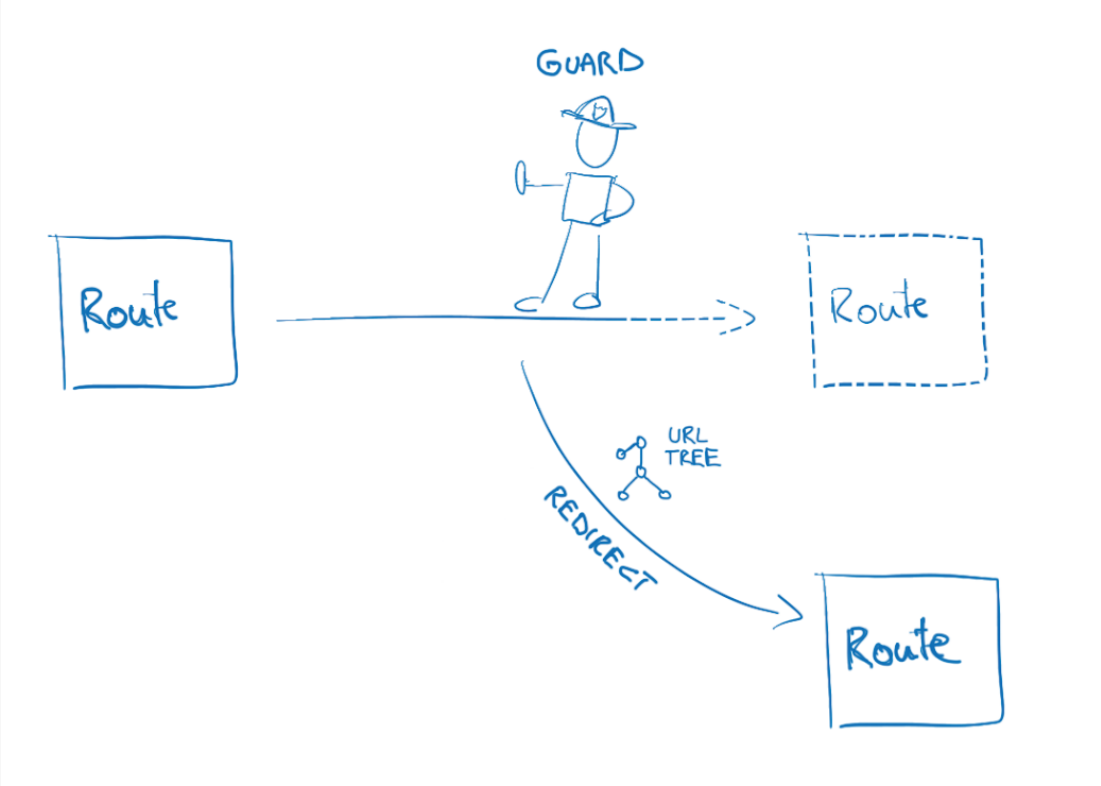

# Guards Incerceptors Configs

*Fravezzi Mattia*
*m.fravezzi@almaviva.it*

# Guards



AlmavivA

# Routes

```
const routes: Routes = [{
    path: AppRoutings.adminPage,
    loadChildren: () => import('./features/admin-page/admin-page.module').then(res => res.AdminPageModule),

    canLoad: [AuthGuard, RoleGuard],
    canActivate: [AuthGuard, RoleGuard],
    data: {
        role: AppRoles.Admin
    }
}];

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

AlmavivA

# Preventing unauthorized accesslink

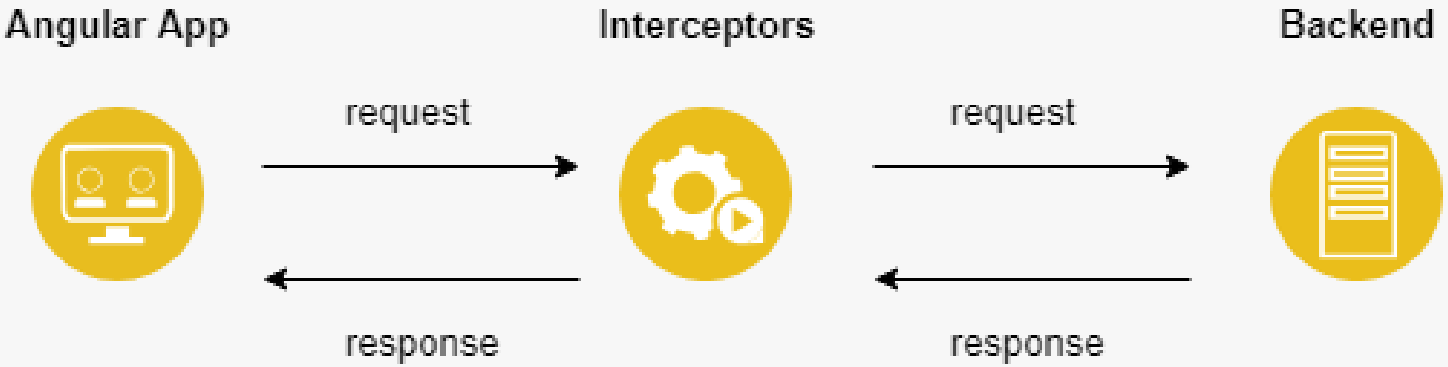
- `CanActivate`
- `CanActivateChild`
- `CanDeactivate`
- `Resolve`
- `CanLoad`

AlmavivA

# Auth Guard

