flight-booking**

Logo + Menu

Menu 2

Footer

SPA

FlightBookingComponent

# Hierarchical Routing

/FlightDemo**flight-booking**

# Hierarchical Routing

/FlightDemo**flight-booking/passenger**
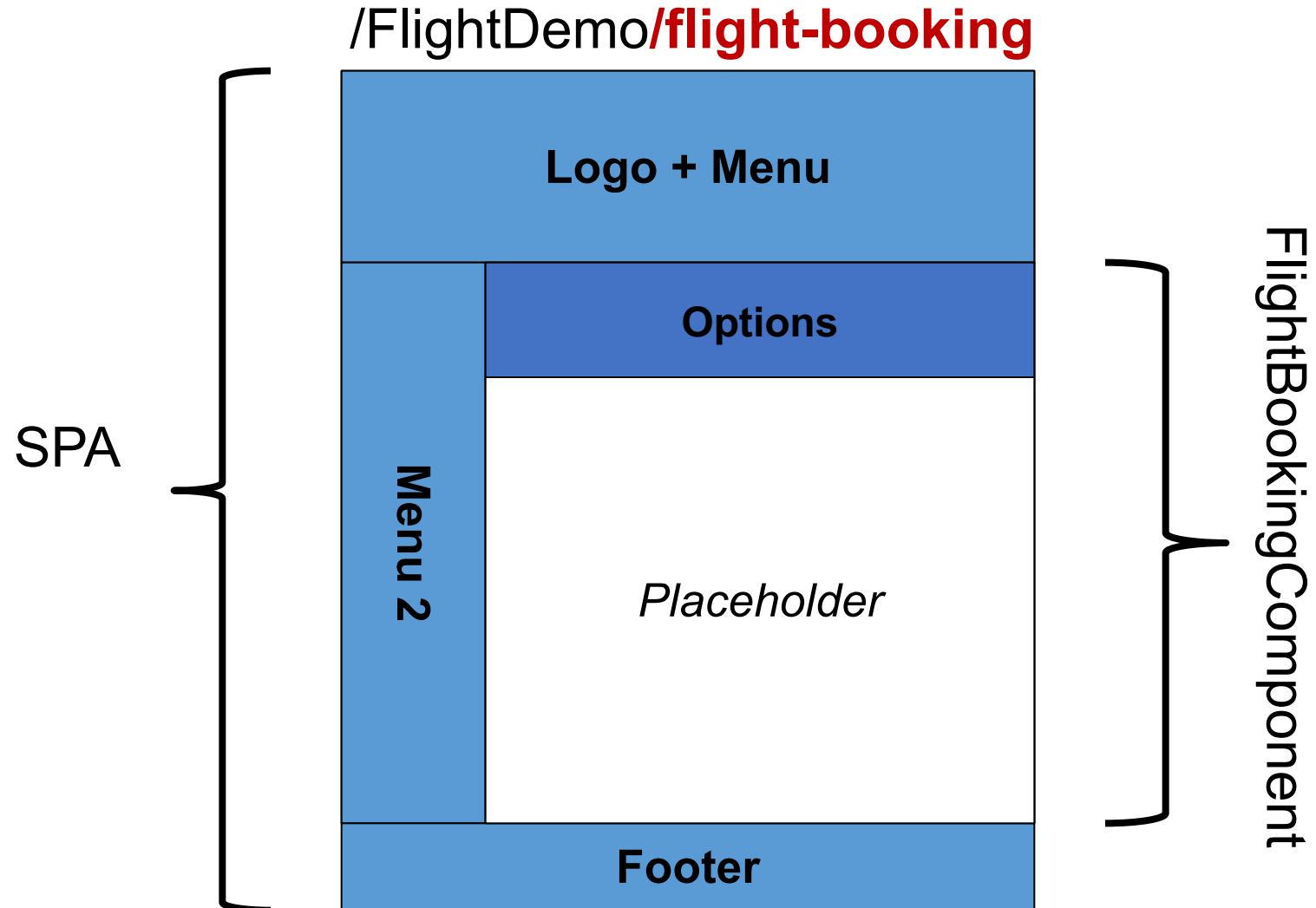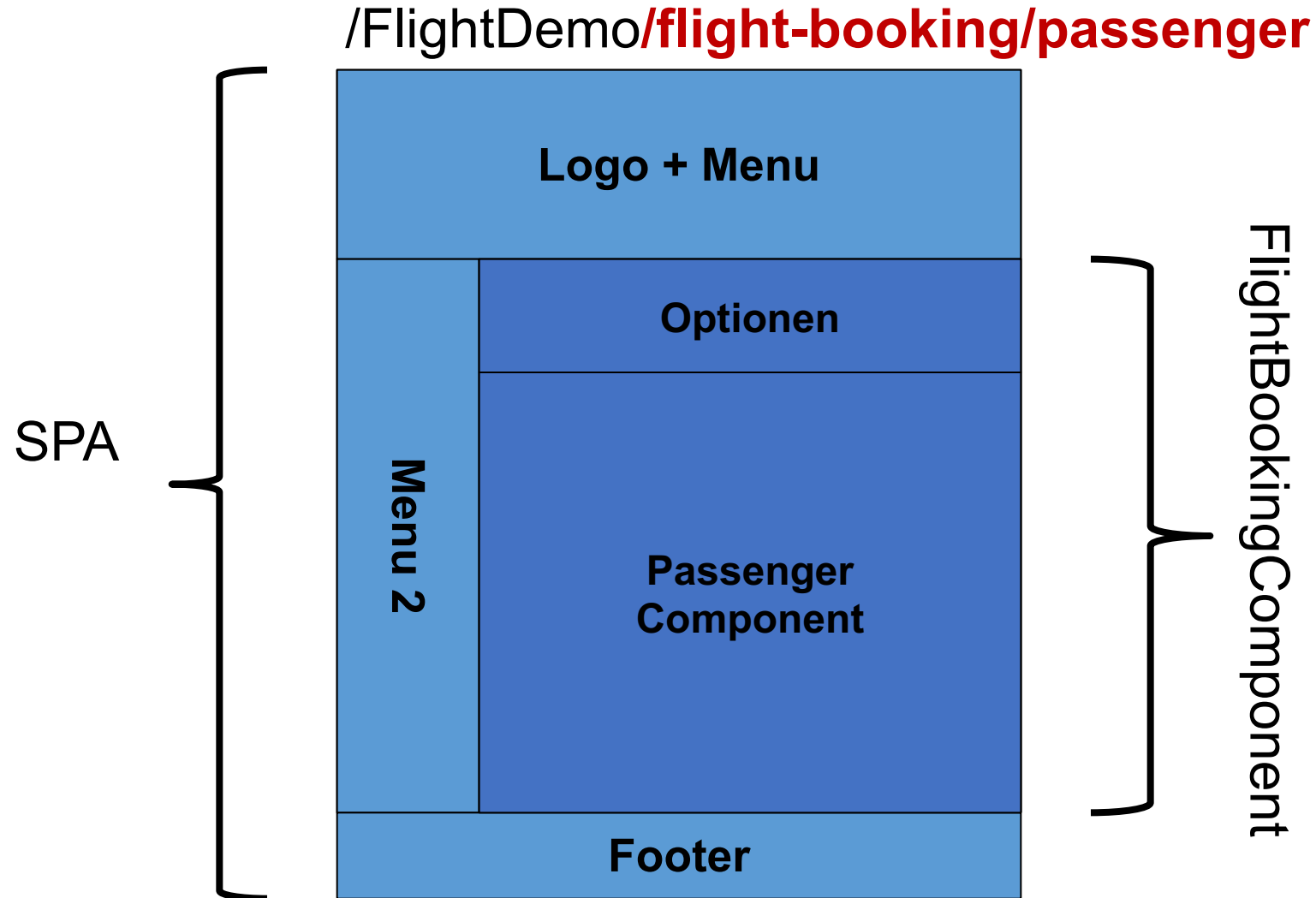