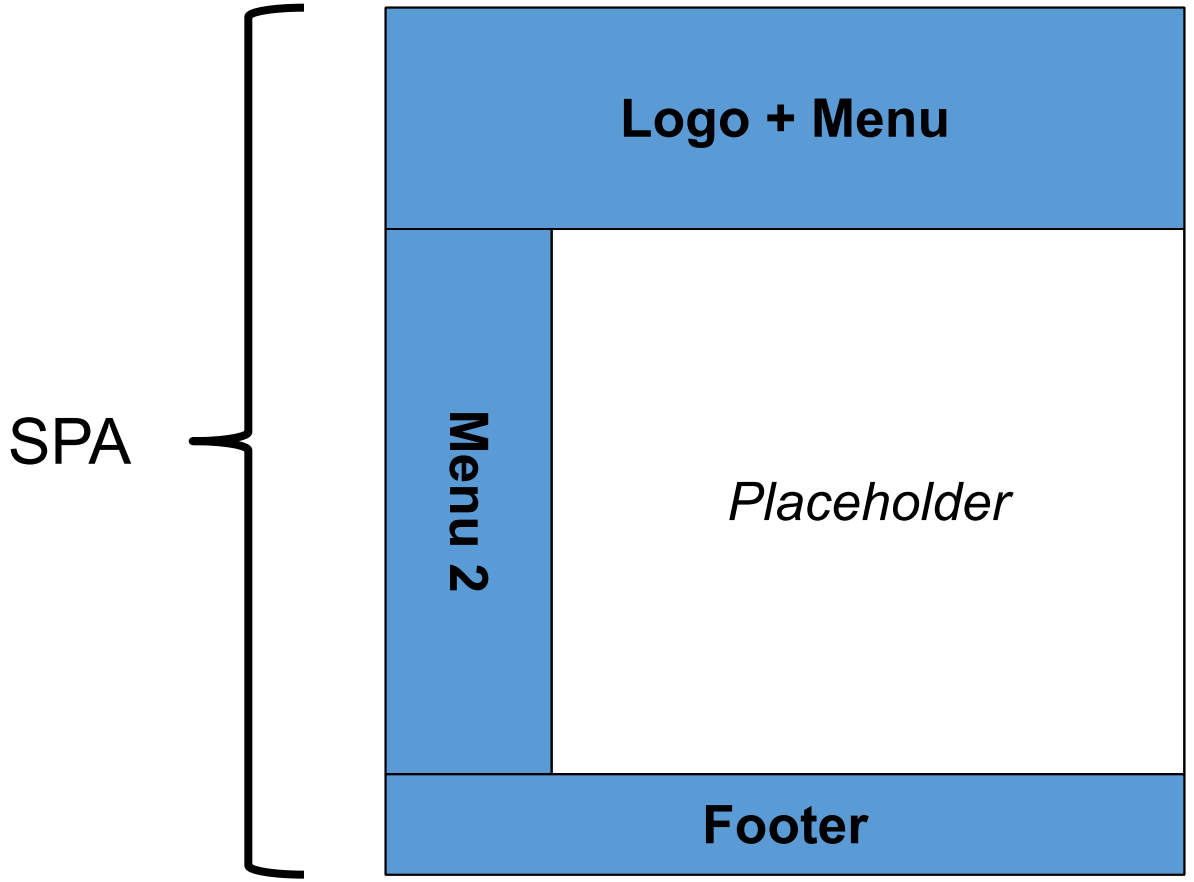

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE
Router Deep Dive
ANGULARarchitects.io

Contents

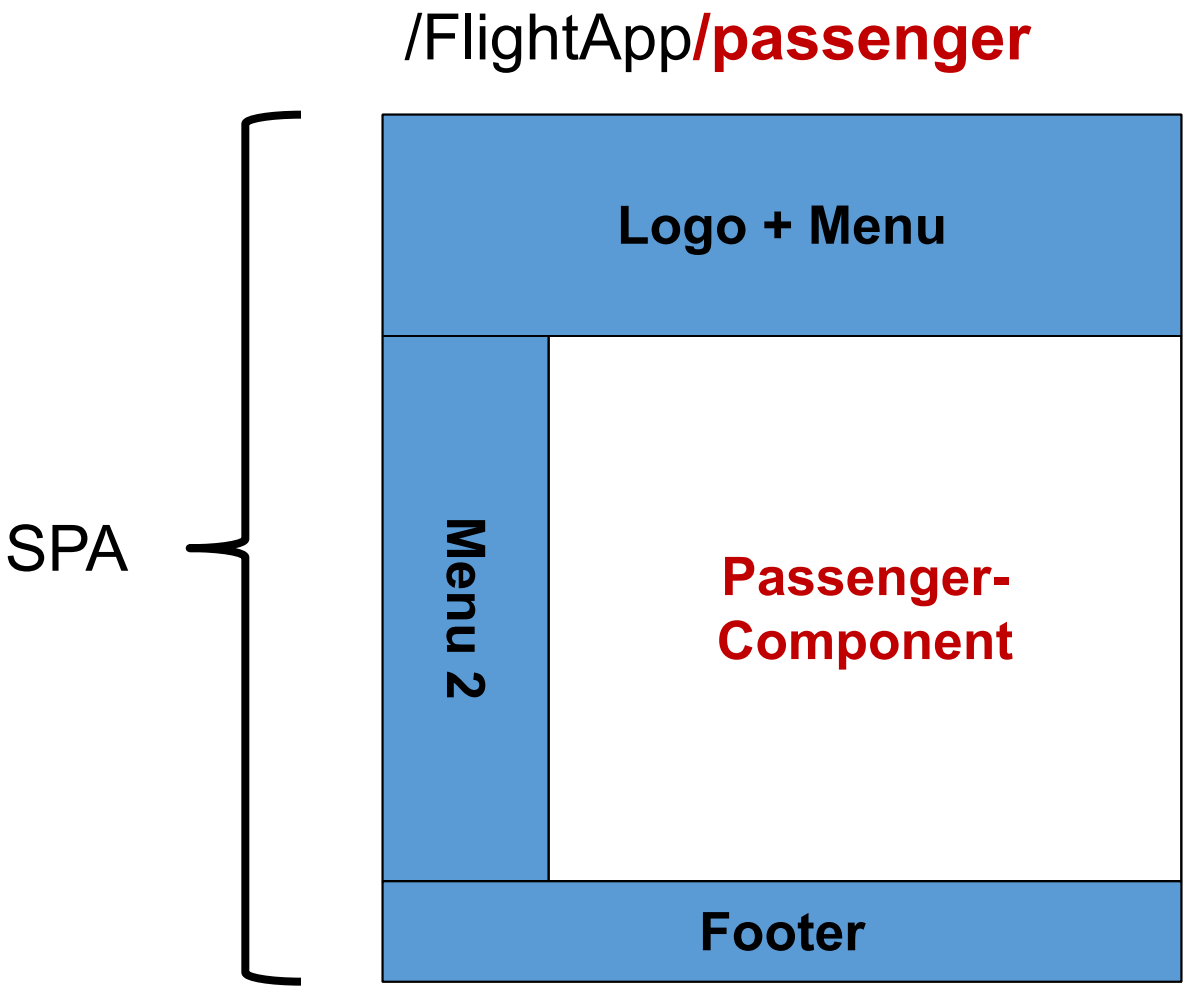
- Basics & Parameters
- Child Routes
- Aux Routes
- Guards
- Resolver
- Lazy Loading

Angular Router

Routing in Angular



Routing in Angular



Configuration

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

Configuration

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

For Root-Module

For Feature-Module: forChild

AppComponent

```
<a routerLink="/home">Home</a>
<a [routerLink]="variableName">Flight Search</a>

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

Parameters

- passenger/7
- 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
    }
}
```

