Sesión 02

Seguridad en angular

Instructor:

ERICK ARÓSTEGUI

earostegui@galaxy.edu.pe

GALAXY
TRAINING

ÍNDICE

GALAXY
TRAINING

01

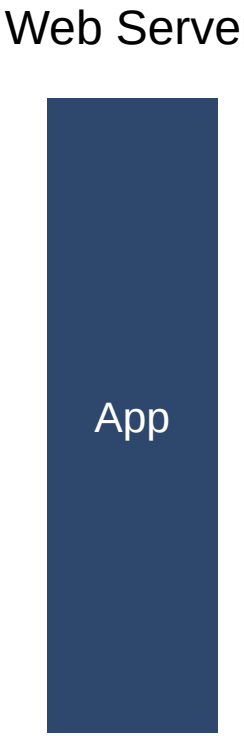
Lazy Loading de componentes

GALAXY
TRAINING

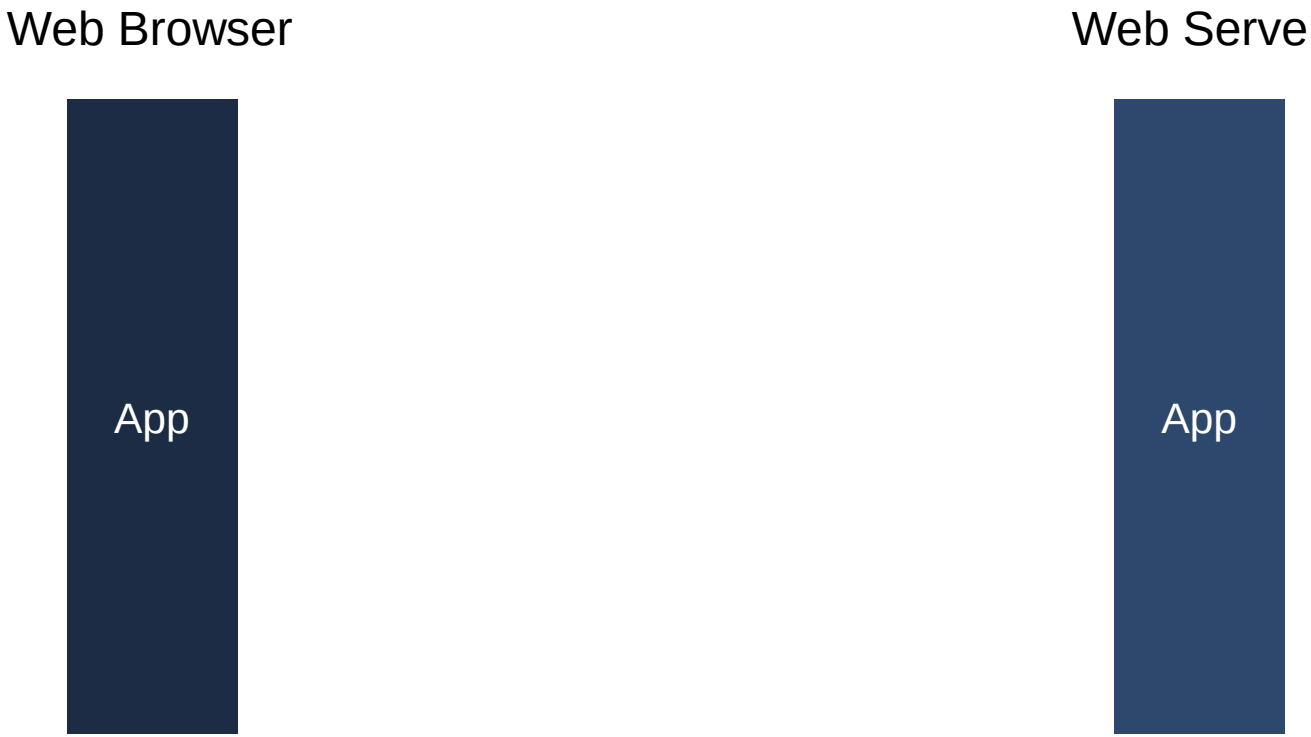
GALAXY TRAINING

Carga de componentes

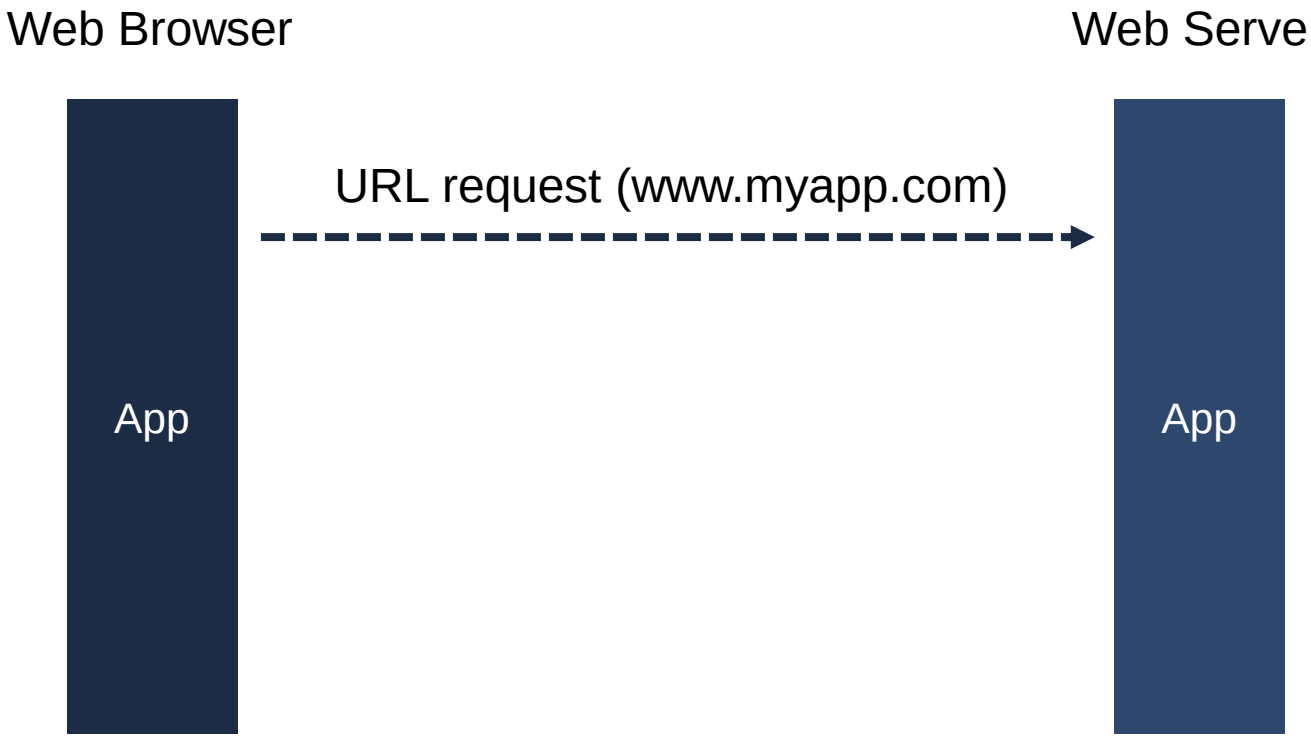
Carga de componentes



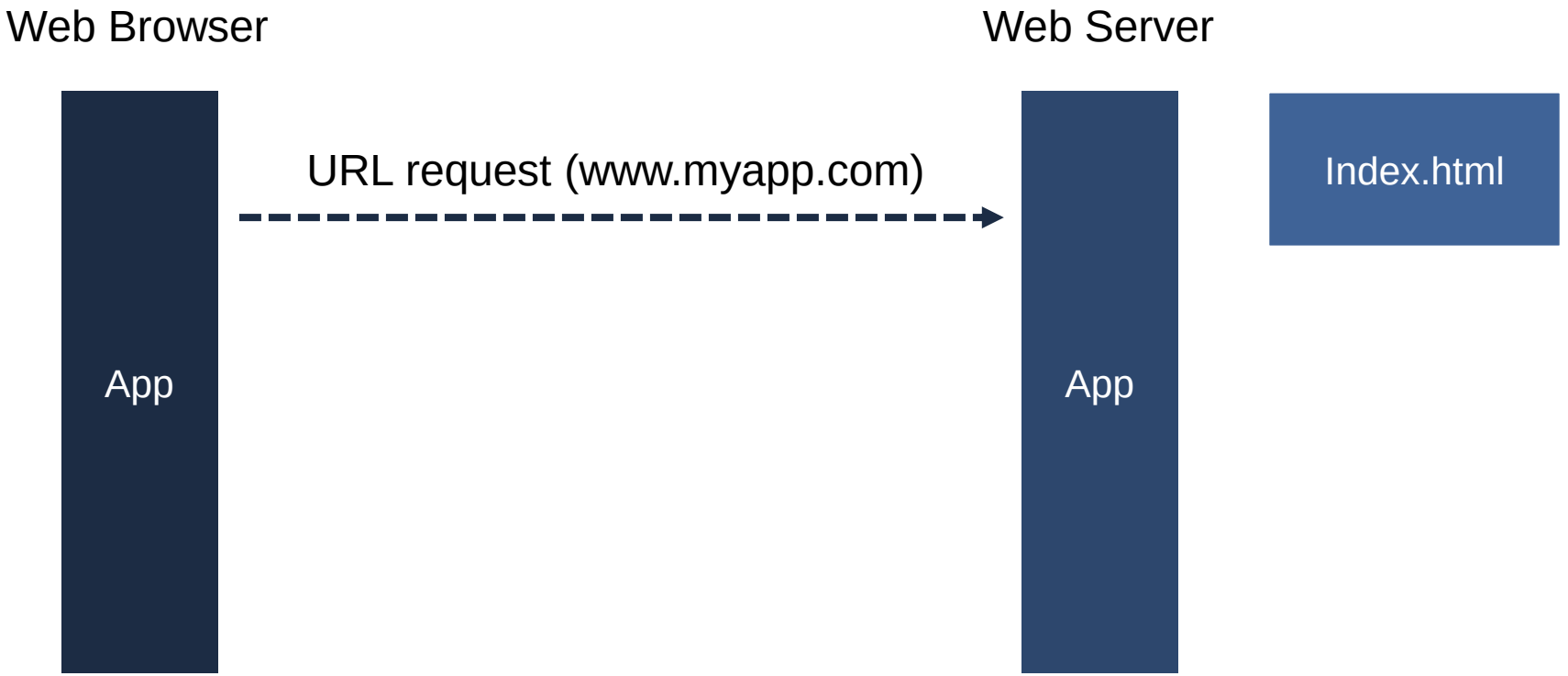
Carga de componentes



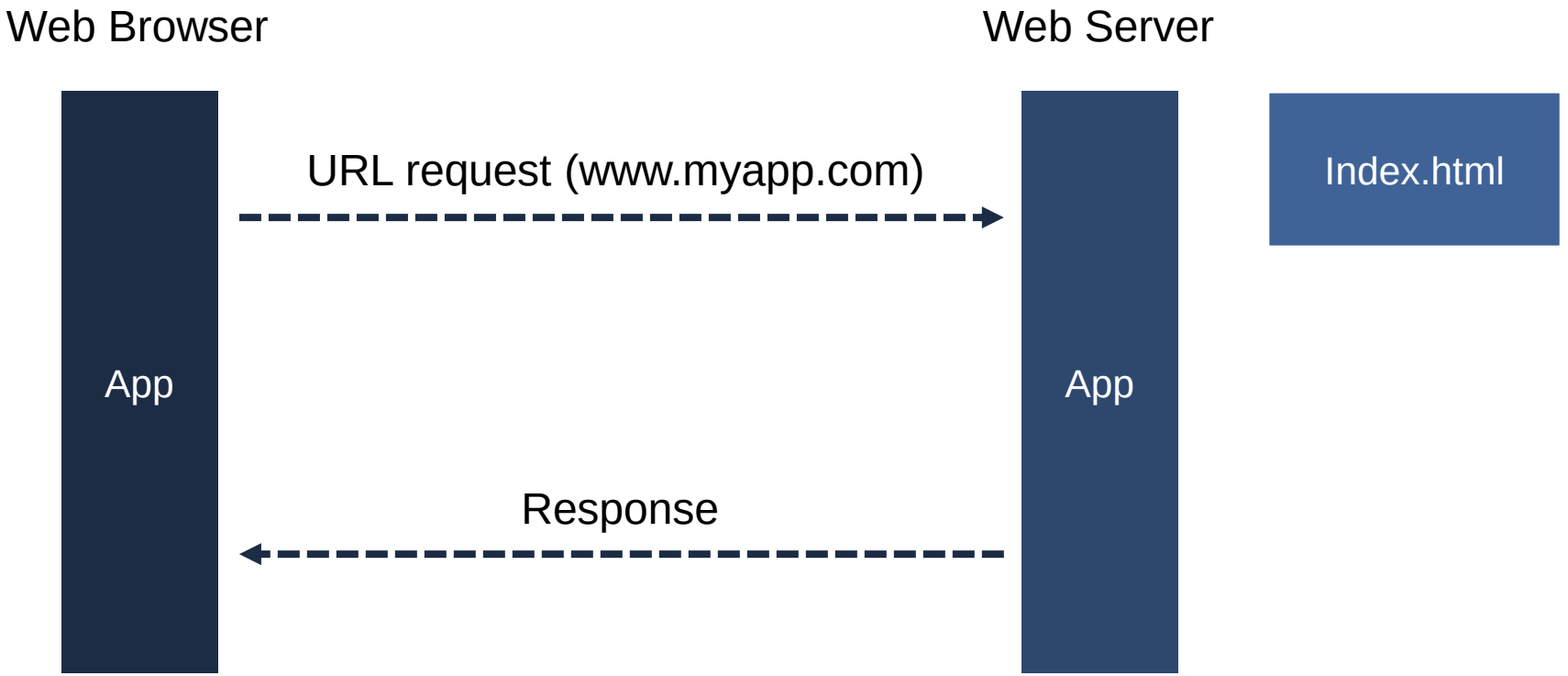
Carga de componentes



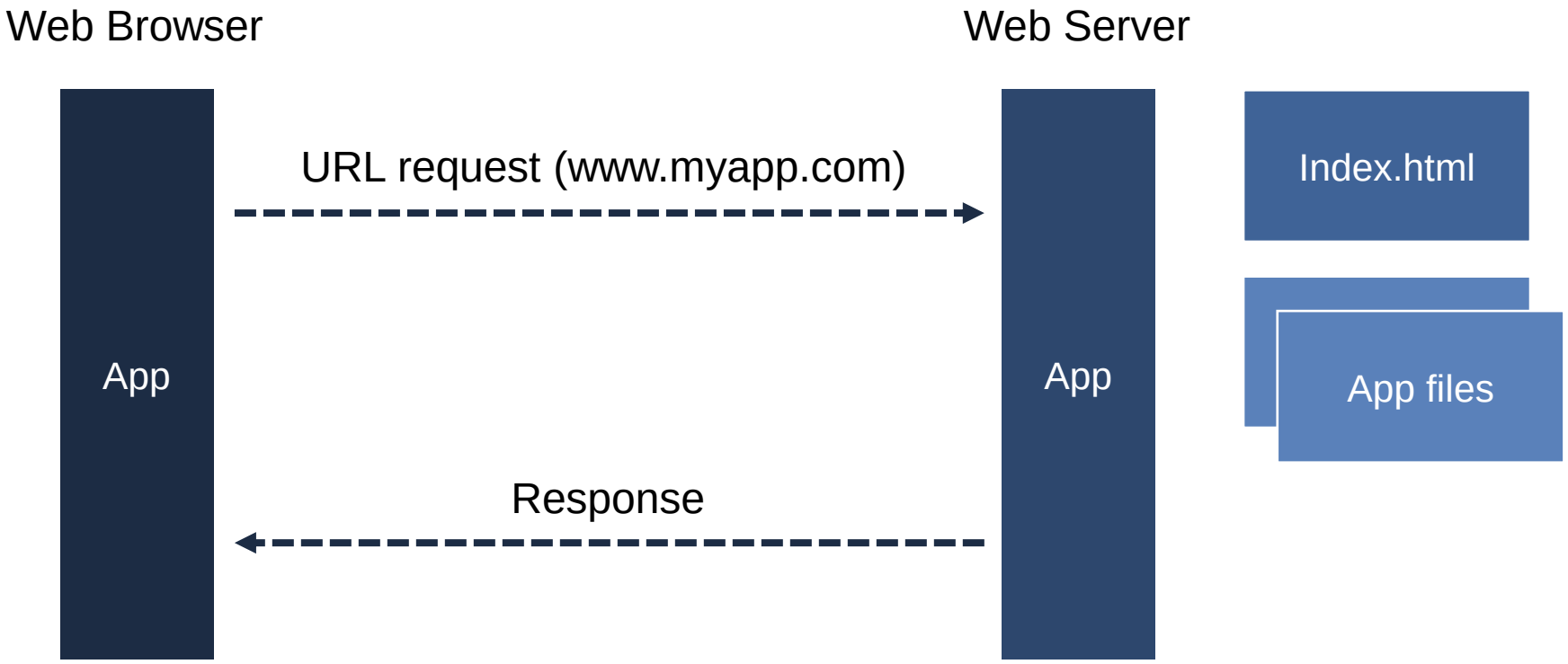
Carga de componentes



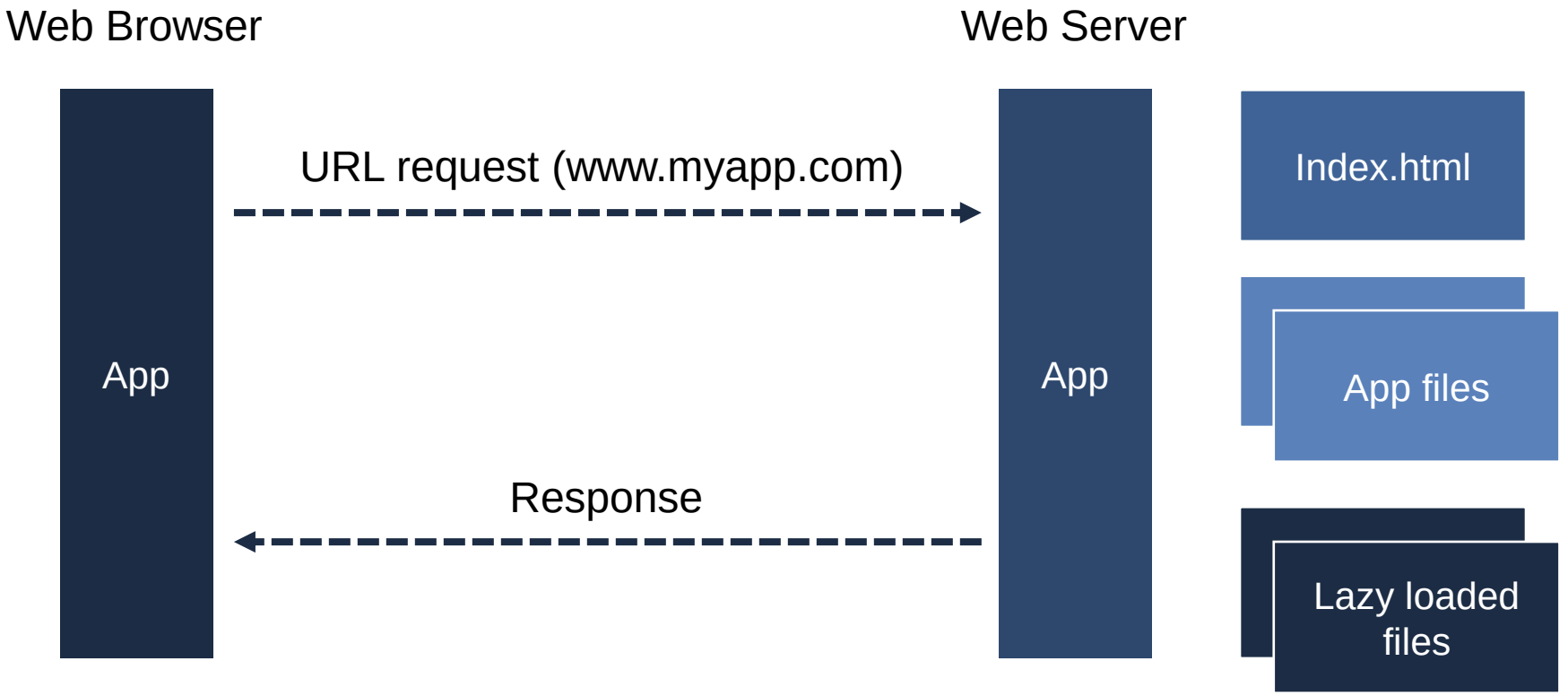
Carga de componentes



Carga de componentes



Carga de componentes



GALAXY TRAINING

Recomendaciones para el uso de Lazy loading