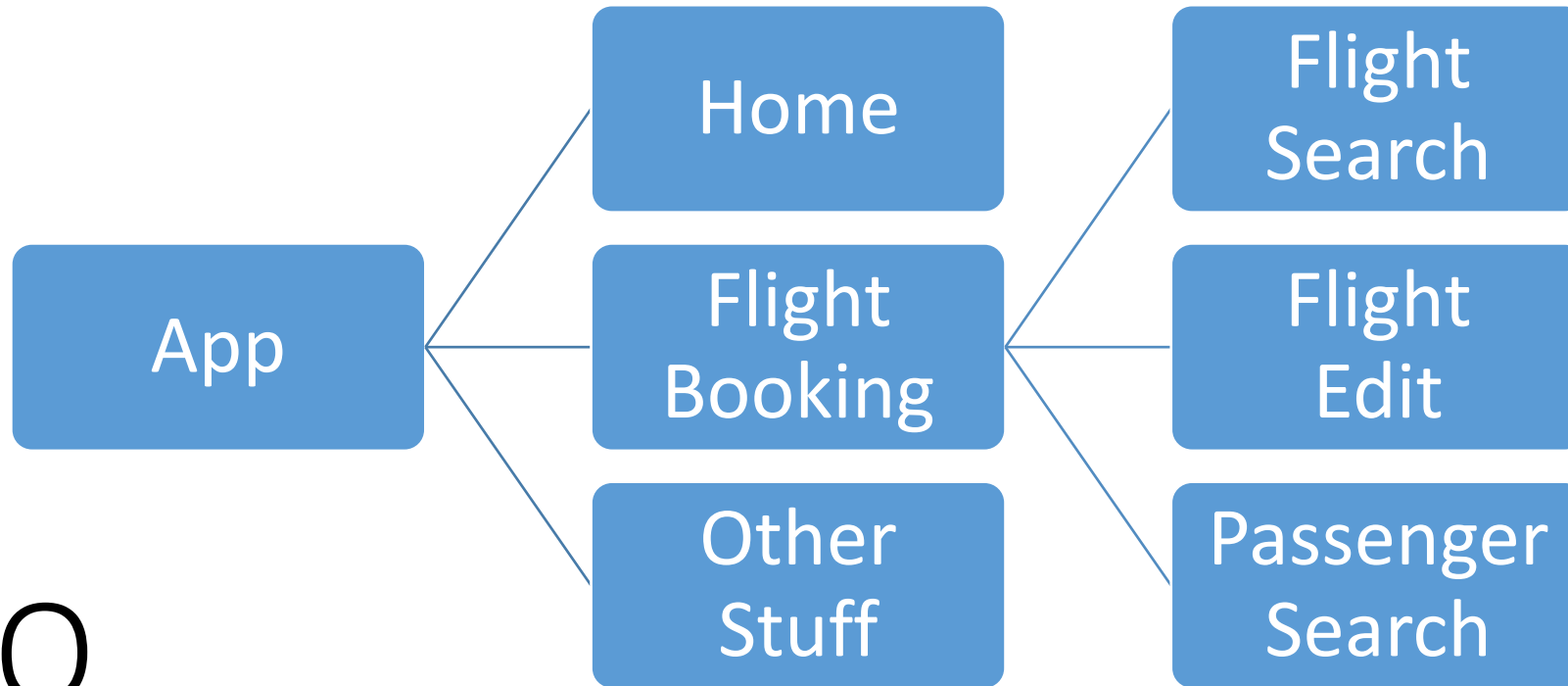
# Configuration

```
const APP_ROUTES: Routes = [
    {
        path: '',
        component: HomeComponent
    },
    {

        path: 'flight-booking',
        component: FlightBookingComponent,
        children: [
            {
