Angular
Padrões de Arquitetura

Objetivos de uma boa arquitetura Angular

- Design Modular
- Alta performance
- Utilização de padrão Smart/Dumb components
- Boas abstrações entre as camadas da aplicação
- Lazy-load de módulos
- Fluxo de dados unidirecional
- Gerenciamento de estado reativo

Desafios de Escalabilidade no Front-End

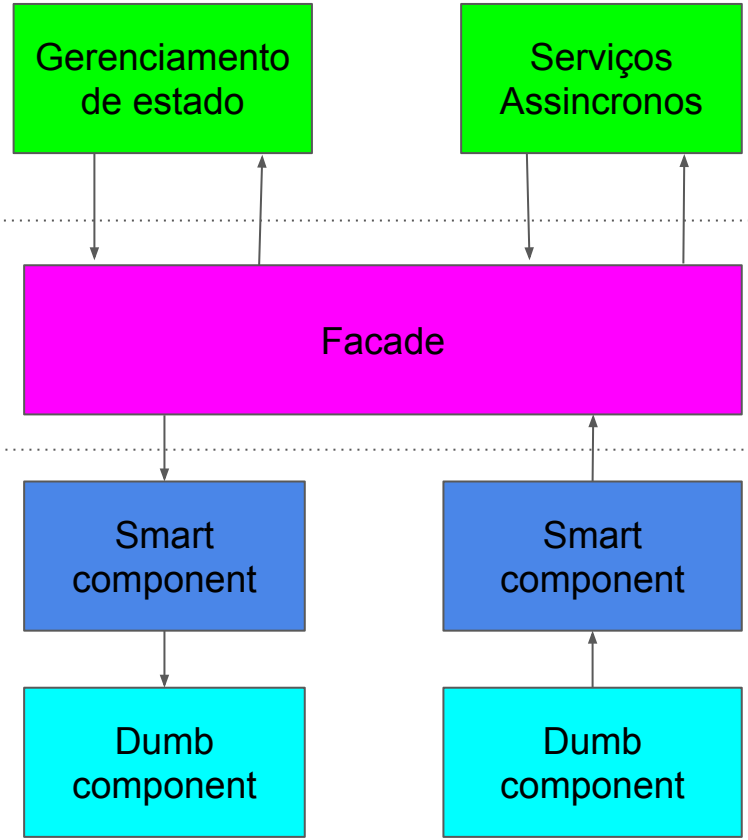
- Front-end tem abrigado cada vez mais responsabilidades que antes eram do back-end
- Tempo de carregamento inicial
- Performance da aplicação
- Velocidade de desenvolvimento de novas funcionalidades
- Quantidade de dados carregada

Problemas em aplicações sem arquitetura bem definida

- Componentes podem acabar dependendo de muitos serviços
- Violação do princípio de responsabilidade única
- Fluxo de dados complexo, não intuitivo
- Problemas de performance
- Arquivos muito grandes
- Dificuldade para implementar testes
- Dificuldade de onboarding de novos desenvolvedores
- Dificuldade para implementar novas funcionalidades

Camadas da aplicação

- Core
  - Gerenciamento de estado
  - Serviços assincronos (Http, Websockets)
- Abstração
  - Facade
- Apresentação (View)
  - Componentes
  - Diretivas



Camada de Apresentação

- Apresentação de dados
- Delegação de eventos/ações para a camada Core, por meio da camada de Abstração
- **Não** sabe como as operações do usuário devem ser manipuladas

Camada de Apresentação - Smart Components

- Tem a facade e/ou outros serviços injetados
- Passa os dados para os dumb components
- Reaje a eventos dos dumb components
- Geralmente são componentes roteados
- Se comunica com a camada Core por meio da facade

Camada de Apresentação - Dumb Components

- Recebe seus dados por meio de inputs
- Emite eventos via outputs
- Provavelmente pode usar a estratégia de change-detection OnPush

Camada de Abstração

- Expõe métodos para a camada de apresentação consumir, que delegam para a camada Core a lógica de execução
- Expõe streams de dados (Observables) para a interface, derivando eles da camada de gerenciamento de estado.
- Pode combinar e transformar dados da camada de estado
- Pode cachear dados da camada de API
- Pode gerenciar a estratégia de sincronização de dados (otimista vs pessimista)

Camada de Abstração - Fluxo de dados e Reatividade

- Dados fluem das camadas de cima (Core/gerenciamento de estado) e alimentam as camadas de abstração e apresentação por meio de observables
- Eventos gerados na camada de apresentação sobem para a camada de abstração e então para a camada core, onde são executados e geram novos dados, que então são propagados de volta para a camada de abstração e apresentação.

Camada Core

- Lógica de negócio
- Comunicação com serviços externos
- Gerenciamento do estado da aplicação

Fluxo de dados unidirecional