Usar en Feature Module

Rutas agrupadas bajo un mismo padre

No importado en otro módulo

GALAXY TRAINING

Lazy loading - modules

app-routing.module.ts

```
RouterModule.forRoot([
  ...
  {
     path: 'authors',
     loadChildren: () =>
         import('./authors/authors.module')
             .then(m => m.AuthorsModule)
  },
  ...
])
```

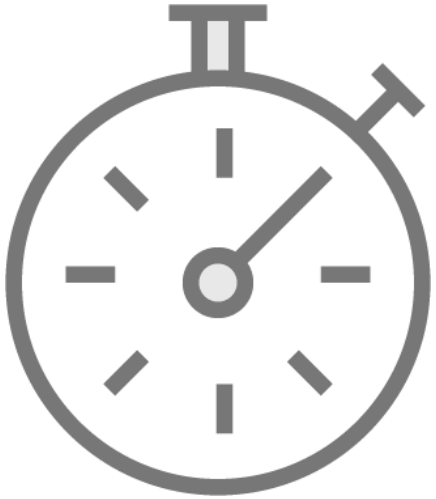
GALAXY TRAINING

Lazy loading - Standalone

app.routes.ts

```
export const APP_ROUTES: Routes = [
 ...

{ path: 'libary', loadChildren: () =>
import('../../features/library/routes/library.routes').then(m => m.LIBRARY_ROUTES) },

 { path: 'security', loadChildren: () =>
import('../../security/routes/security.routes').then(m => m.SECURITY_ROUTES) },

 ...
];
```