Reading Parameters

```
export class FlightEditComponent {

    public id: string;

    constructor(
        private route: ActivatedRoute) {

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

Links for Routes with Parameters

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

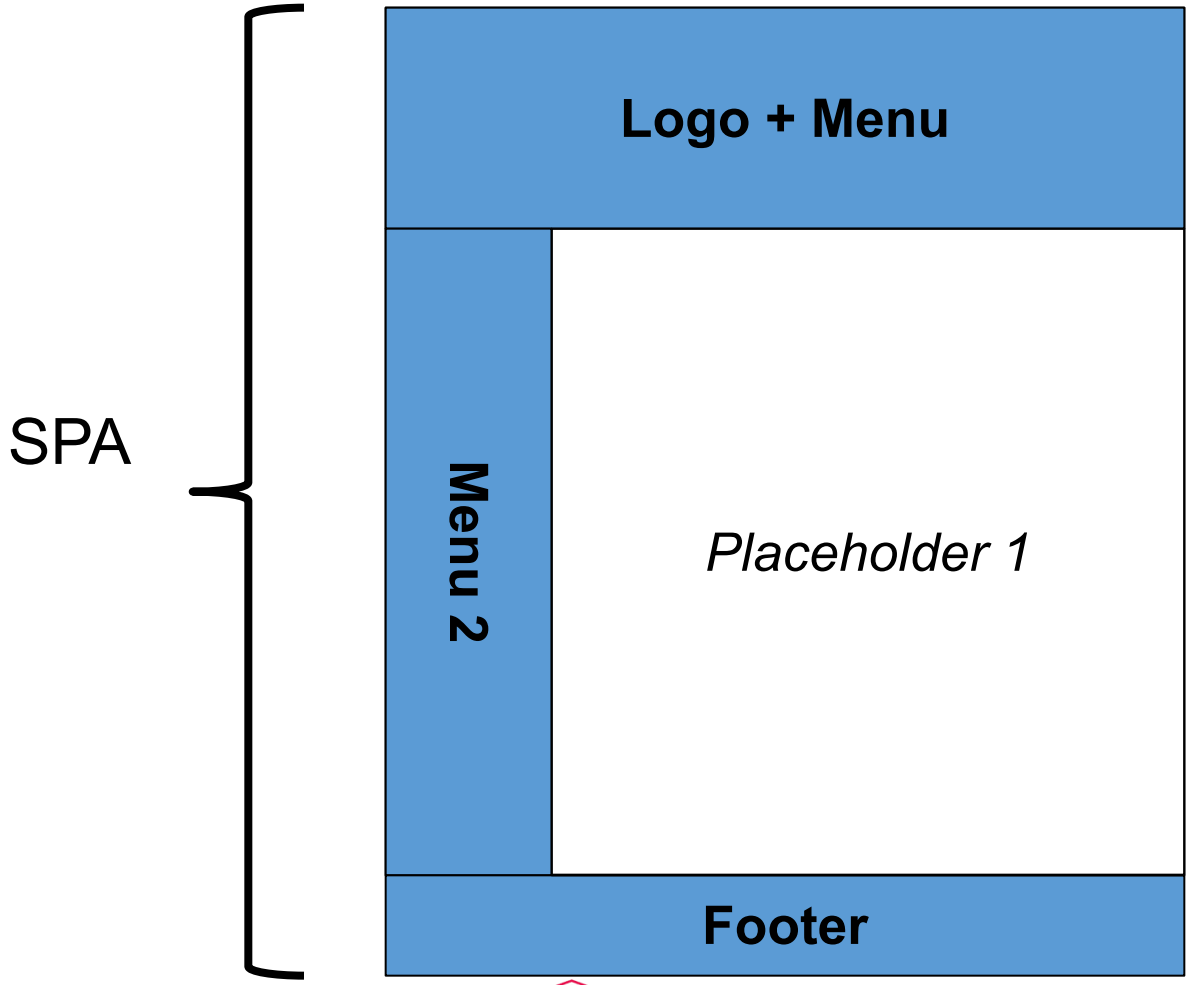
DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

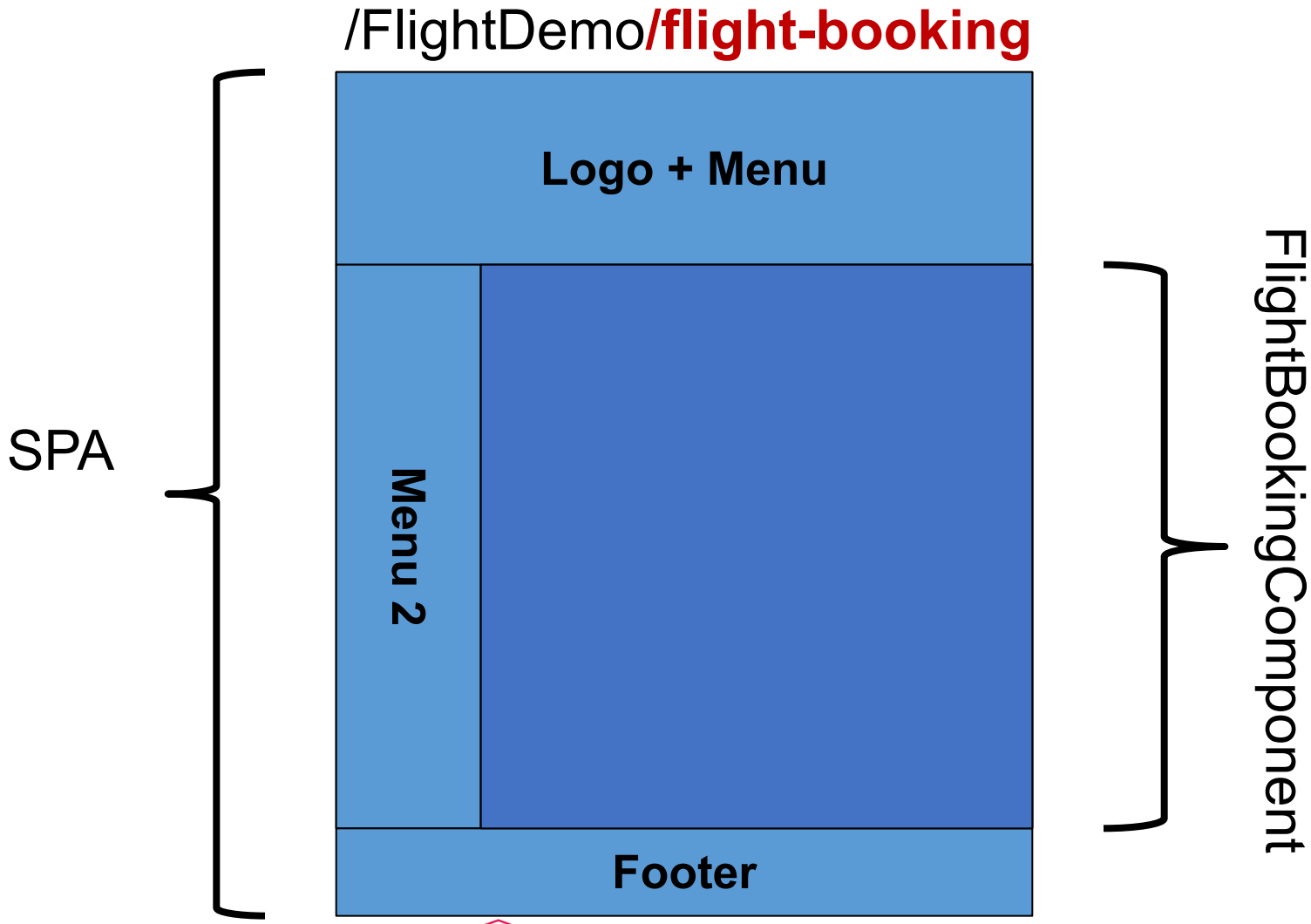
SOFTWARE
ARCHITECT

Hierarchical Routing

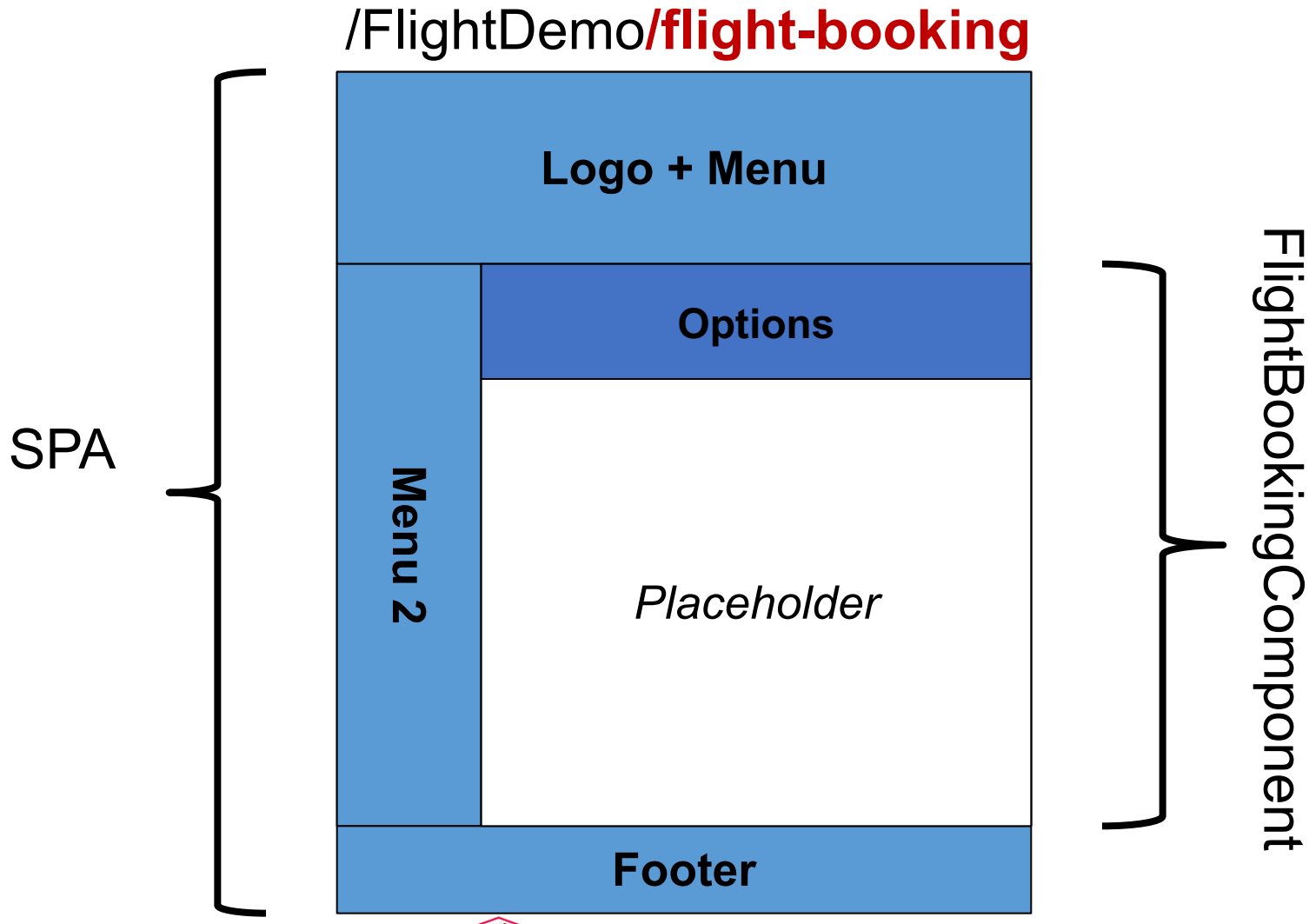
