O que é um componente?

Arquitetura de componentes

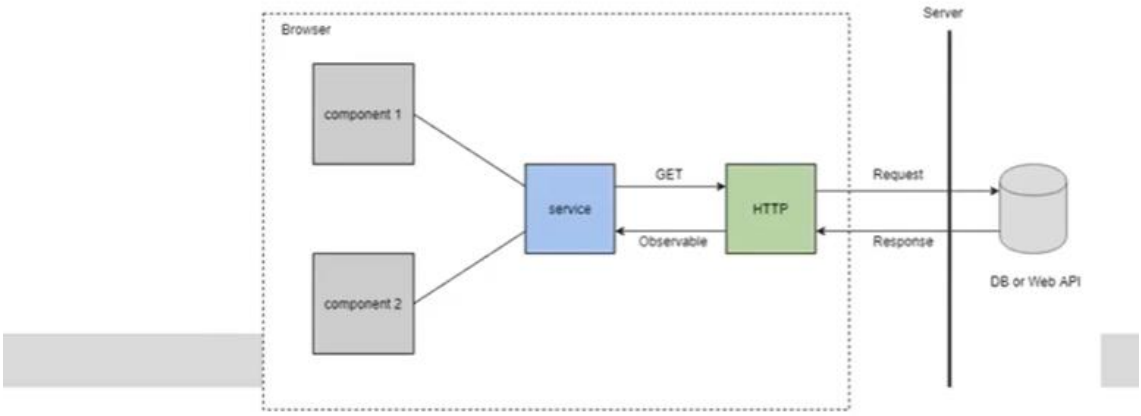
A arquitetura de componentes baseia-se na construção de **componentes independentes, substituíveis** e **modulares** que auxiliem no **gerenciamento da complexidade** e **encorajem** a **reutilização**.

Seus benefícios incluem:

- Escalabilidade
- Manutenção
- Performance

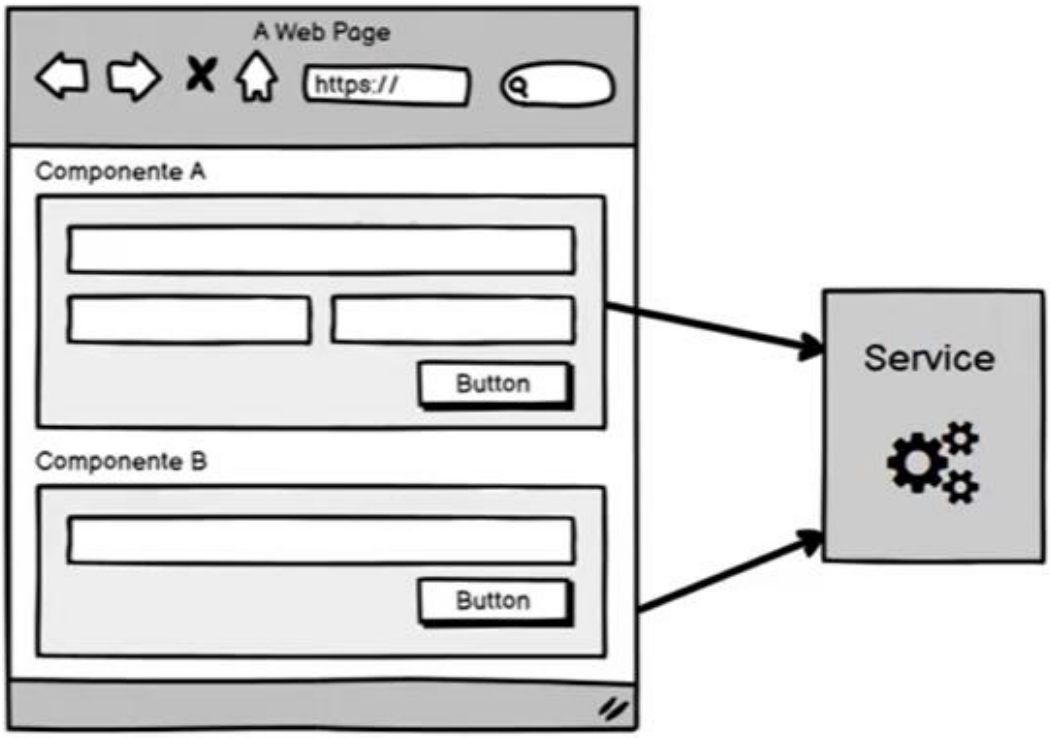
Serviços

- Responsáveis por **organizar** e **compartilhar** lógica de negócios
- **Reutilizáveis** entre diferentes componentes de um aplicação
- **Mandatorios** para uma **arquitetura modular** e **reutilizável**