Lazy loading

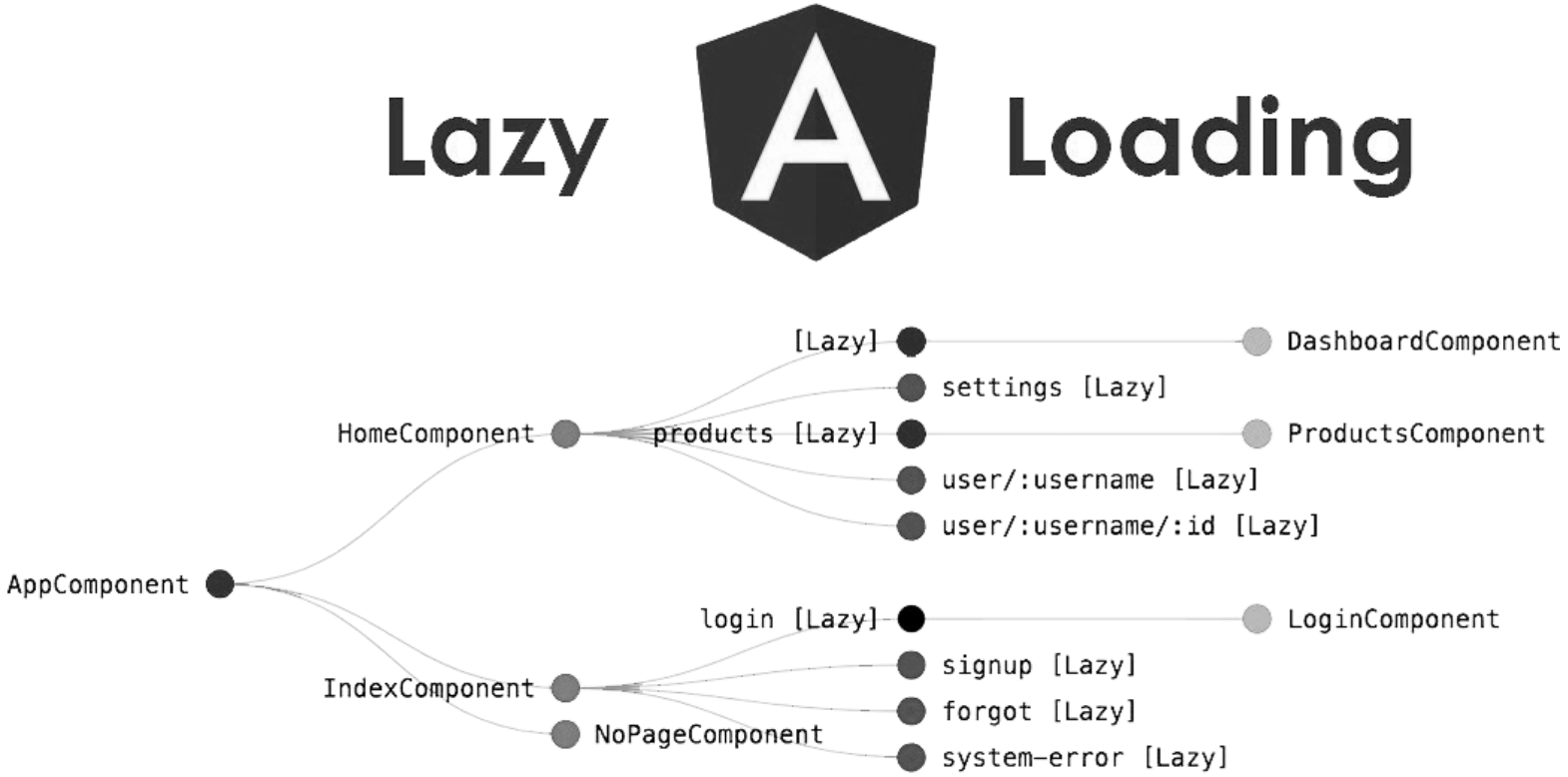


Retrasar la descarga hasta que sea necesario

Acelerar el tiempo de carga inicial

Lazy Loading de componentes

DEMO



GALAXY TRAINING

02

Protección de rutas a través de Guards

Protección de rutas con guards

canActivate

- Proteger la navegación a una ruta

canActivateChild

- Proteger la navegación a una ruta secundaria

canDeactivate

- Proteger la navegación lejos de una ruta

canLoad

- Impedir el enrutamiento asincrónico

resolver

- Cargar datos antes de activar una ruta

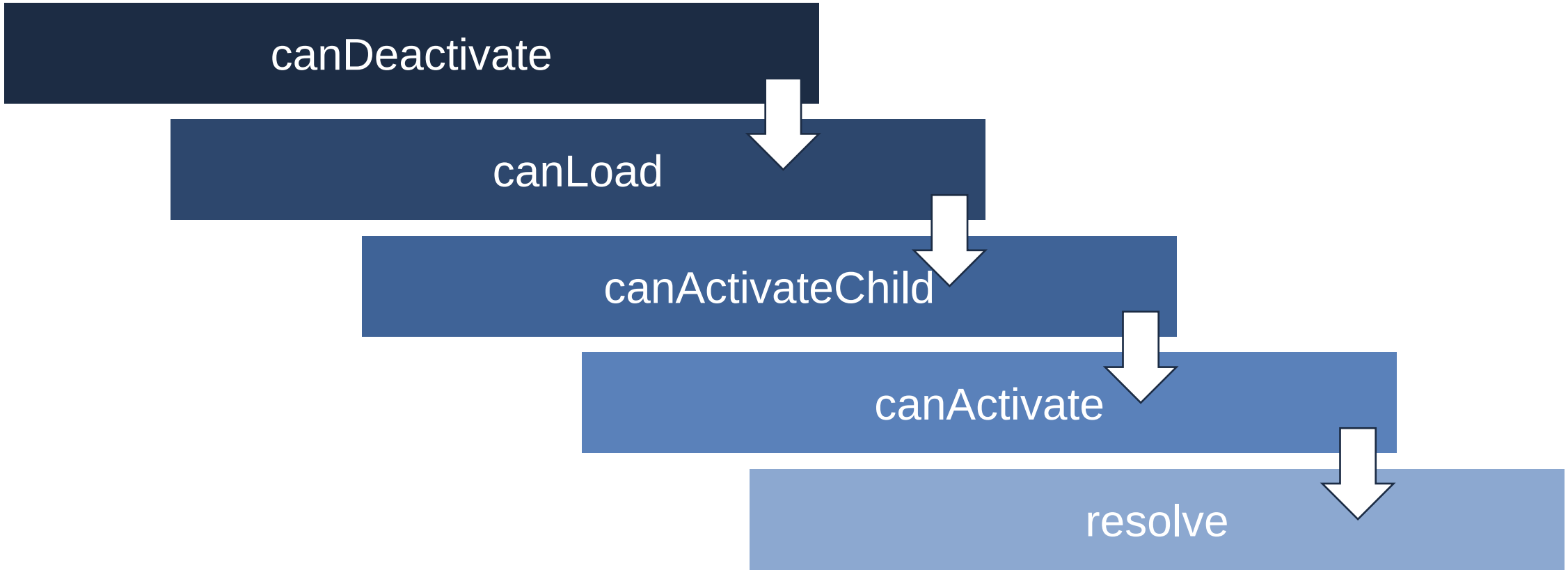
Uso de Route Guards

Limitar el acceso a una ruta

Avisar antes de abandonar una ruta

Recuperar datos antes de acceder a una ruta

Procesamiento de guards



GALAXY TRAINING

Creación de un guard

auth.guard.ts

```
import { inject } from '@angular/core';
import { CanActivateFn, Router } from '@angular/router';
import { AuthService } from '../services/auth.service';

export const authGuard: CanActivateFn = (route, state) => {
  const router = inject(Router);
  const authService = inject(AuthService);

  if (authService.isAuthenticated) {
    return true;
  }

  authService.redirectUrl = state.url;
  return router.parseUrl('/security/login');
};
```

GALAXY TRAINING

Protegiendo una ruta

Angular Module

```
...
import { authGuard } from "../../../../core/guards/auth.guard";
...

{
  path: '', component: LibraryShellComponent, canActivate: [authGuard],
}
```

canActivate Guard

Criterios de comprobación antes de activar una ruta

Comúnmente utilizado para:

- Limitar el acceso de ruta a usuarios específicos
- Asegúrese de que se cumplen los requisitos previos

Se llama cuando la URL cambia a la ruta

GALAXY TRAINING