Hierarchical Routing



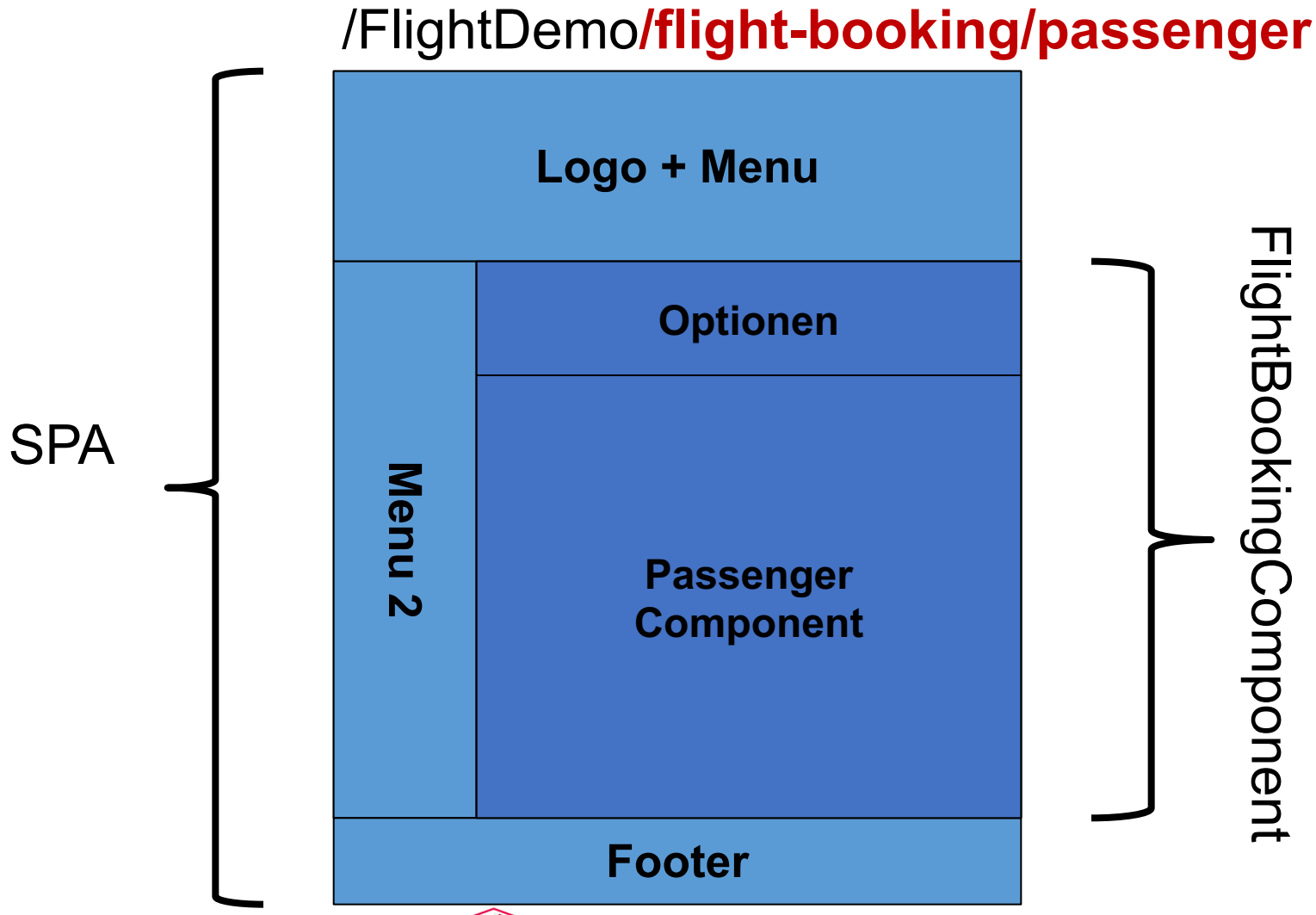
Hierarchical Routing



Hierarchical Routing



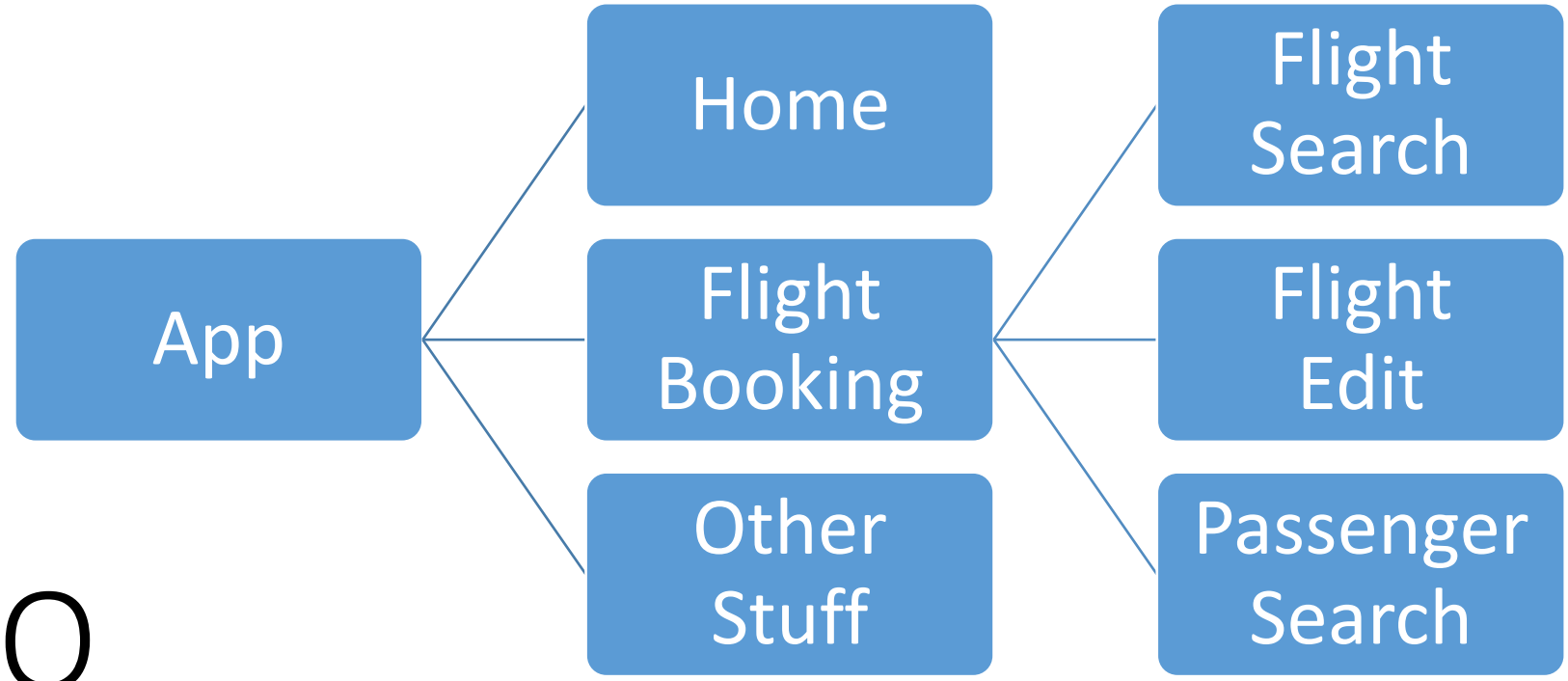
Hierarchical Routing



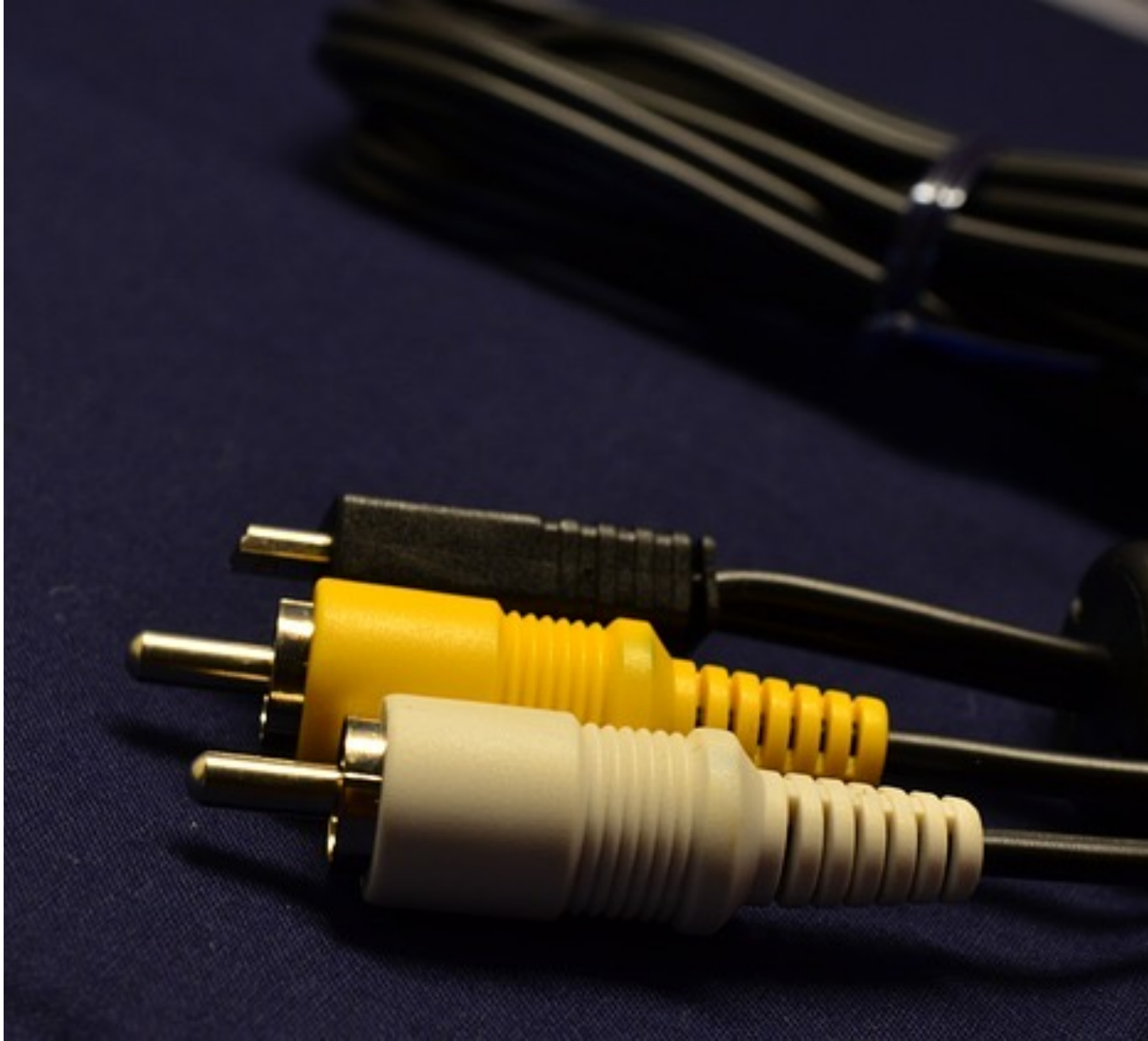
Configuration