                path: 'flight-search',
                component: FlightSearchComponent
            },
            […]
        ]
    }
];
```
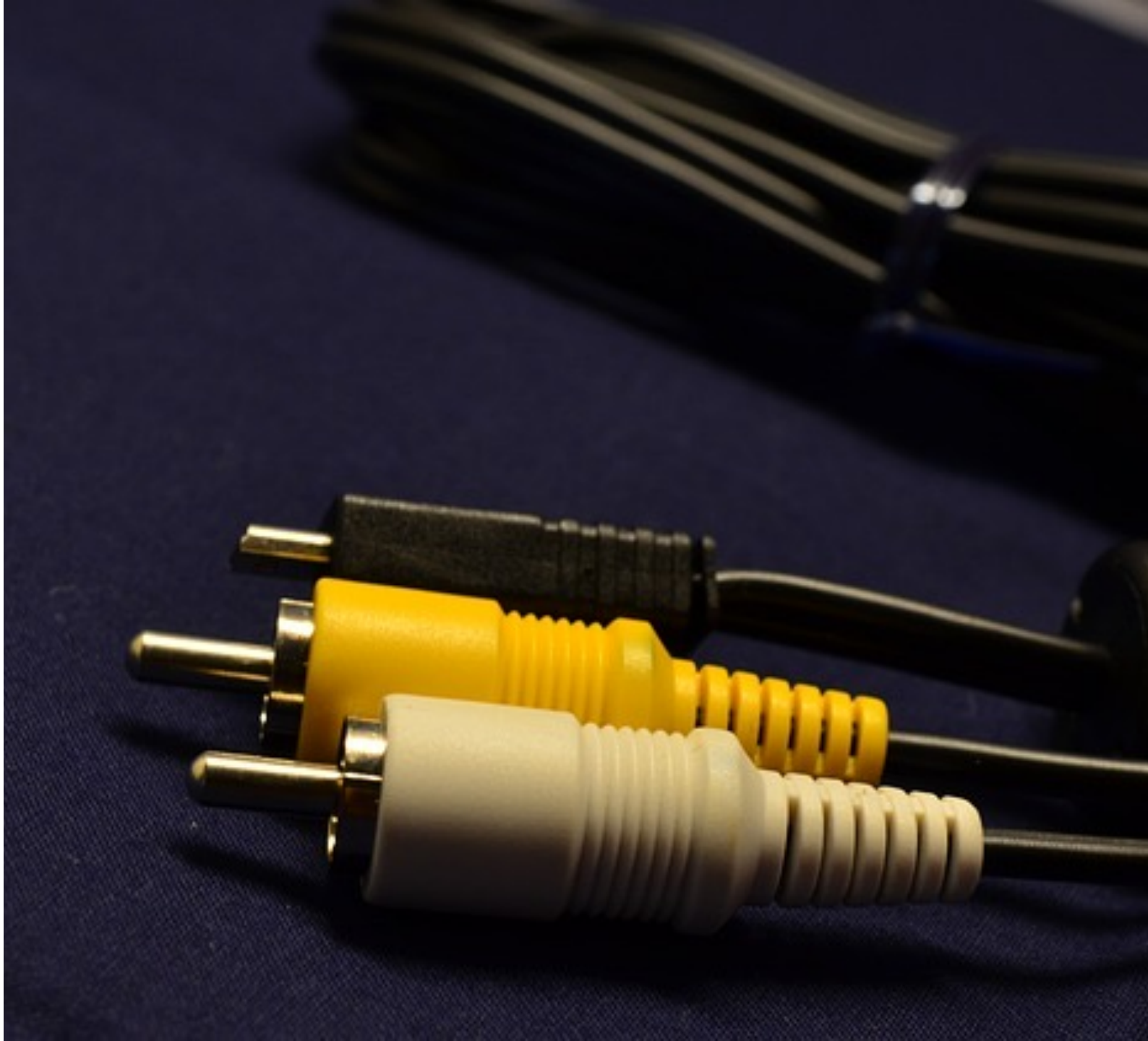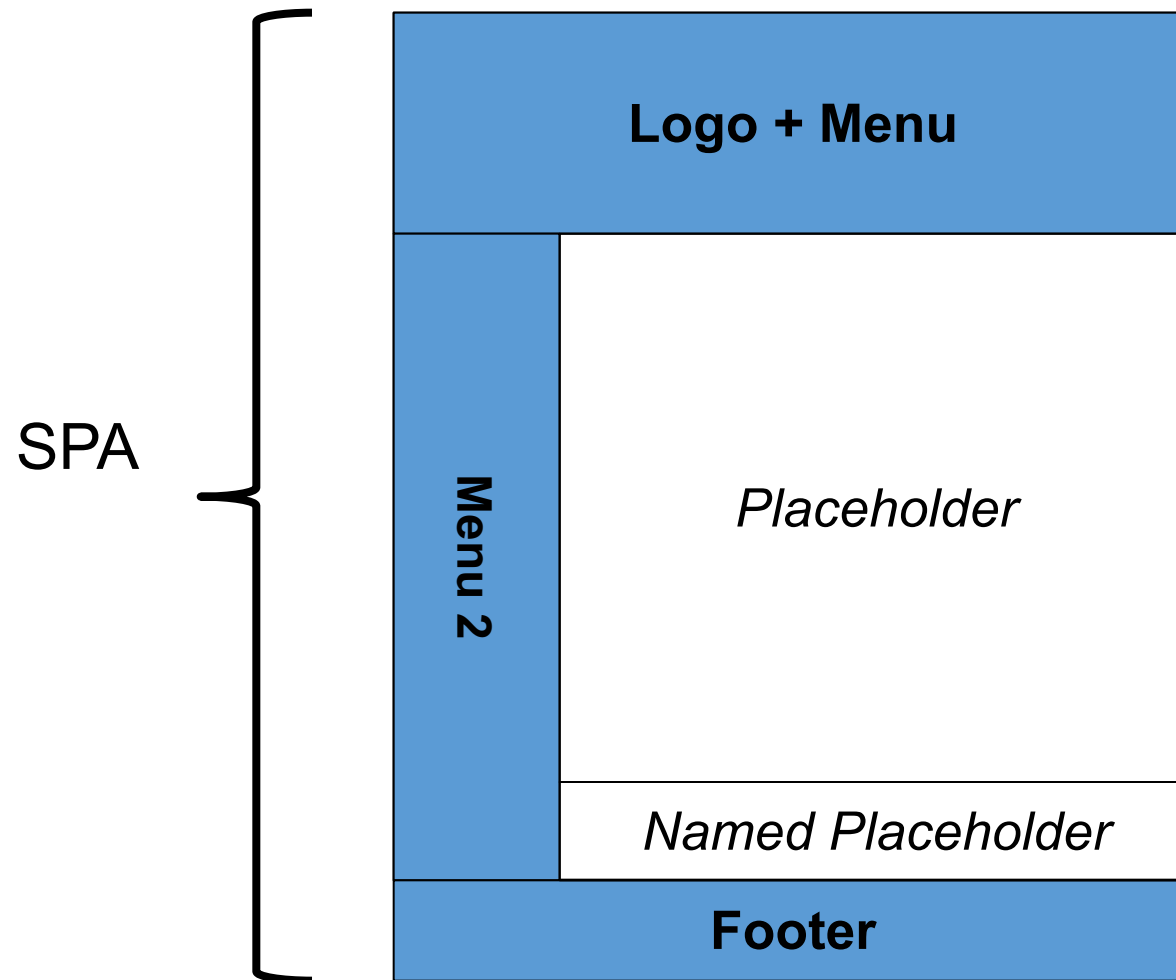
ANGULAR
**ARCHITECTS**
INSIDE KNOWLEDGE