Compartir datos

Route parameters

```
canActivate(route: ActivatedRouteSnapshot,
                state: RouterStateSnapshot): boolean {
    console.log(route.paramMap.get('authorId'));
    ...
}
```

Route data

```
canActivate(route: ActivatedRouteSnapshot,
                state: RouterStateSnapshot): boolean {
    console.log(route.data['author']); // undefined
    ...
}
```

Service property

```
export class AuthService {
     currentUser: User;
     redirectUrl: string;
}
```

canActivateChild Guard

Comprueba los criterios antes de activar una ruta secundaria

Comúnmente utilizado para:

- Limitar el acceso a las rutas secundarias
- Garantizar que se cumplan los requisitos previos para las rutas secundarias

Se llama cuando la dirección URL cambia a la ruta secundaria

canActivateChild Guard

Comprueba los criterios antes de activar una ruta secundaria

Comúnmente utilizado para:

- Limitar el acceso a las rutas secundarias
- Garantizar que se cumplan los requisitos previos para las rutas secundarias

Se llama cuando la dirección URL cambia a la ruta secundaria

canDeactivate Guard

Criterios de comprobación antes de abandonar una ruta

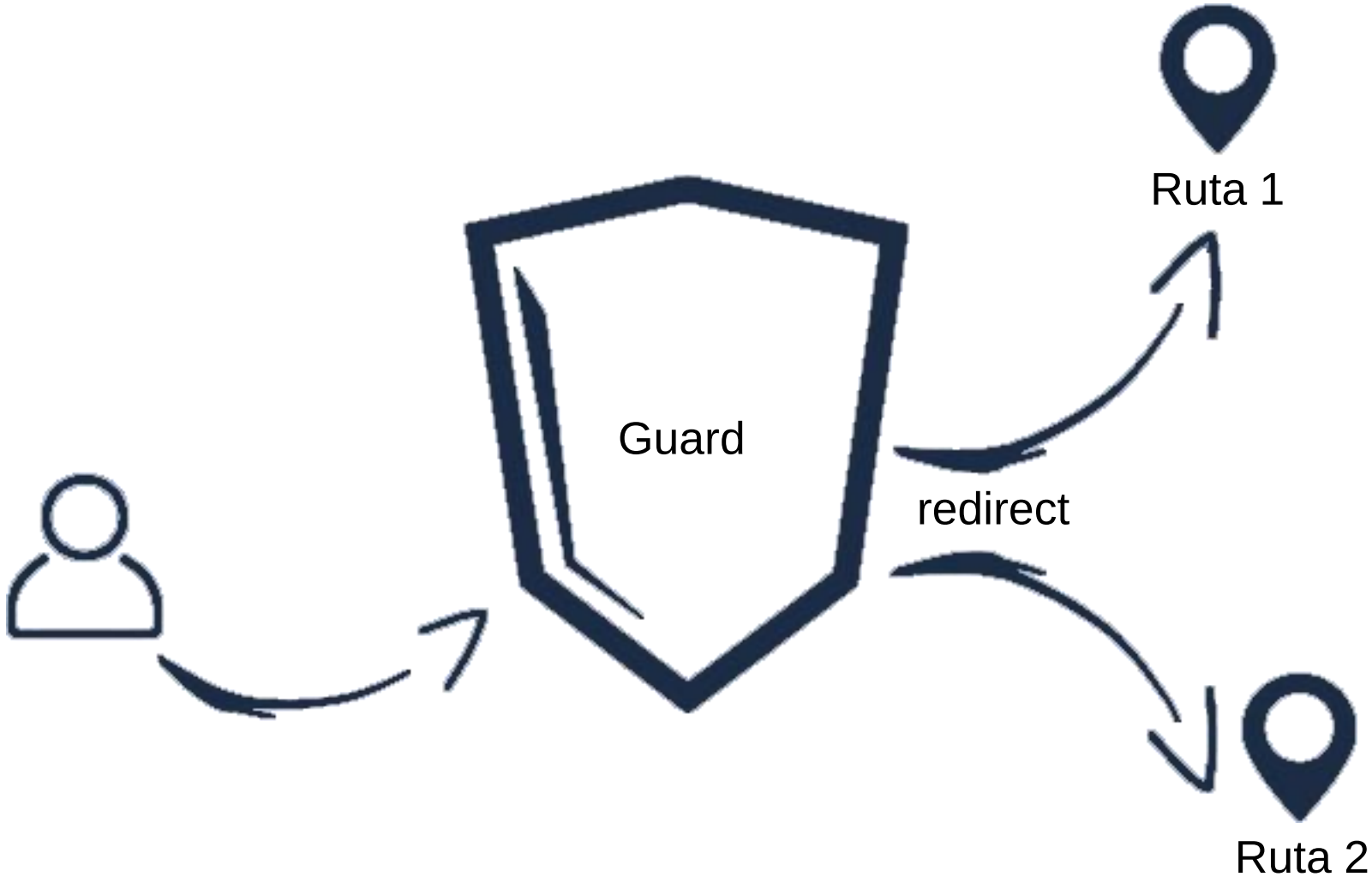
Comúnmente utilizado para:

- Comprobar si hay cambios no guardados
- Confirmar que se ha dejado una operación incompleta

Se llama cuando la dirección URL cambia a una ruta diferente

Protección de rutas a través de Guards

DEMO



GALAXY TRAINING

03

Consumo de servicios REST

RESTful CRUD

C

Create
- POST - http://localhost/api/authors
- Si tiene éxito, devuelve HTTP 201 Created

R

Read
- GET - http://localhost/api/authors o http://localhost/api/authors/1
- Si tiene éxito, devuelve HTTP 200 OK