```
const APP_ROUTES: Routes = [
    {
        path: '',
        component: HomeComponent
    },
    {
        path: 'flight-booking',
        component: FlightBookingComponent,
        children: [
            {
                path: 'flight-search',
                component: FlightSearchComponent
            },
            […]
        ]
    }
];
```

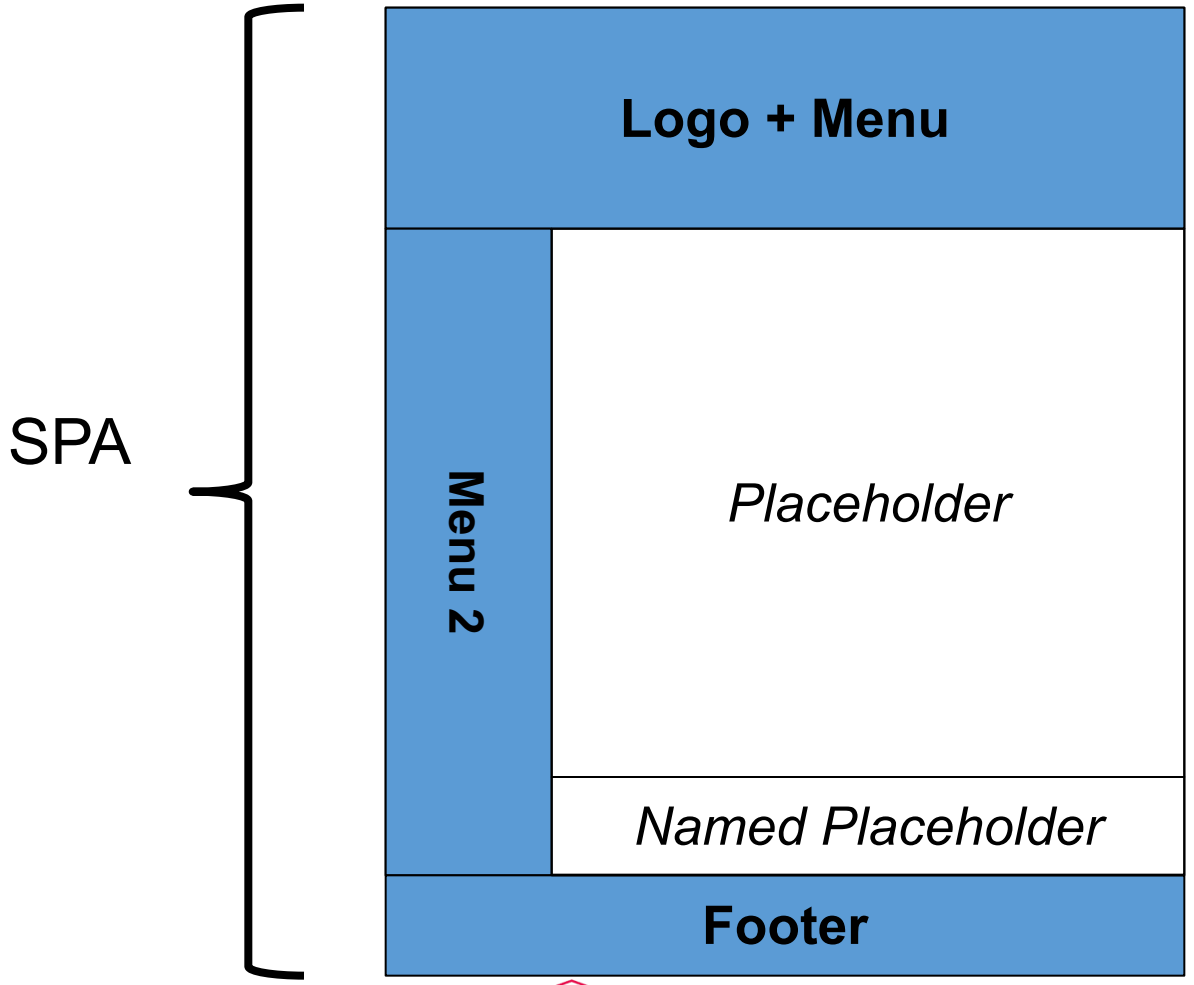
DEMO



Aux Routes

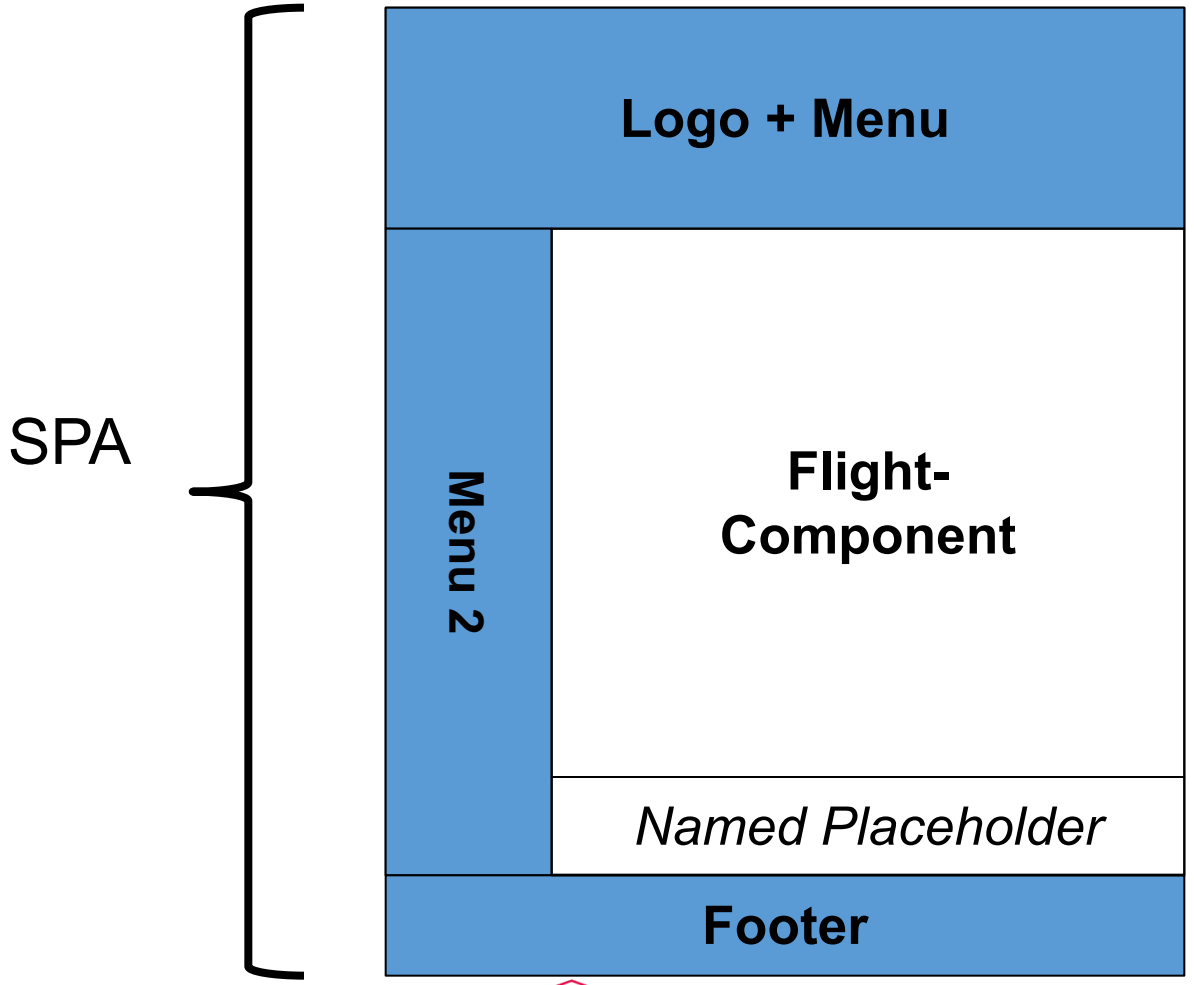


Aux-Routes



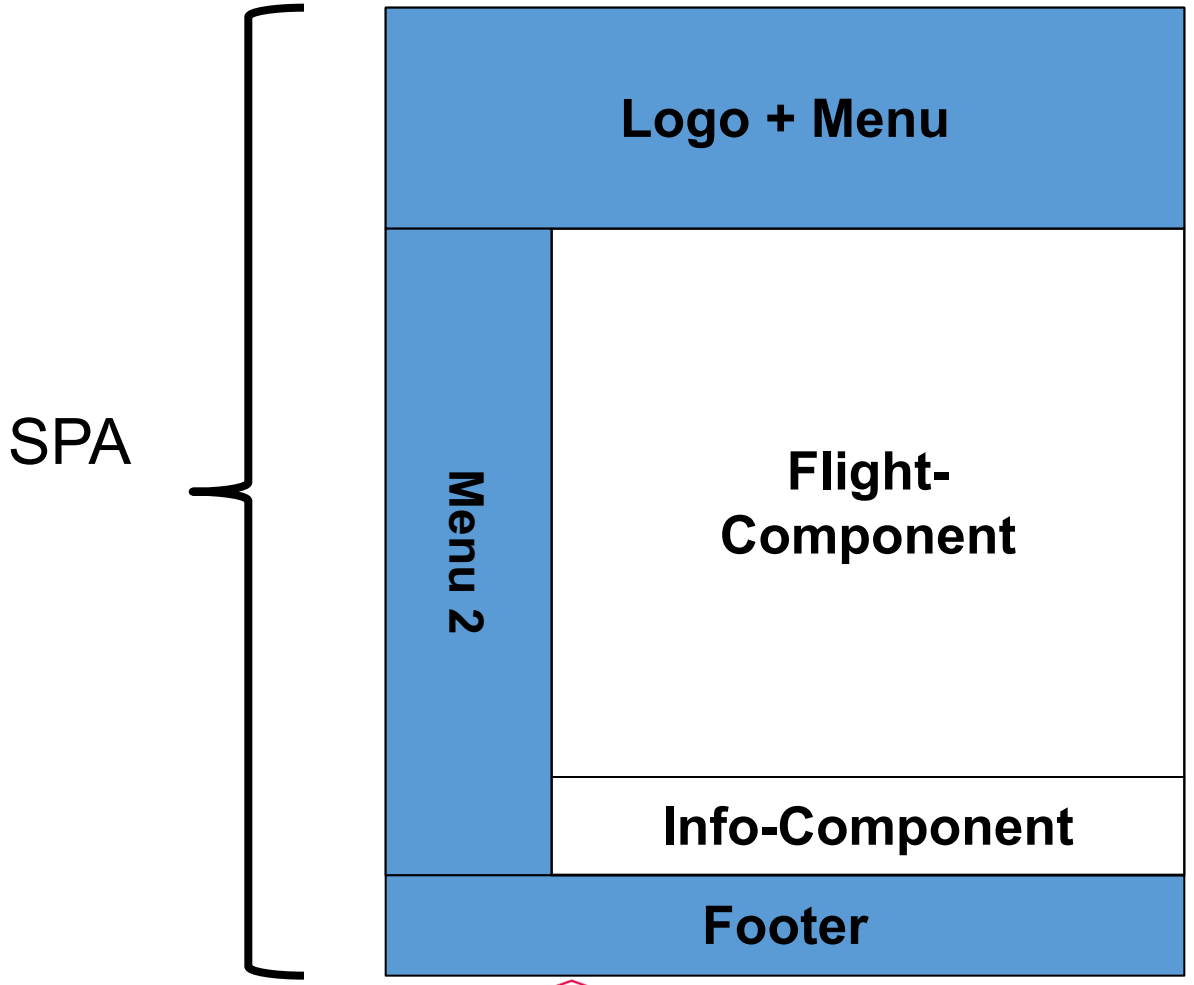
Aux-Routes

/FlightApp/**flights**



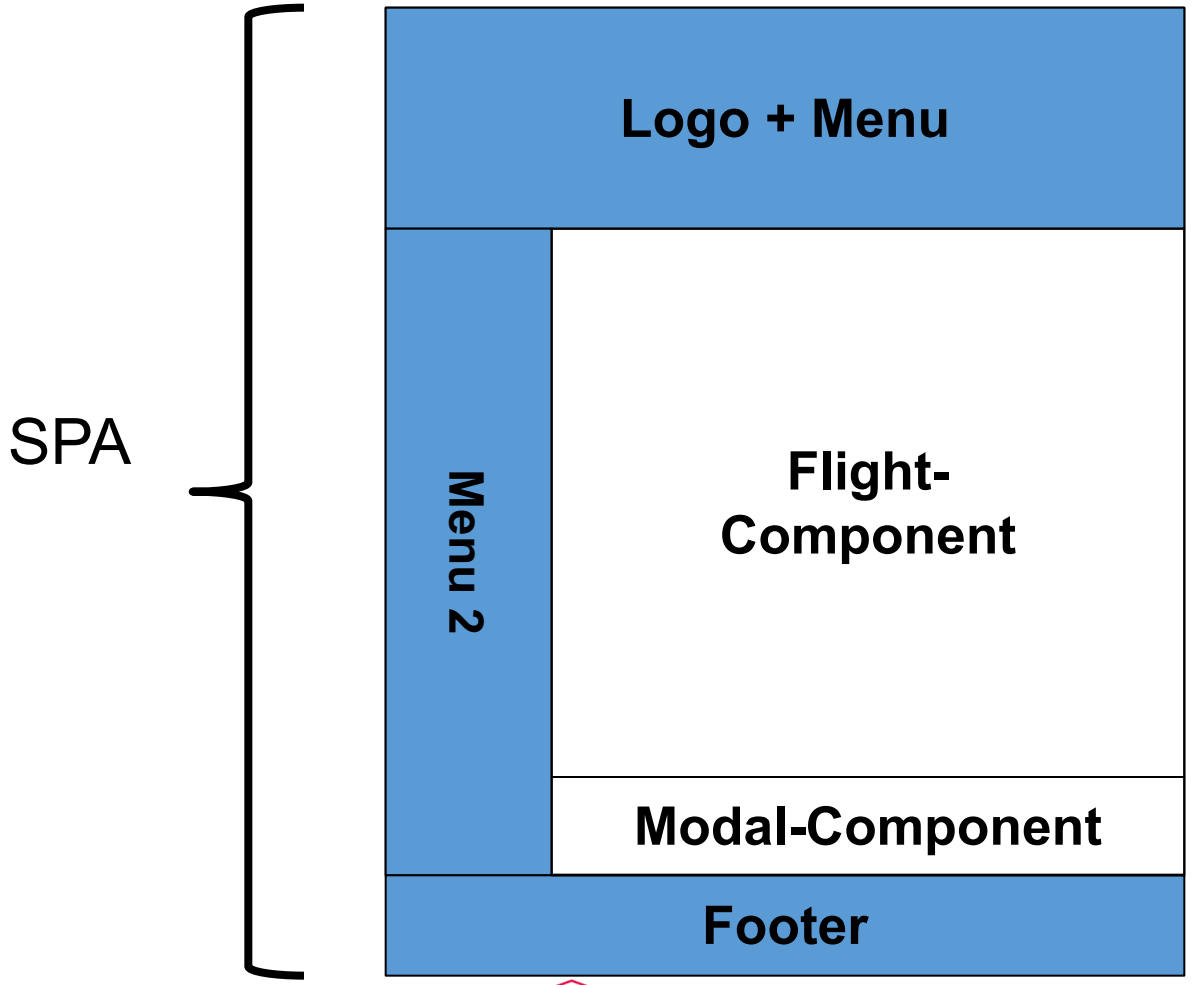
Aux-Routes

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



