ToDo App com arquitetura MVC, MVP e MVVM com backend REST e reativo

Integrantes: Yasmim Rodrigues, Miller Monteiro, Isaque Oliveira e Kevin Valentim.

1. Implementação lógica de cada arquitetura

Ao utilizar diferentes arquiteturas de frontend, temos uma drástica mudança na organização do código e no acoplamento entre cada camada de software, mesmo mantendo inalterado o funcionamento final do aplicativo. Em nosso caso, podemos observar que, nas arquiteturas MVC, MVP e MVVM, a maior parte da lógica se concentra nos componentes responsáveis por coordenar o fluxo da aplicação e intermediar a comunicação entre a interface de usuário e a camada de dados.

MVC – Na arquitetura Model-View-Controller, como esperado, a maior parte da lógica se concentra no Controller, sendo este o responsável por coordenar os casos de uso, decidir acessos ao backend e atualizar o estado local. Isto é, o Controller aqui tem baixo acoplamento com a View e com o Model, permitindo que tanto a View quanto o Model recebam os dados corretamente, independente do backend usado ser REST ou reativo.

MVP – Já na arquitetura Model-View-Presenter, é no Presenter que a maior parte da lógica reside. Aqui, o Presenter atua como mediador direto entre a View e o Model, não sendo necessário que a View seja observadora do Model, mantendo uma View passiva que apenas mostra a interface e coleta os dados do usuário. Essa separação reduz ainda mais o acoplamento entre as camadas e torna a lógica de apresentação mais reutilizável, sendo melhor para projetos mais complexos que teriam o Controller como gargalo.

MVVM – Na arquitetura Model-View-ViewModel, a maior parte da lógica reside no ViewModel, responsável por expor estados e comportamentos da aplicação de forma reativa. O ViewModel não conhece diretamente a View, comunicando-se com ela por meio de dados observáveis. Dessa forma, alterações no estado do Model ou respostas do backend são automaticamente refletidas na interface de usuário, permitindo que a View seja declarativa, tornando-a mais legível.

2. Arquitetura ideal para backend reativo

Para a implementação do backend reativo, optamos por utilizar um hub SSE para manter a conexão aberta com os múltiplos clientes e notificá-los de todas as mudanças via broadcast. Dessa forma, independente da arquitetura escolhida para o frontend, todas as arquiteturas podiam consumir um backend reativo. Entretanto, a complexidade de integração foi diferente para cada uma das arquiteturas.

A arquitetura MVVM, aquela que mais se adequa ao fluxo de desenvolvimento em React, também é a mais simples de integrar ao backend reativo. Isto se deve ao fato de o ViewModel ser capaz de comunicar-se com a View para expor os estados da aplicação de forma totalmente reativa, sendo esse a fonte da verdade que se conecta ao backend por meio de repositórios, eliminando a necessidade de sincronização adicional entre as camadas e permitindo uma integração mais direta com a lógica de push.

3. Adaptação do frontend para REST e reativo

Apesar das tentativas de manter a arquitetura mais desacoplada entre frontend e backend, nenhum dos modelos arquiteturais utilizados no frontend é capaz de abstrair totalmente o backend. Por isso, fizeram-se necessárias diversas mudanças ao trocar a implementação em REST para reativa.

Com o intuito de manter um único frontend que se integrasse a ambos os backends, utilizamos uma lógica adicional de controle nos diversos componentes das três arquiteturas. Dessa forma, foi possível, com apenas um clique, selecionar de qual backend os dados seriam consumidos.

Quando está conectado ao backend REST, o frontend solicita de forma explícita (pull) sempre que precisa atualizar dados. Por isso, sempre que alguma alteração é feita na View, a sincronização com o backend acontece somente após o usuário clicar em “Atualizar” ou dar refresh na página.

Para adequar o frontend de ambas as arquiteturas utilizadas para o modo reativo, foi necessária a inserção de um fluxo baseado em eventos diretamente no código, no qual os componentes se inscrevem nas notificações do backend e passam a reagir automaticamente a essas notificações.

4. Arquitetura ideal para sistemas maiores

No contexto visto no nosso projeto do ToDo App, a arquitetura mais simples de implementar foi o MVVM, já que o React é mais ajustado a esse tipo de implementação, além de permitir uma rápida integração a ambos os backends. Entretanto, para sistemas maiores, o React não irá definir o grau de complexidade das implementações, mas sim a lógica de negócio necessária para atender aos requisitos do sistema. Por isso, para cada sistema, é necessário se ponderar qual arquitetura será a mais adequada.

Um aplicativo como o ToDo App, mesmo em um sistema que tivesse milhares de usuários, ainda assim poderia manter um backend REST e uma arquitetura MVC sem grandes gargalos. Porém, para um sistema diferente, que tivesse uma lógica de negócio muito complexa, a arquitetura MVVM, que favorece maior desacoplamento e organização do código, se manteria como a arquitetura ideal.