SOFTWARE
**ARCHITECT**

DEMO

App

Home

Flight Booking

Other Stuff

Flight Search

Flight Edit

Passenger Search

# Aux Routes

# Aux-Routes

SPA

**Logo + Menu**

**Menu 2**

*Placeholder*

*Named Placeholder*

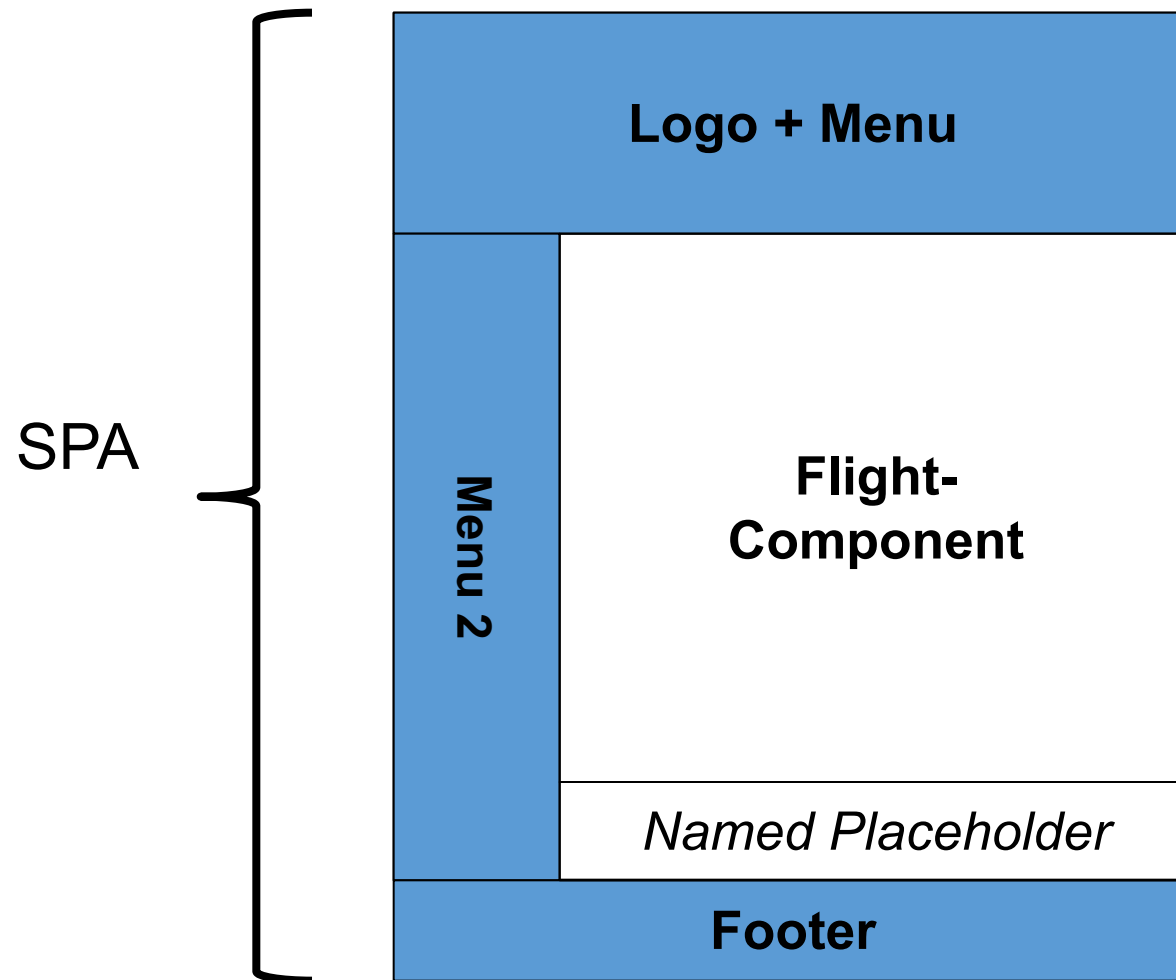**Footer**

# Aux-Routes

/FlightApp**/flights**

# Aux-Routes

/FlightApp**flights(aux:info)**



SPA

| Logo + Menu |
| Menu 2 | Flight-Component |
| | Info-Component |
| Footer |
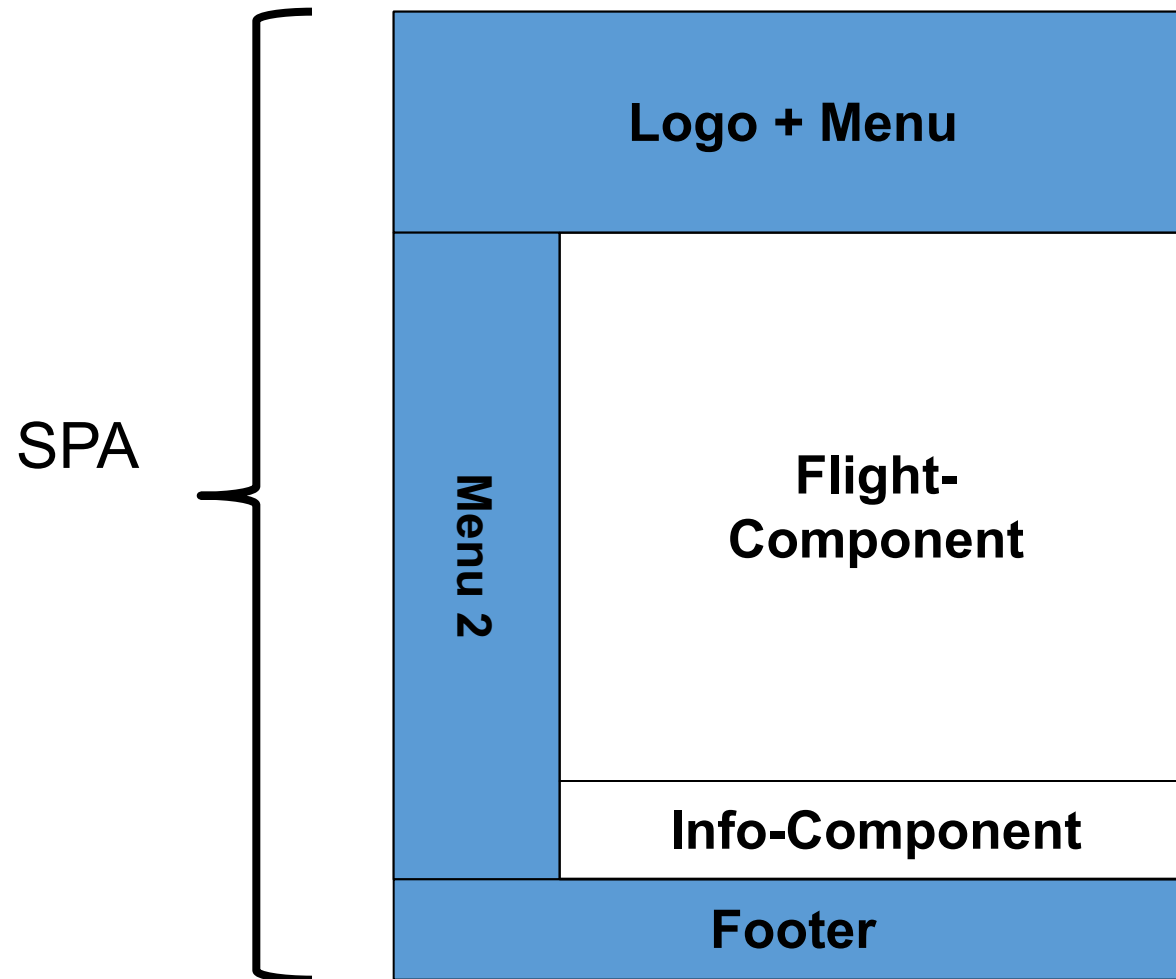
# Aux-Routes