U

Update
- PUT - http://localhost/api/authors/1
- Si tiene éxito, devuelve HTTP 204 No Content

D

Delete
- DELETE - http://localhost/api/authors/1
- Si tiene éxito, devuelve HTTP 204 No Content

GALAXY TRAINING

Suscribirse a Observables

```
getAllAuthors(): Observable<Author[]> {
    return this.http.get<Author[]>('/api/authors');
}
```

GALAXY TRAINING

Suscribirse a Observables

```
getAllAuthors(): Observable<Author[]> {
    return this.http.get<Author[]>('/api/authors');
}
```

GALAXY TRAINING

Suscribirse a Observables

```
getAllAuthors(): Observable<Author[]> {
    return this.http.get<Author[]>('/api/authors');
}
```

Suscribirse a Observables

```
getAllAuthors(): Observable<Author[]> {
    return this.http.get<Author[]>('/api/authors');
}
```

Suscribirse a Observables

```
getAllAuthors(): Observable<Author[]> {
    return this.http.get<Author[]>('/api/authors');
}


this.dataService.getAllAuthors()
    .subscribe(




   );
```

GALAXY TRAINING

Suscribirse a Observables

```
getAllAuthors(): Observable<Author[]> {
    return this.http.get<Author[]>('/api/authors');
}


this.dataService.getAllAuthors()
    .subscribe(
        (data: Author[]) => this.allAuthors = data,



    );
```

Suscribirse a Observables

```
getAllAuthors(): Observable<Author[]> {
    return this.http.get<Author[]>('/api/authors');
}


this.dataService.getAllAuthors()
    .subscribe(
        (data: Author[]) => this.allAuthors = data,
        (err: any) => console.log(err),


    );
```

Suscribirse a Observables

```
getAllAuthors(): Observable<Author[]> {
    return this.http.get<Author[]>('/api/authors');
}


this.dataService.getAllAuthors()
    .subscribe(
        (data: Author[]) => this.allAuthors = data,
        (err: any) => console.log(err),
        () => console.log('All done getting authors.')
    );
```

Suscribirse a Observables

```
getAllAuthors(): Observable<Author[]> {
    return this.http.get<Author[]>('/api/authors');
}


this.dataService.getAllAuthors()
    .subscribe(
        (data: Author[]) => this.allAuthors = data,
        (err: any) => console.log(err),
        () => console.log('All done getting authors.')
    );
```

Suscribirse a Observables

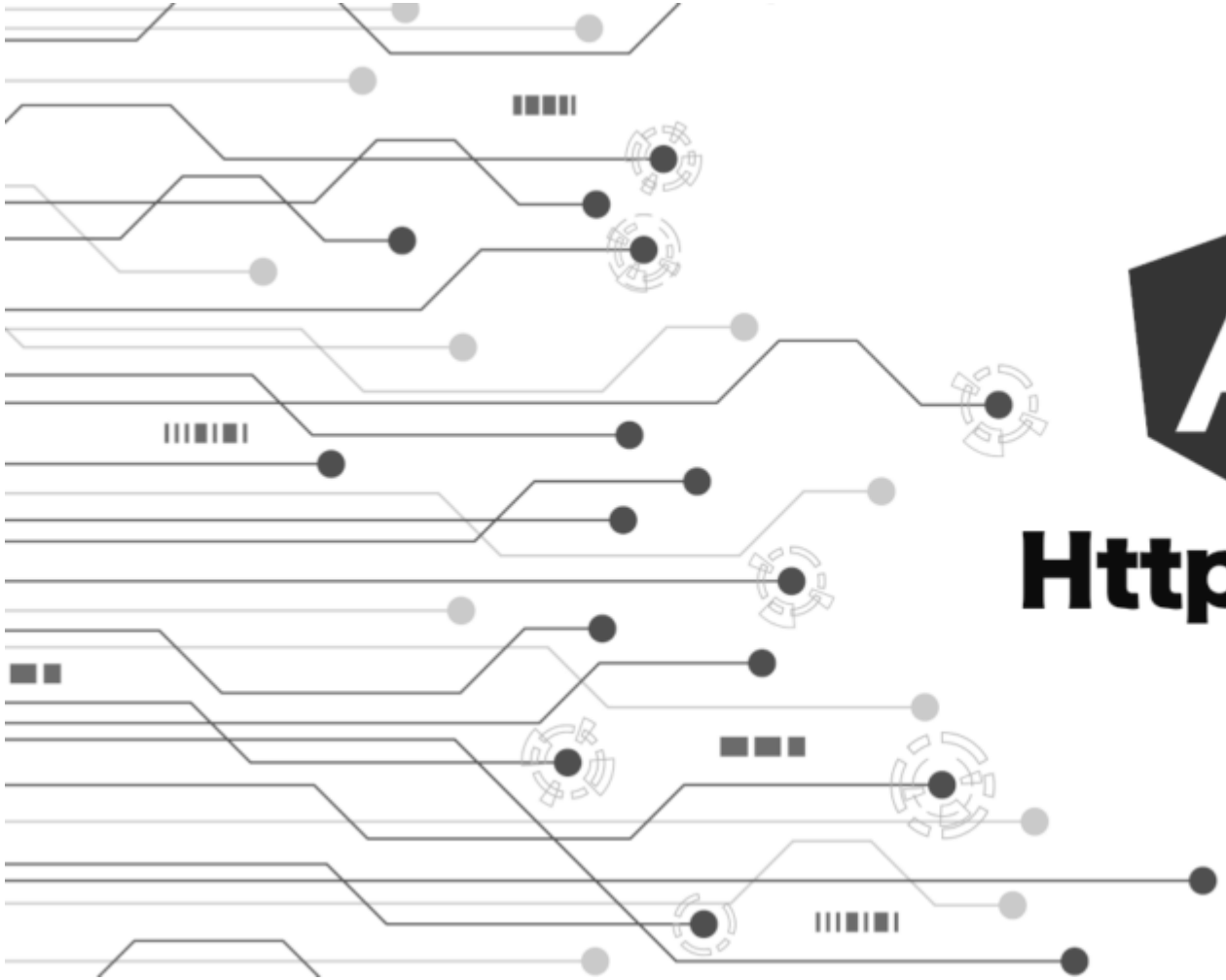
```
getAllAuthors(): Observable<Author[]> {
    return this.http.get<Author[]>('/api/authors');
}


this.dataService.getAllAuthors()
    .subscribe(
        (data: Author[]) => this.allAuthors = data,
        (err: any) => console.log(err),
        () => console.log('All done getting authors.')
    );
```

Promise

```
 public async getAuthors(): Promise<Author[]> {
  ...
  const response = await firstValueFrom(this.http.get<AuthorResponse[]>(url, { headers })
  .pipe(
    catchError(this.handleError)
  ));
  ...
}

export class AuthorRepositoryImpl implements AuthorRepository {
  ...
  public getAuthors(): Promise<Author[]> {
    return this.authorAgentService.getAuthors();
  }
}

export class GetAuthorsUseCase{
  ...
  public async execute(): Promise<AuthorModel[]> {
    const authors = await this.authorRepository.getAuthors();
    return applicationMapper.mapArray(authors,Author, AuthorModel);
  }
}

const authors = getAuthorsUseCase.execute();
```

Consumo de servicios REST

DEMO



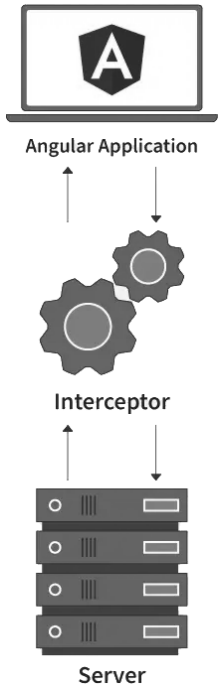
HttpClient

GALAXY
TRAINING

04

Creando Interceptores

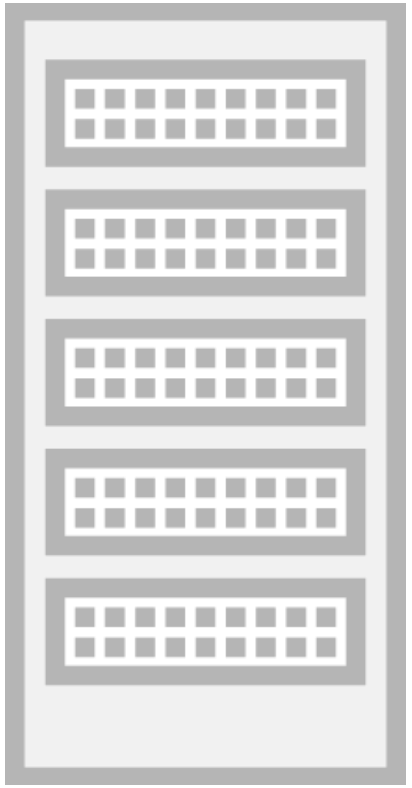
¿Que son interceptores?