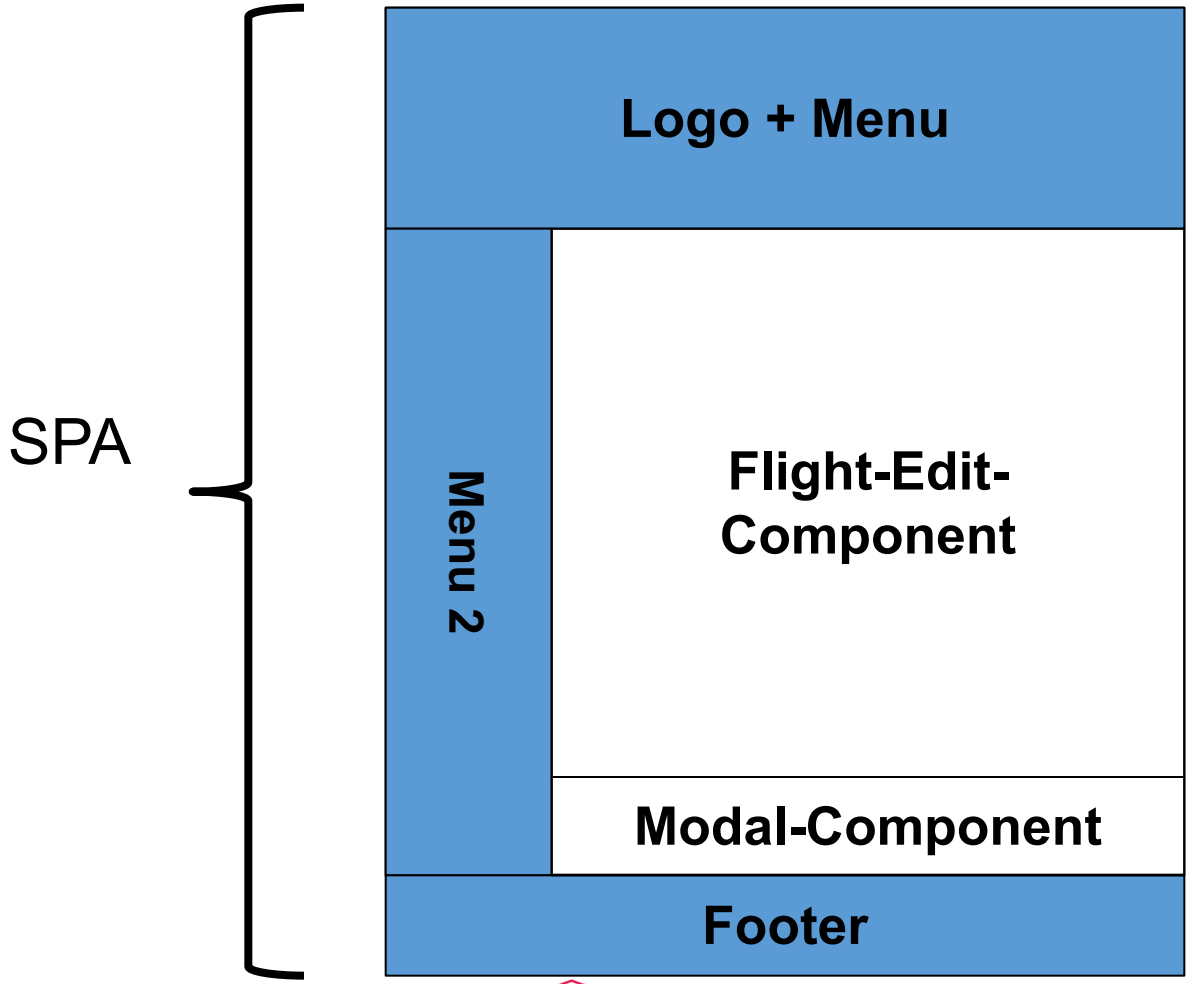
Aux-Routes

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



Aux-Routes

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



Use Cases

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

Define Outlets

Default Name: primary

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

<hr>

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

Configuration

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

Activating Aux-Routes

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

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

Activating Several Aux Routes at Once