/FlightApp**flights(aux:info/modal)**
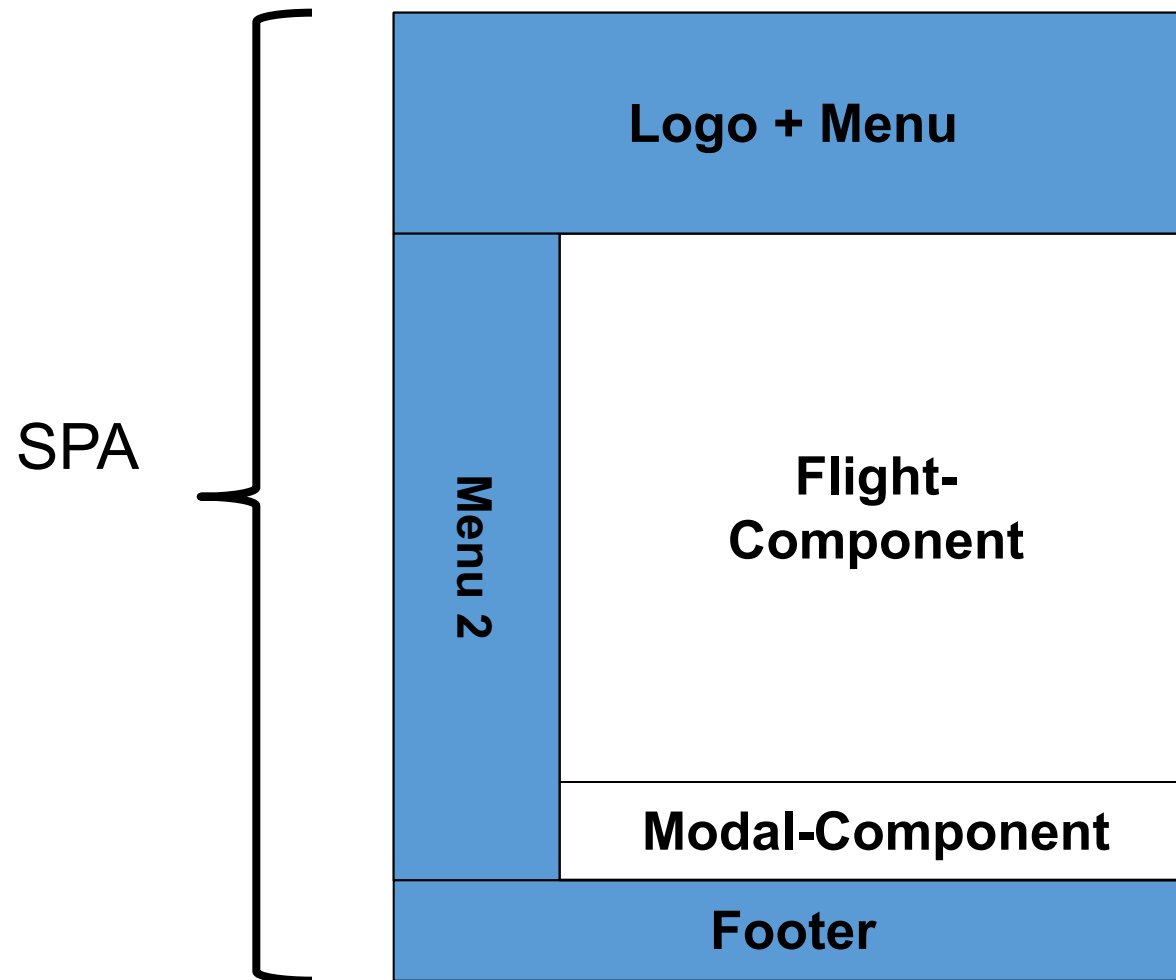
# Aux-Routes