- Aplicado a servicios
- Implementa la interfaz HttpInterceptor
- Manipula la solicitud HTTP antes de enviarla al servidor
- Manipula las respuestas HTTP antes de que se devuelvan a su aplicación

Interceptando requests y responses

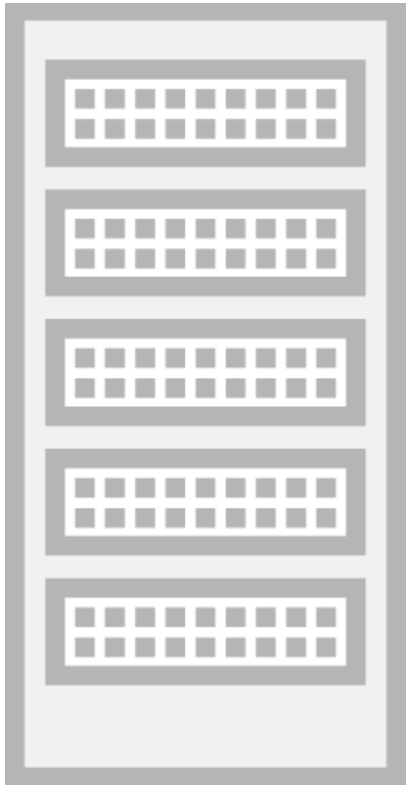
Interceptando requests y responses



Server

Interceptando requests y responses

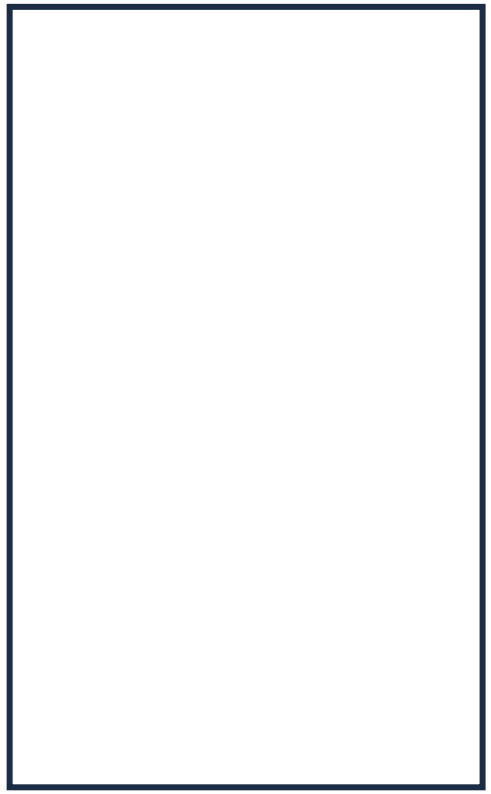
Client



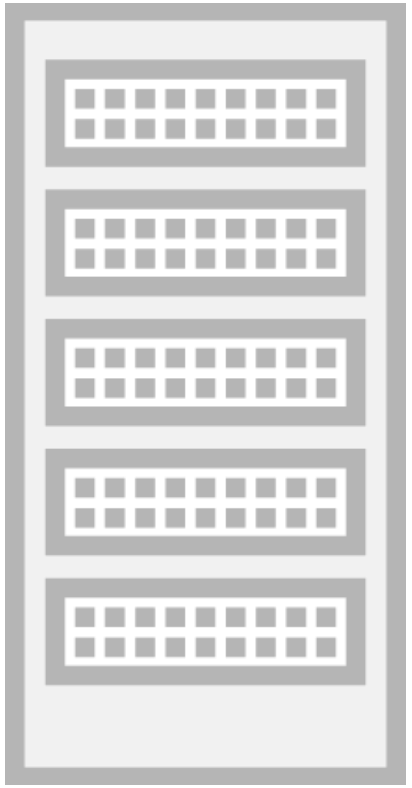
Server

Interceptando requests y responses

Client

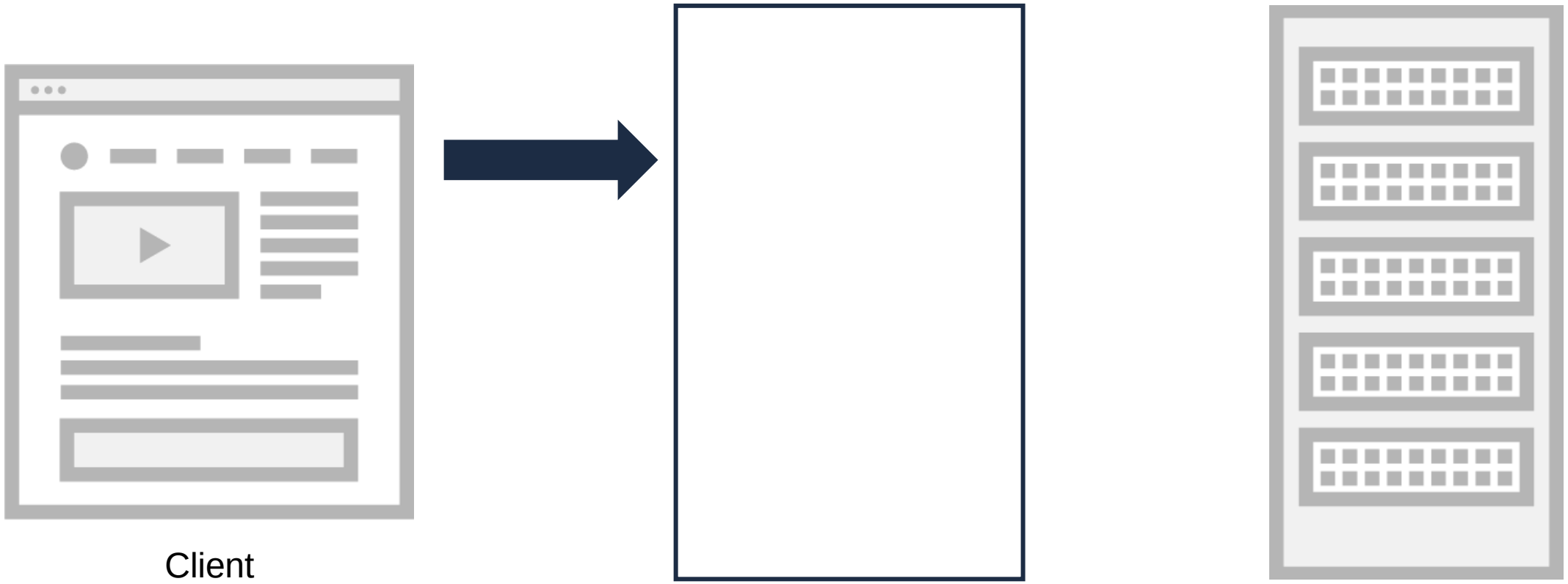


Interceptor

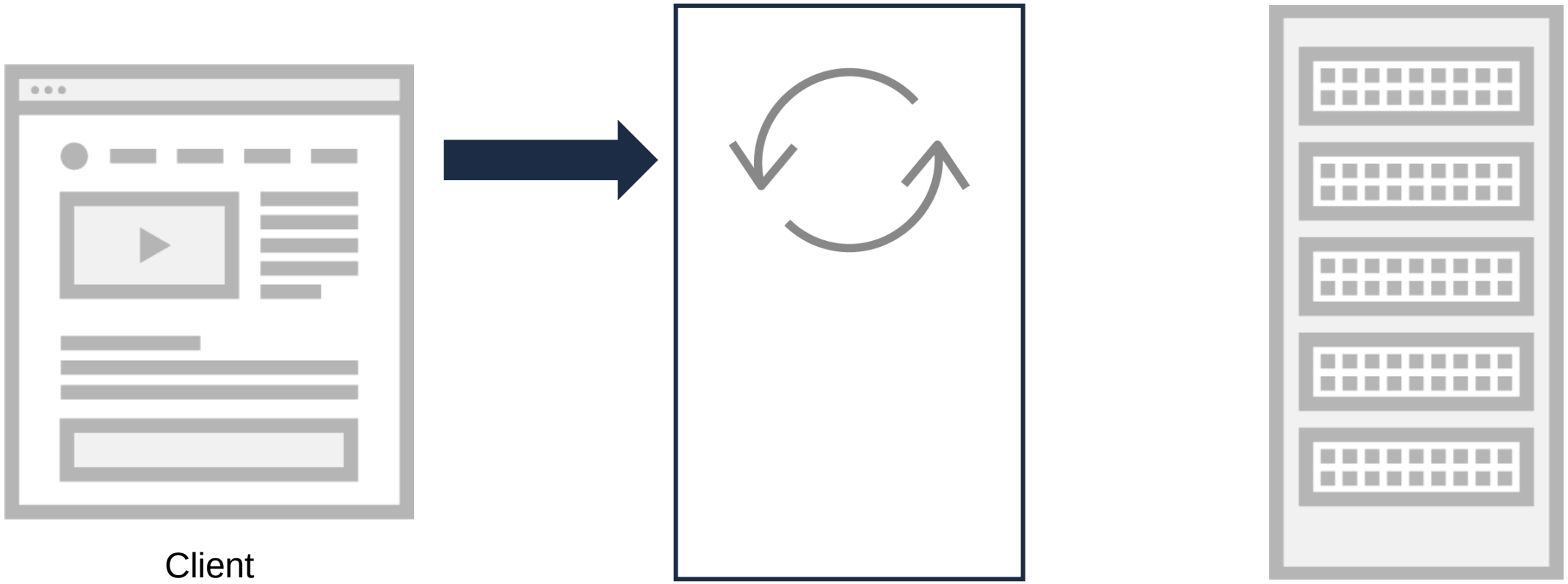


Server

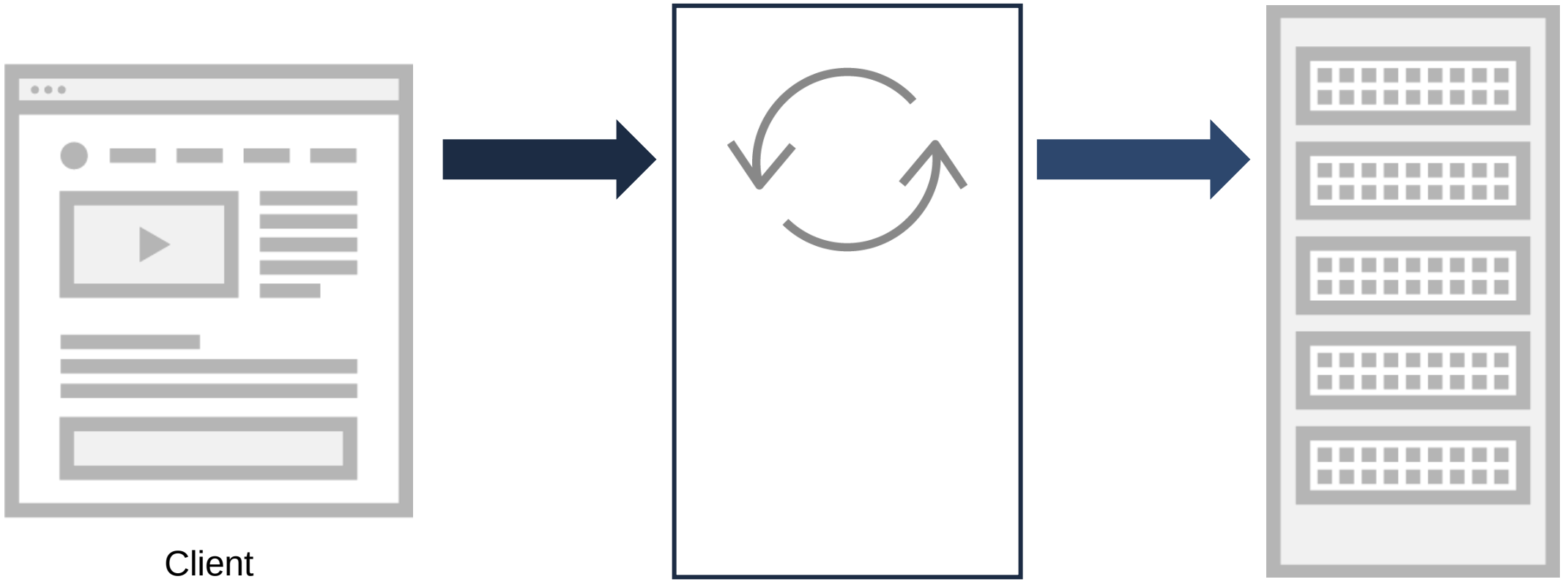
Interceptando requests y responses



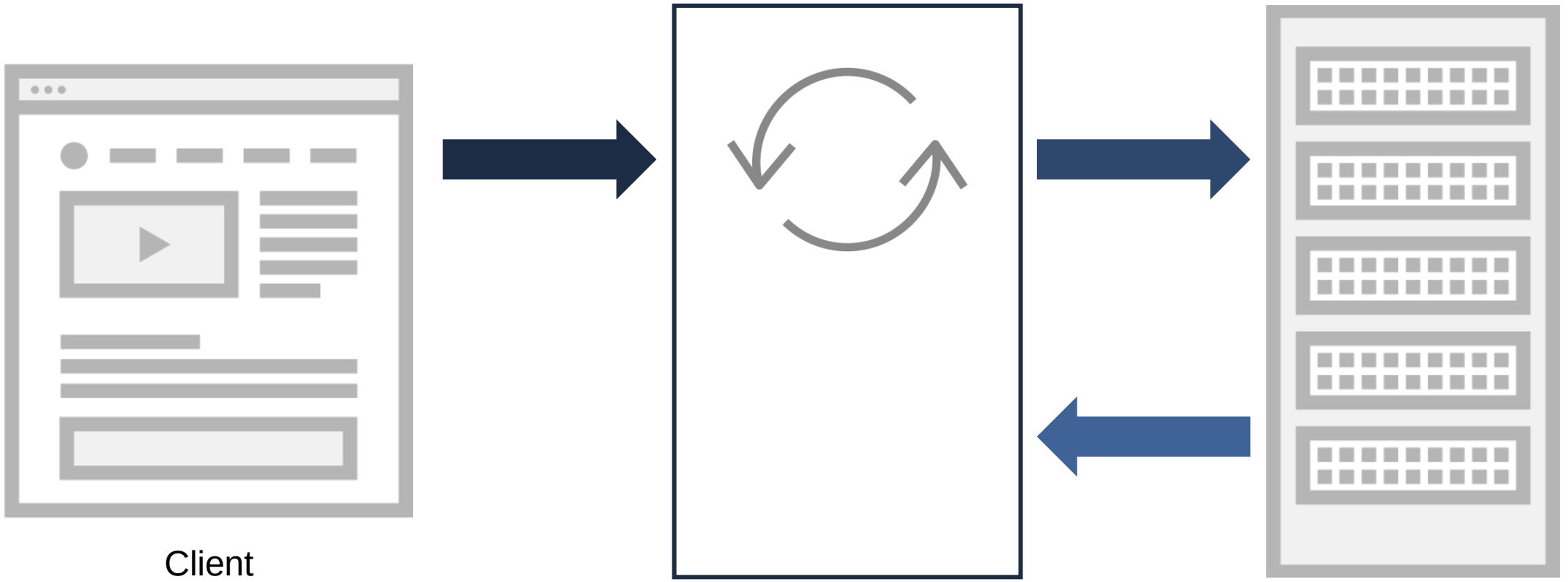
Interceptando requests y responses



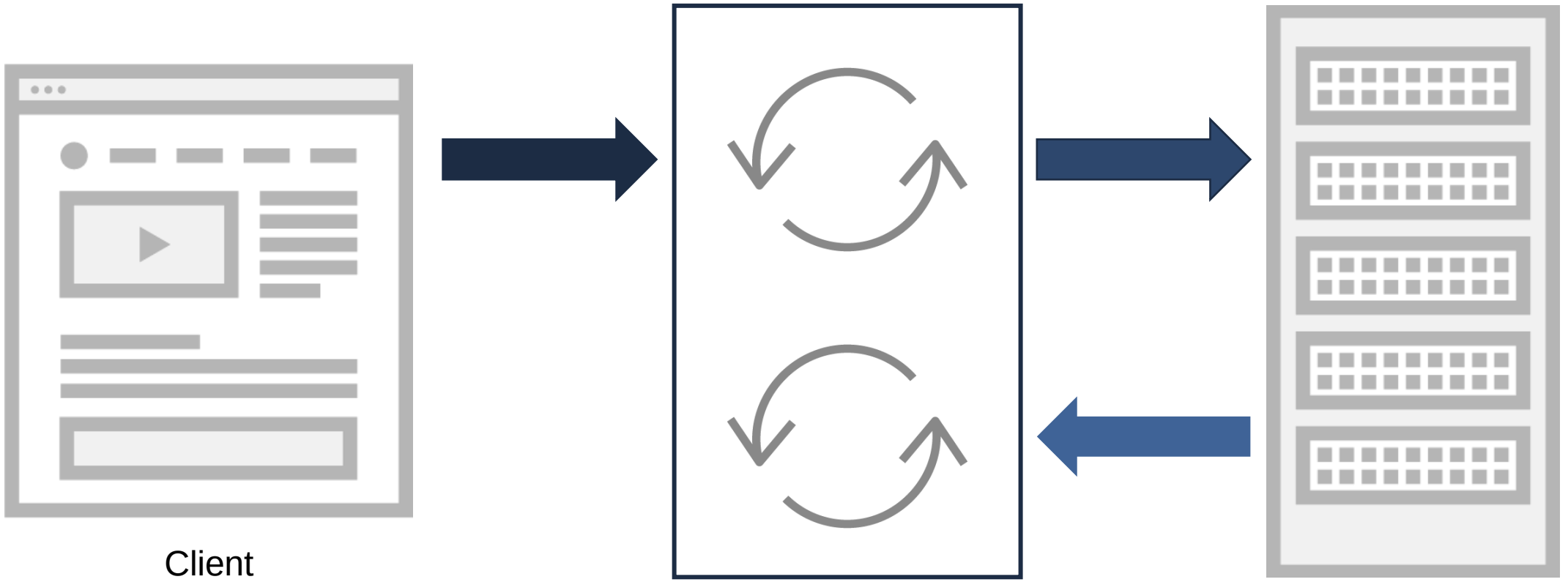
Interceptando requests y responses



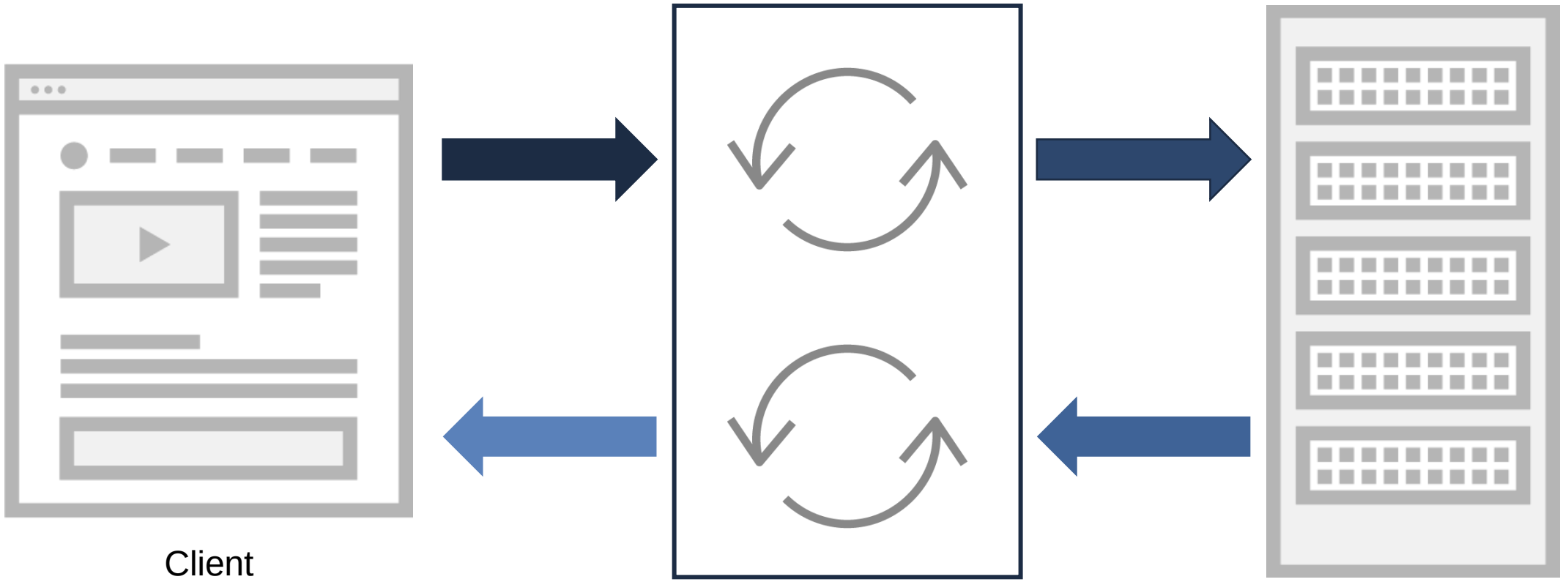
Interceptando requests y responses



Interceptando requests y responses



Interceptando requests y responses



GALAXY TRAINING

Uso de interceptores

- Agregar headers a todos los requests
- Logging
- Informes de eventos en progreso
- Caching del lado del cliente

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
```

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
```

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      const modifiedRequest = req.clone();
```

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      const modifiedRequest = req.clone();
```

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      const modifiedRequest = req.clone();
      // change modifiedRequest here
```

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      const modifiedRequest = req.clone();
      // change modifiedRequest here
      return next.handle(modifiedRequest)
```

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      const modifiedReq = req.clone();