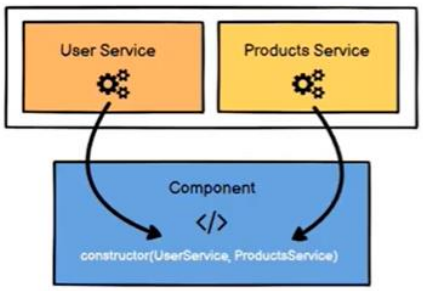
Componente- interface do usuário

Service – Trata da regra de negócio



Injeção de dependência

Todo serviço é uma **dependência** que precisa ser **instanciada dentro do componente** para ser utilizada pelo mesmo. No angular, o componente pede para aplicação quais dependências ele precisa e então as injeta dentro de si.

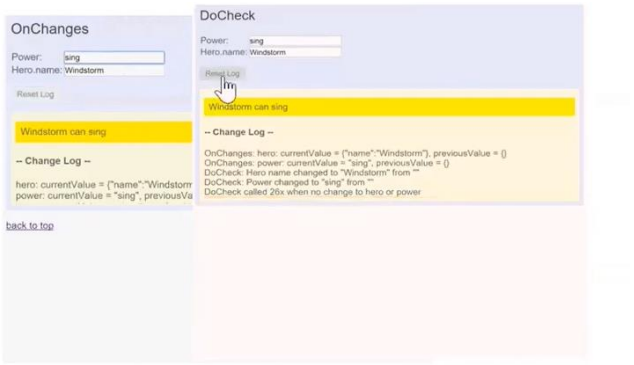



```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  constructor(
    private userService: UserService,
    private productsService: ProductsService
  ){}
}
```

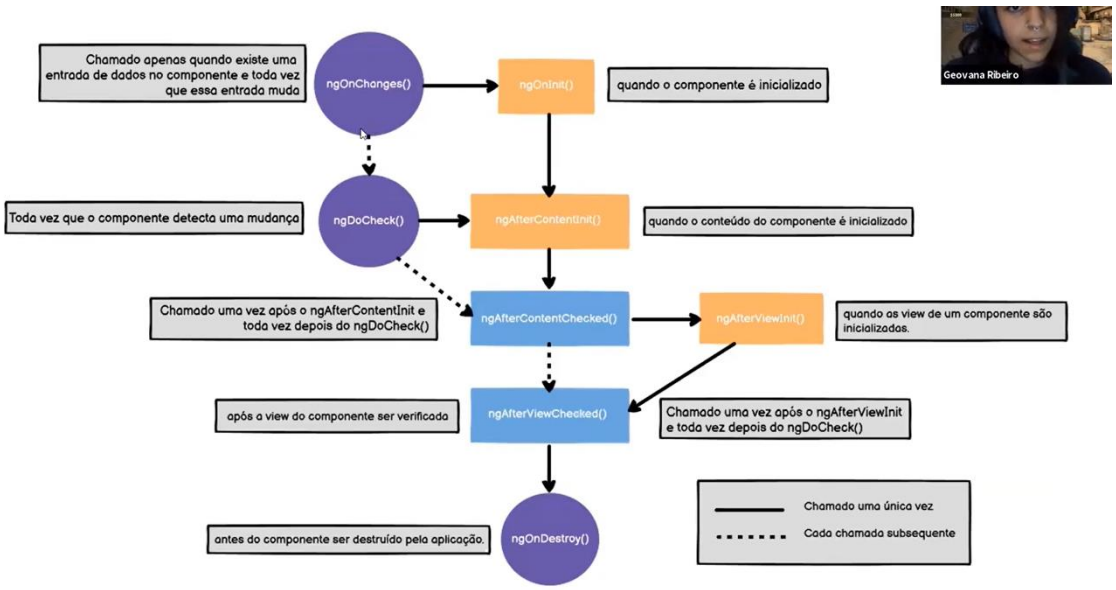
Ciclo de vida do Componente

Todo componente possui seu **ciclo de vida** (normalmente chamado de **lifecycle hooks**), que começa assim que o Angular o instancializa na aplicação e através deles é possível executar diferentes lógicas nos vários estágios de um componente.

```
@Component({
  ...
})
export class AppComponent implements OnInit, OnDestroy {

  constructor() {}

  ngOnInit() {
    console.log(`Component it's created! \o/`);
  }

  ngOnDestroy() {
    console.log(`Component has destroyed! =(`);
  }
}
```



⚠ **Atenção:** Use com sabedoria para não comprometer a performance de sua aplicação!



Constructor vs ngOnInit

Constructor

- Deve ser utilizado apenas para inicializar serviços injetados via DI (injeção de dependência)

ngOnInit

- Deve ser utilizado para todo tipo de lógica que o componente precisar executar após ter sido criado.