/FlightApp**flights(aux:info/modal)/edit/17**



SPA

Logo + Menu

Menu 2

Flight-Edit-Component

Modal-Component

Footer
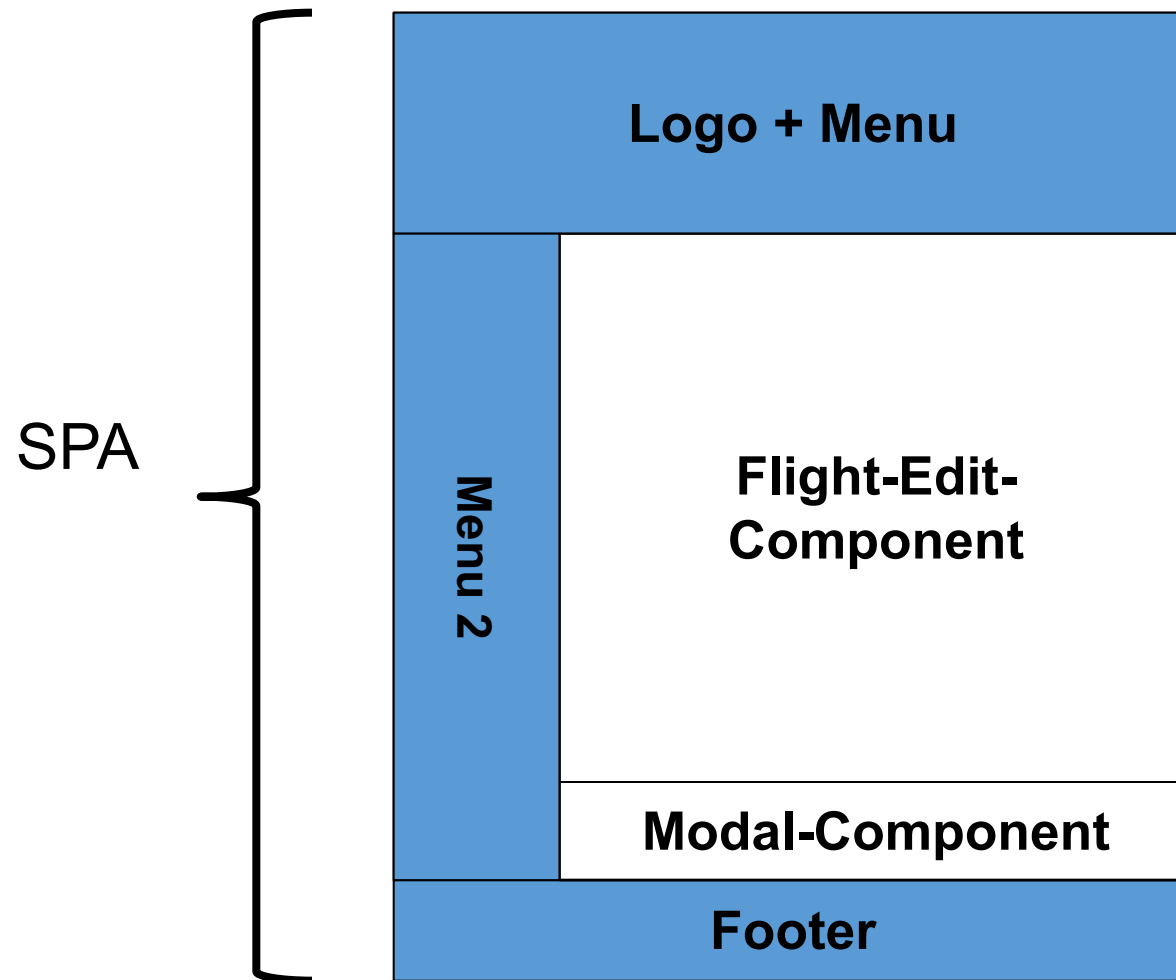
# Use Cases

- Partly autonomous parts of an application
- „Norton Commander Style"
- (CSS-based) Popups and Modals

# Define Outlets

**Default Name: primary**

```html
<router-outlet></router-outlet>

<hr>

<router-outlet name="aux"></router-outlet>
```

# Configuration

```typescript
export const ROUTE_CONFIG: Routes = [
    {
        path: 'home',
        component: HomeComponent
    },
    {
        path: 'info',
        component: InfoComponent,
        outlet: 'aux'
    },
    {
        path: 'dashboard',
        component: DashboardComponent,
        outlet: 'aux'
    }
]
```

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Activating Aux-Routes

```html
<a [routerLink]="[{outlets: { aux: 'info' }}]">
    Activate Info
</a>

<a [routerLink]="[{outlets: { aux: null }}]">
    Deactivate Info
</a>
```

# Activating Several Aux Routes at Once

```html
<a [routerLink]="[{outlets: {
                aux: 'basket',
                primary: 'flight-booking/flight-search' }}]"> … </a>


<a [routerLink]="[{outlets: {
                aux: 'basket',
                primary: ['flight-booking', 'flight-search'] }}]"> … </a>


<a [routerLink]="[{outlets: {
                aux: 'basket',
                primary: ['flight-booking', 'flight-edit', 17] }}]"> … </a>
```

# Code-based Routing

```typescript
export class AppComponent {

    constructor(private router: Router) {}

    activateInfo(): void {
        this.router.navigate([{outlets: { aux: 'info' }}]);
    }

    deactivateInfo(): void {
        this.router.navigate([{outlets: { aux: null }}]);
    }
}
```
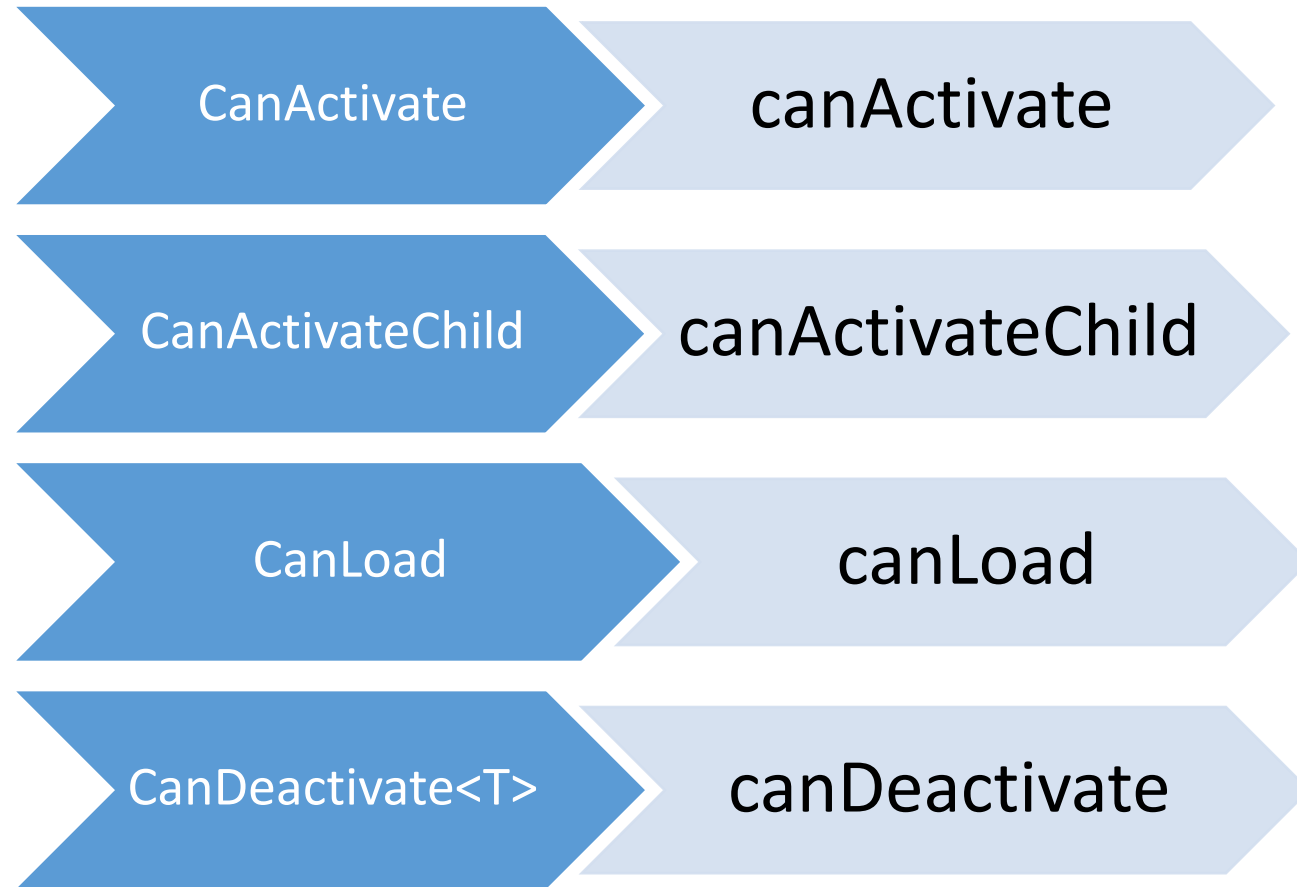
ANGULAR
**ARCHITECTS**
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# DEMO