```
export class AuthGuard implements CanLoad, CanActivate {
    constructor( private localStorageService: LocalStorageService, private router: Router ) { }
    public canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean | UrlTree |
Observable<boolean | UrlTree> | Promise<boolean | UrlTree> {
        return this.checkAuthorization();
    }
    public canLoad(
        route: Route,
        segments: UrlSegment[]): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
        return this.checkAuthorization();
    }
    private checkAuthorization(): boolean | UrlTree | Observable<boolean | UrlTree> | Promise<boolean | UrlTree> {
        return this.localStorageService.getItem(LocalStorageItem.user)
            ? true
            : this.router.createUrlTree([AppRoutings.loginPage]);
    }
}
```

# Interceptor



AlmavivA

# Intercep request

```
@Injectable()
export class JwtInterceptor implements HttpInterceptor {
    constructor( public router: Router, public localStorageService: LocalStorageService ) { }
    intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpEvent<unknown>> {
      let currentUser = this.localStorageService.getItem(LocalStorageItem.user);
      if (currentUser && currentUser.token) {
          request = request.clone({
              setHeaders: {
                  Authorization: `Bearer ${currentUser.token}`
              }
          });
      }
      return next.handle(request);
    }
}
```

AlmavivA

# Intercep response

```
@Injectable()
export class ErrorInterceptor implements HttpInterceptor {
    constructor( public router: Router ) { }
    intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
        return next.handle(request)
            .pipe(catchError((err: any) => {
                if (err.status === 401) {
                    this.router.navigate([AppRoutings.loginPage]);