      // change modifiedRequest here
      return next.handle(modifiedRequest)
```

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      const modifiedRequest = req.clone();
      // change modifiedRequest here
      return next.handle(modifiedRequest)
```

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      const modifiedRequest = req.clone();
      // change modifiedRequest here
      return next.handle(modifiedRequest)
```

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      const modifiedRequest = req.clone();
      // change modifiedRequest here
      return next.handle(modifiedRequest)
```

GALAXY TRAINING

Definiendo un interceptor

```
export class FirstInterceptor implements HttpInterceptor {
   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      const modifiedRequest = req.clone();
      // change modifiedRequest here
      return next.handle(modifiedRequest)
         .pipe(
            .tap(event => {
               if (event instanceof HttpResponse) {
                  // modify the HttpResponse here
               }
            })
         );
   }
}
```

GALAXY TRAINING

Registrando interceptor

```
@NgModule({
   imports: [],
   declarations: [],
   providers: [




   ]
})
export class AppModule { }
```

GALAXY TRAINING

Registrando interceptor

```
@NgModule({
   imports: [],
   declarations: [],
   providers: [
      { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },


   ]
})
export class AppModule { }
```

GALAXY TRAINING

Registrando interceptor

```
@NgModule({
   imports: [],
   declarations: [],
   providers: [
      { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },
      { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
   ]
})
export class AppModule { }
```

GALAXY TRAINING

Registrando interceptor

```
@NgModule({
   imports: [],
   declarations: [],
   providers: [
      { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },
      { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
   ]
})
export class AppModule { }
```

Registrando interceptor

```
@NgModule({
   imports: [],
   declarations: [],
   providers: [
      { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },
      { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
   ]
})
export class AppModule { }
```

GALAXY TRAINING

Registrando interceptor

```
@NgModule({
   imports: [],
   declarations: [],
   providers: [
      { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },
      { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
   ]
})
export class AppModule { }
```

GALAXY TRAINING

Registrando interceptor

```
@NgModule({
   imports: [],
   declarations: [],
   providers: [
      { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },
      { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
   ]
})
export class AppModule { }
```

Definiendo un interceptor - Standalone

```
export const authInterceptor: HttpInterceptorFn = (req, next) => {
  const configService = inject(ConfigService);
  const authService = inject(AuthService);

  if (req.url.includes(configService.getValue('AUTHORS_URL'))) {
    req = req.clone({
      setHeaders: {
        Authorization: `Bearer ${authService.appUserAuth.access_token}`
      }
    });
  }

  return next(req);
};
```

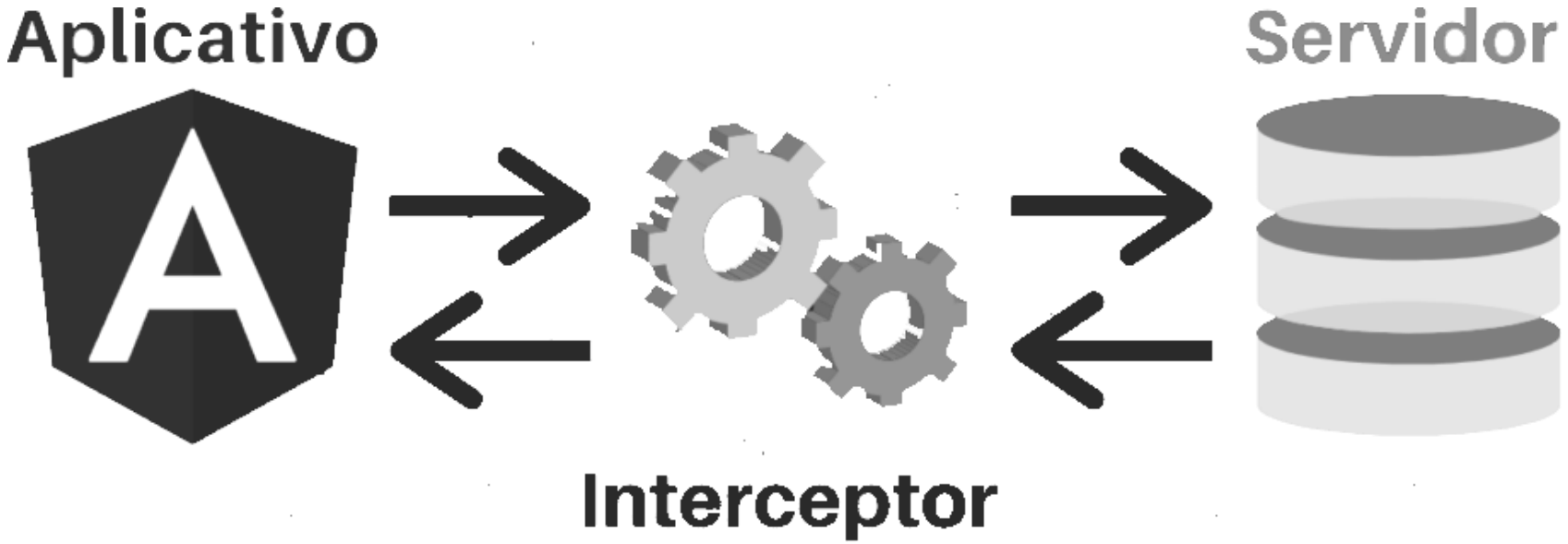
GALAXY TRAINING

Registrando interceptor - Standalone

```
import { provideHttpClient, withFetch, withInterceptors } from '@angular/common/http';
...

export const appConfig: ApplicationConfig = {
  providers: [
     ...
     provideHttpClient(withFetch(),withInterceptors([authInterceptor])),
     ...
  ]
};
```

Creando Interceptores

DEMO



GALAXY
TRAINING

Integración con Keycloak

DEMO

GALAXY
TRAINING

GRACIAS

POR SU PREFERENCIA

GALAXY
TRAINING