# Lab

# Guards

# What are Guard?

- Services
- Can prevent the Activation or Deactivation of a Route

# Guards

CanActivate → canActivate

CanActivateChild → canActivateChild

CanLoad → canLoad

CanDeactivate<T> → canDeactivate

**Result: boolean | Observable<boolean> | Promise<boolean>**

# Guards and the Router Configuration

```
const APP_ROUTES: Routes = [
    {
        path: '/flight-booking',
        component: FlightBookingComponent,
        canActivate: [AuthGuard],
        children: [
            {
                path: 'flight-edit/:id',
                component: FlightEditComponent,
                canDeactivate: [FlightEditGuard]
            },
            [...]
        ]
    }
]
```

ANGULAR
**ARCHITECTS**
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Provider for Guards

```typescript
// app.module.ts
@NgModule({
    providers: [
        FlightEditGuard,
        AuthGuard
    ],
    […]
})
export class AppModule {}
```

# DEMO

# Resolver

# What are Resolver?

- Services
- Are activated when the Router switches over to another route
- Can load needed data
- Postpone activation of target route until data is loaded
- Meanwhile, a loading indicator can be shown

# Resolver

```typescript
@Injectable()
export class FlightResolver implements Resolve<Flight>
{
    constructor(private flightService: FlightService) {
    }


    resolve(route, state):
            Observable<Flight> | Promise<Flight> | any {


        return […]
    }
}
```

# Register Resolver

```
const FLIGHT_BOOKING_ROUTES: Routes = [
    […]

    {

        path: 'flight-edit/:id',
        component: FlightEditComponent,
        resolve: {
            flight: FlightResolver    <------------------------- Token
        }
    }

];
```