                    return throwError(null);
                }
                ...
                return throwError(error);
            }))
    }
}
```

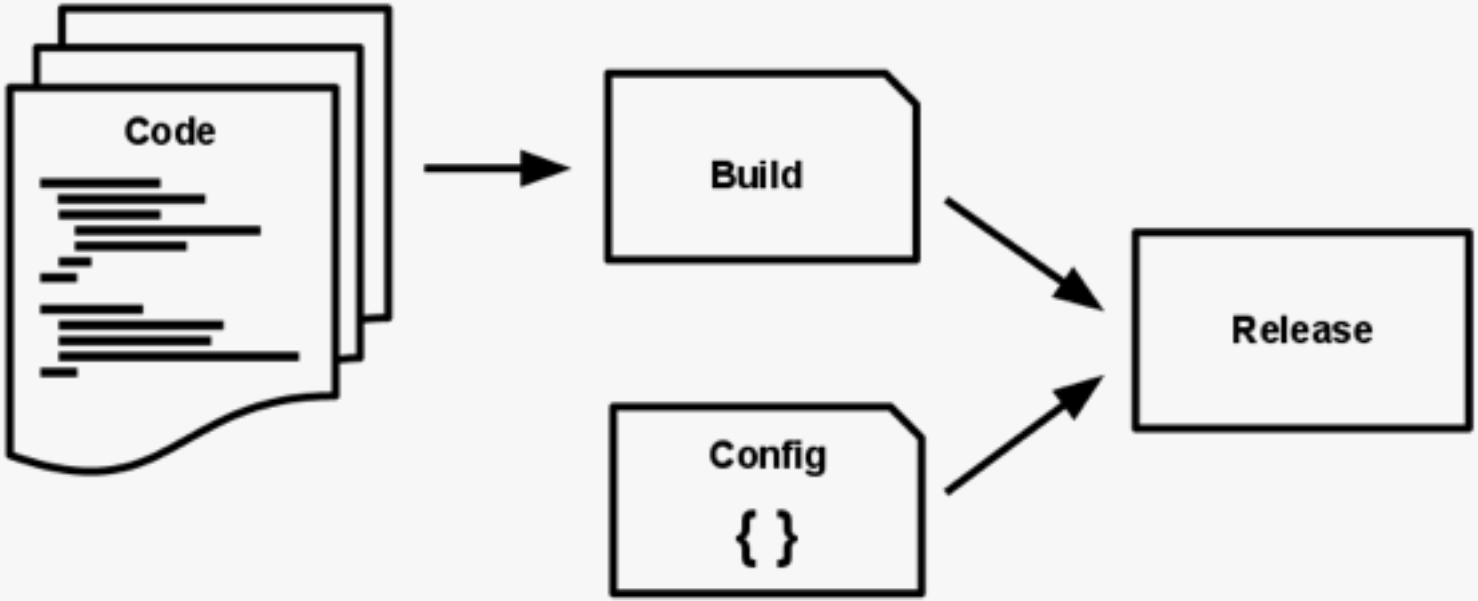
# Twelve factor app

## V. Build, release, run

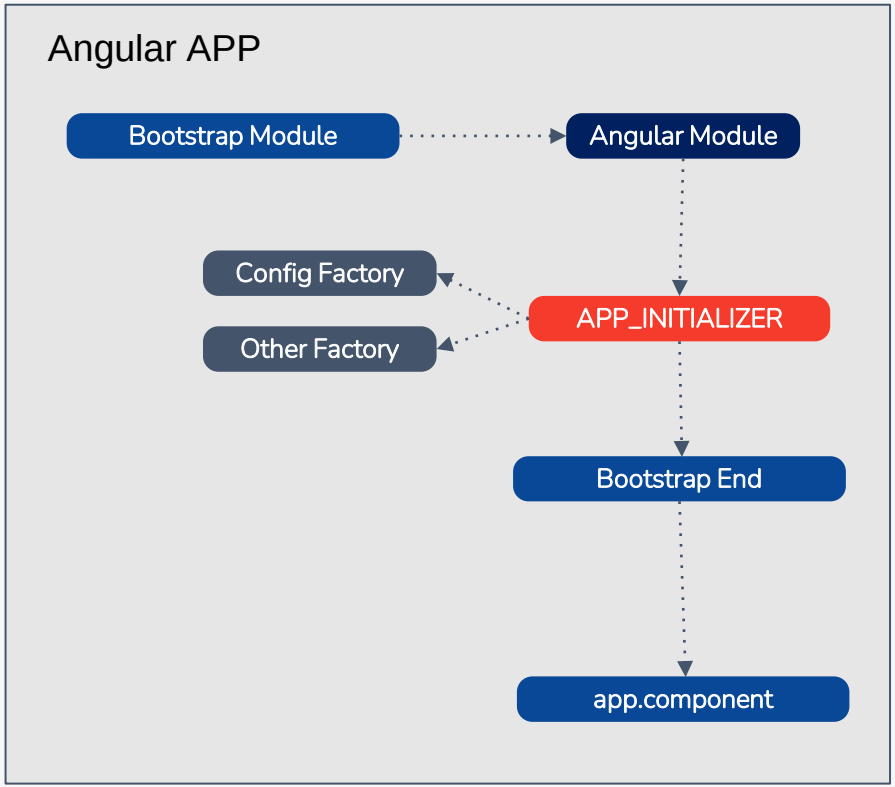
separare in modo netto lo stadio di build dall'esecuzione

- la fase di **build** che converte il codice del repo in una build "eseguibile".
- la fase di **release** prende la build prodotta nella fase precedente e la combina con l'attuale insieme di impostazioni di configurazione del deployment specifico.
- la fase di **esecuzione** vede l'applicazione in esecuzione nell'ambiente di destinazione, attraverso l'avvio di processi della release scelta.

# V. Build, release, run



# Bootstrap




```
providers: [{
    provide: APP_INITIALIZER,
    useFactory: AppConfigLoader.loaderForFactory,
    deps: [AppConfigLoader],
    multi: true
},{
    provide: AppConfig,
    useFactory: AppConfigLoader.getterForFactory,
    deps: [AppConfigLoader]
}]
```

AlmavivA

# Config Loader

```
@Injectable()
export class  AppConfigLoader {
    public config!: AppConfig;
    public httpClient: HttpClient;
    public static loaderForFactory(loader: AppConfigLoader) { return () => loader.read() }
    public static getterForFactory(loader: AppConfigLoader) { return loader.config; }
    constructor( protected handler: HttpBackend) { this.httpClient = new HttpClient(handler);}
    private read() {
        return this.httpClient
            .get<AppConfig>(this.getConfigUri())
            .pipe(tap(res => this.config = res))
            .toPromise()
    }
    private getConfigUri(): string { return `./assets/configs/config.json?v=${environment.appVersion}` }
}
```

# DEMO

AlmavivA
digitale assoluto

almaviva.it