```
<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

```
export class AppComponent {

    constructor(private router: Router) {}

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

    deactivateInfo(): void {
        this.router.navigate([{outlets: { aux: null }}]);
    }
}
```

DEMO

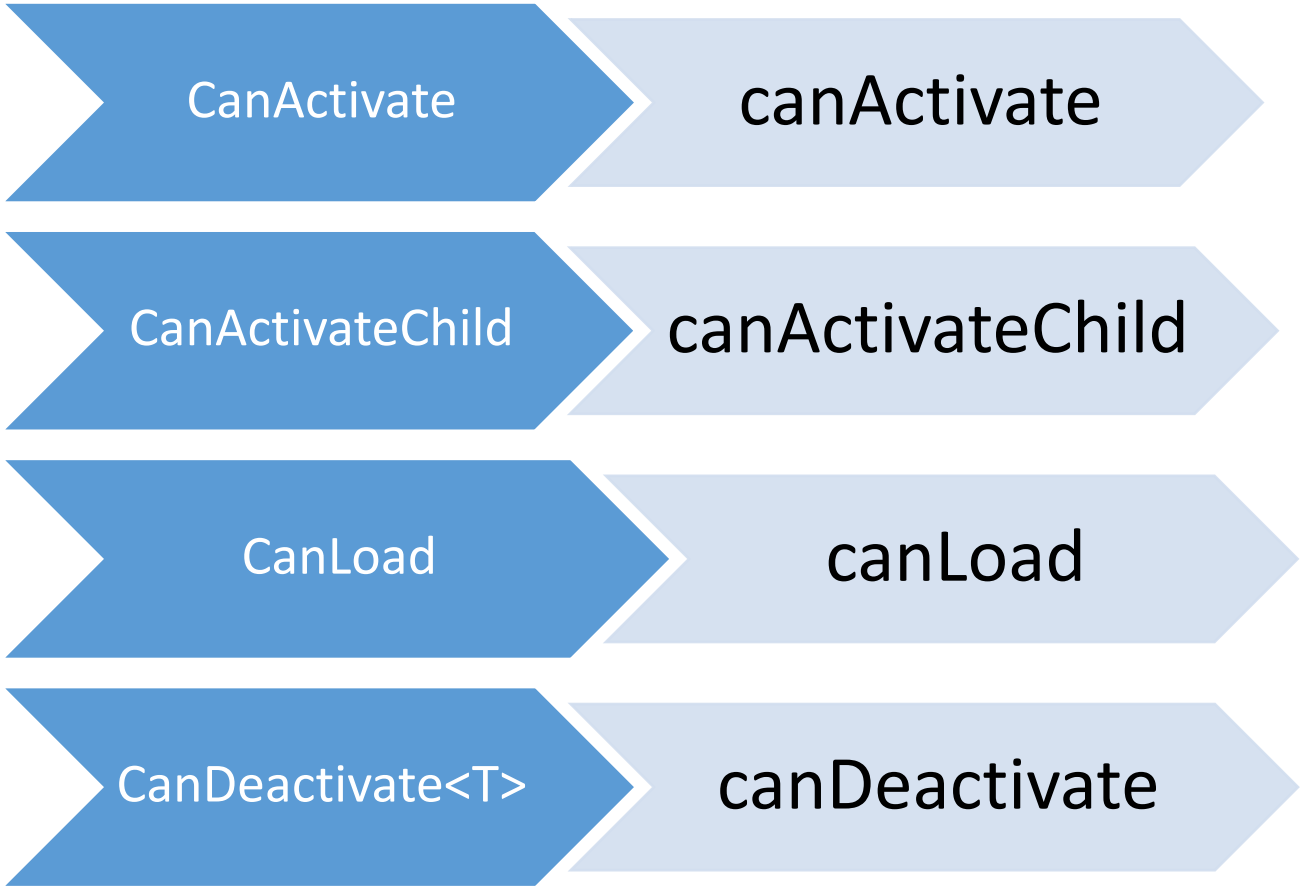
Lab

Guards

What are Guard?

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

Guards



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]
            },
            […]
        ]
]
```

Provider for Guards

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

DEMO

Resolver

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE ARCHITECT

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

```
@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

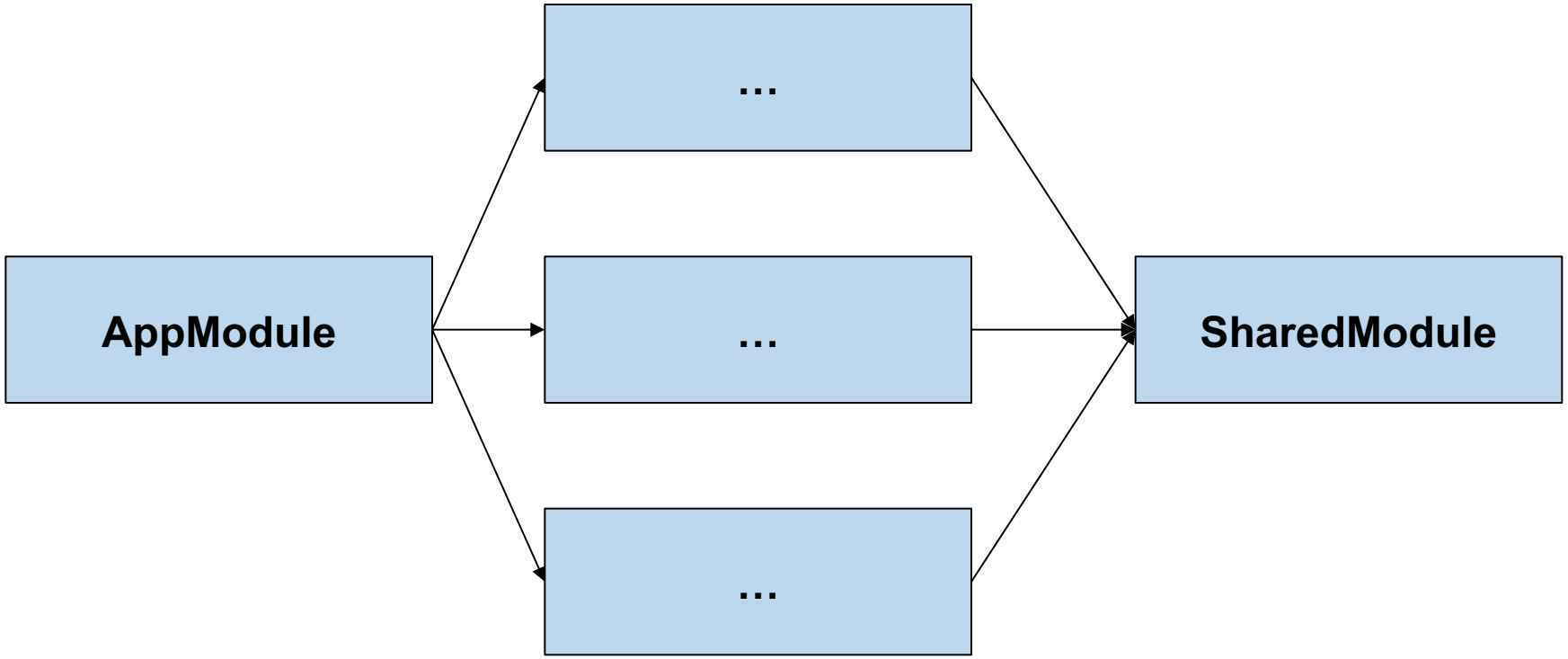
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Lab