# Receive Data in Component

```
@Component({ … })
export class FlightEditComponent {

    flight: Flight;

    constructor(private route: ActivatedRoute) { }

    ngOnInit() {
        this.route.data.subscribe(
            data => {
                this.flight = data['flight'];
            }
        );
    }
}
```
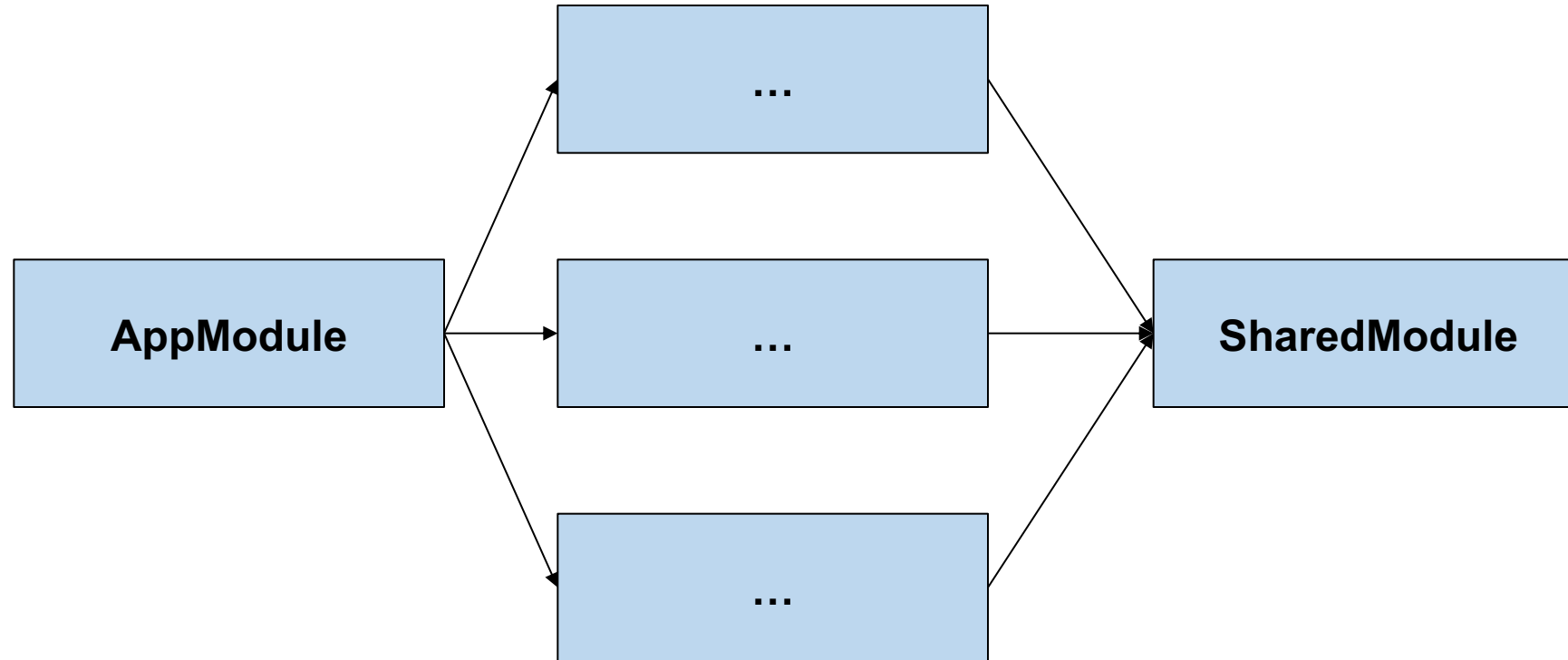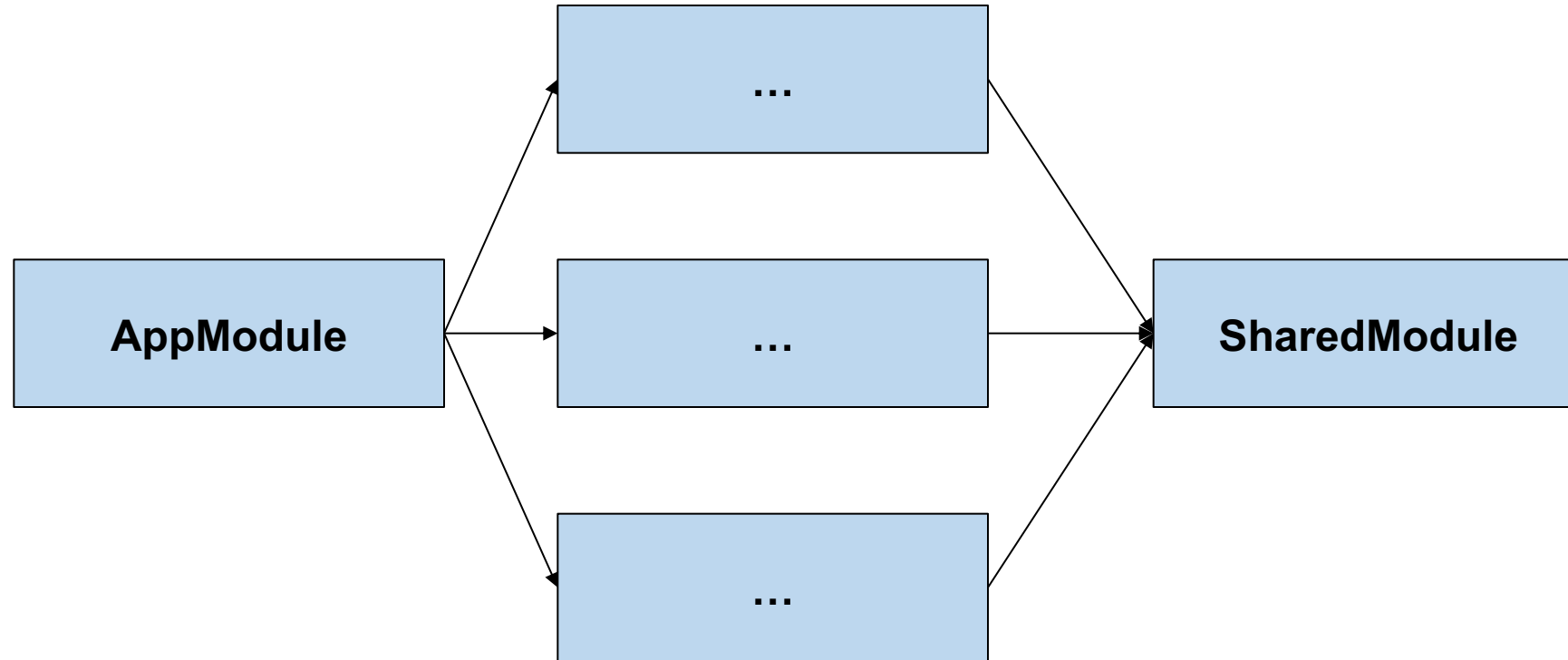
# DEMO

# Lab

ANGULAR
**ARCHITECTS**
INSIDE KNOWLEDGE

SOFTWARE
**ARCHITECT**

Lazy Loading

# Module Structure



**AppModule**

...

...

...

**SharedModule**

**Root Module**          **Feature Modules**          **Shared Module**

ANGULAR
**ARCHITECTS**
INSIDE KNOWLEDGE

SOFTWARE
**ARCHITECT**

# Lazy Loading



**Root Module**          **Feature Modules**          **Shared Module**

```
const APP_ROUTE_CONFIG: Routes = [
    {

        path: 'home',
        component: HomeComponent

    },
    {

        path: 'flight-booking',
        loadChildren: () => import('./[…]/flight-booking.module')
                                .then((m) => m.FlightBookingModule)

    } ,
    {

        path: 'flight-search',
        loadComponent: () => import('./[…]/flight-search.component')
                                .then((c) => c.FlightSearchComponent)

    }
];
```

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Routes for Feature Module

```
const FLIGHT_ROUTES =    [
    {
        path: '',
        component: FlightBookingComponent,
        […]
    },
    […]
}
```

# Routes for Feature Module

```
const FLIGHT_ROUTES =    [
    {
        path: '/bookings',
        component: FlightBookingComponent,
        […]
    },
    […]
}
```

Url: /flight-booking/bookings

**Triggers Lazy Loading**

# DEMO

Preloading

# Idea

- Modules that **might be needed** later are loaded after (!) the start of the application
- When the module is actually needed, it is available **immediately**

# Activating Preloading - Classic

```
const AppRoutesModule =
        RouterModule.forRoot(
            ROUTE_CONFIG,
            { preloadingStrategy: PreloadAllModules });
```

# Activating Preloading - Standalone

```
bootstrapApplication(AppComponent, {
  providers: [
    […]
    provideRouter(APP_ROUTES,
        withPreloading(PreloadAllModules)
    ),
  ],
});
```

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# DEMO

# Lab

# Conclusion

| Child Routes & Parameters | Aux Routes |
|---|---|
| Guards | Lazy Loading |

# Summary

- Child Routes

- Aux Routes

- Guards and Resolvers

- Lazy Loading and Preloading