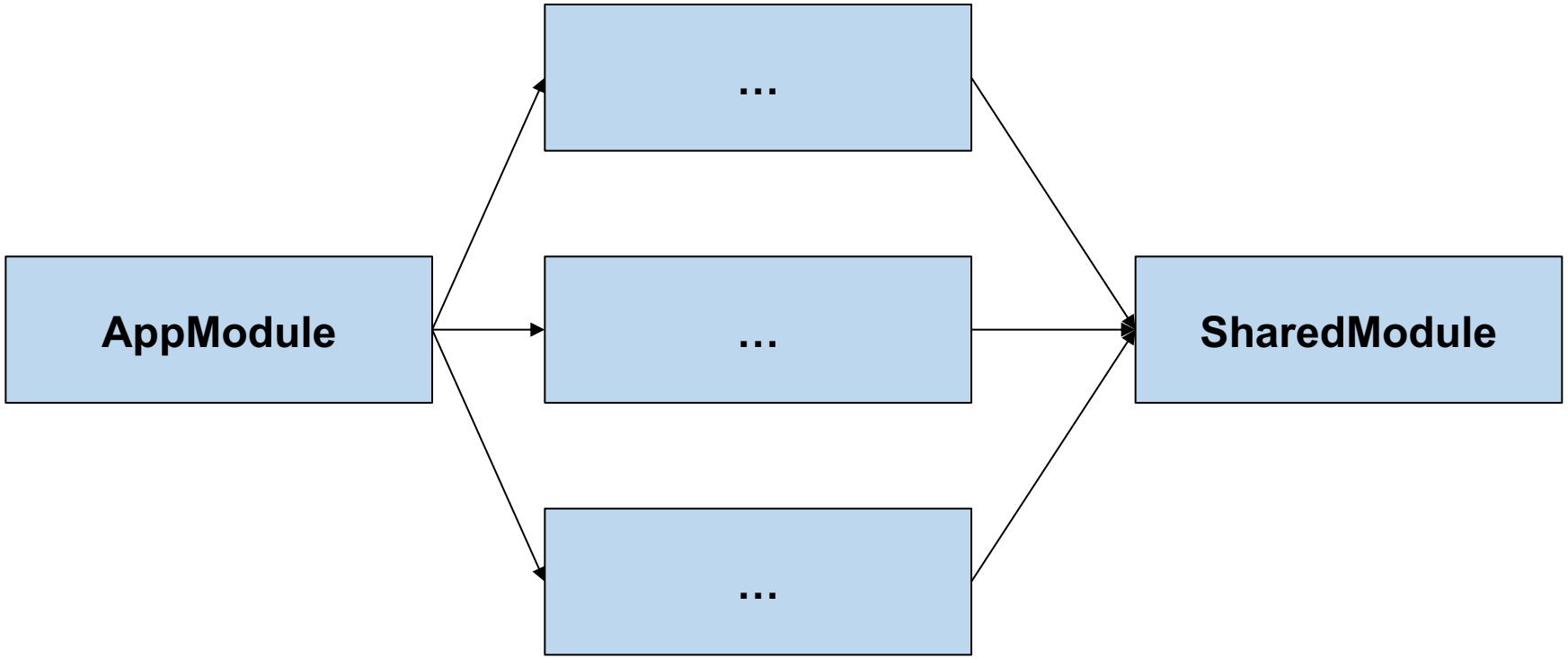
Lazy Loading

Module Structure



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

Lazy Loading



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

Root Module with Lazy Loading

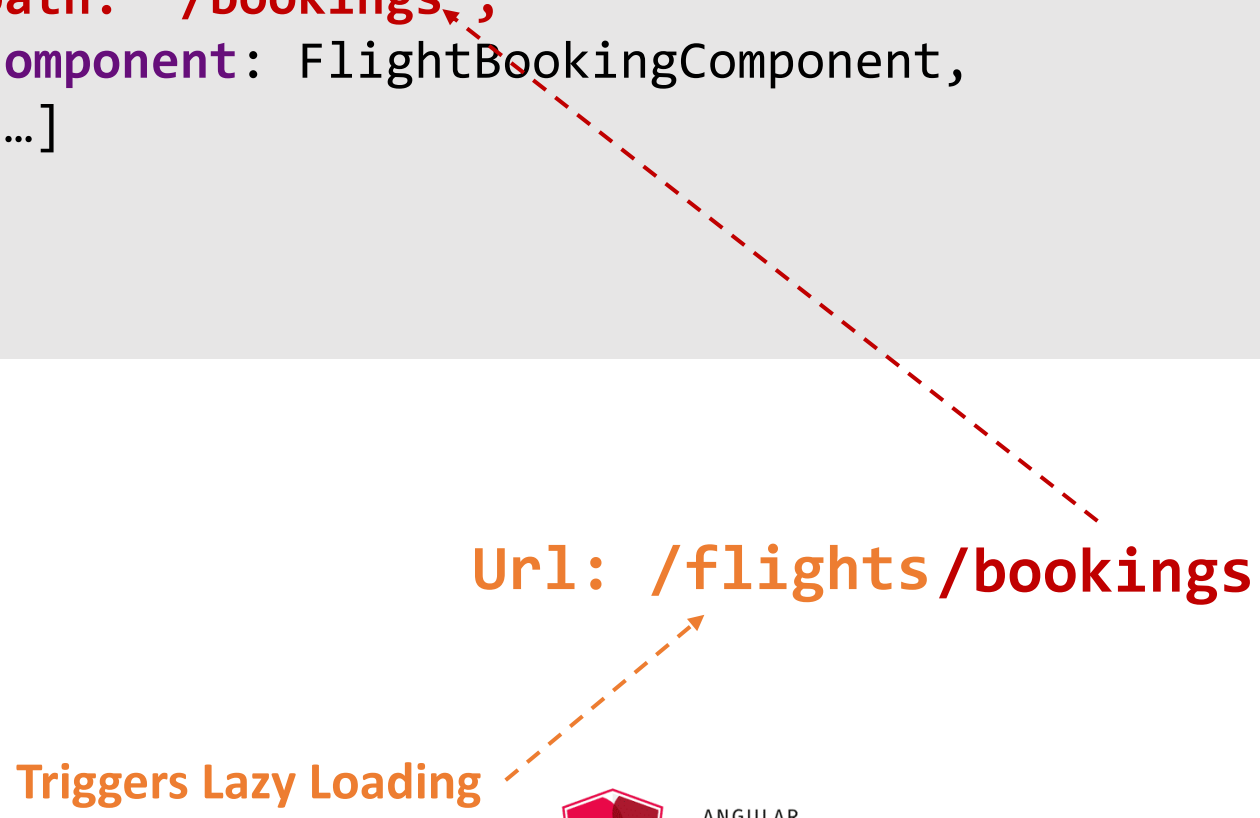
```
const APP_ROUTE_CONFIG: Routes = [
    {
        path: 'home',
        component: HomeComponent
    },
    {
        path: 'flights',
        loadChildren: () => import('./[…]/flight-booking.module')
                                .then(m => m.FlightBookingModule)
    }
];
```

Routes for Feature Module

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

Routes for Feature Module

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



DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

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

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

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Lab

Conclusion

Child Routes

Aux Routes

Guards

Lazy Loading