01

Introdução ao merge de informações

Transcrição

A aplicação consegue inserir, remover e alterar os alunos, tanto em modo online quanto offline. Mas existe um detalhe que precisaremos ficar atentos: se um aluno for alterado, ele deve ser simultaneamente modificado no servidor e na aplicação. Vamos simular essa situação. No exemplo, temos o aluno **Paulo** cadastrado no servidor e na aplicação. No **servidor**, vamos alterar o nome para **Paulo Silveira**, mas na **aplicação**, colocaremos o nome de **Paulo da Silva**.

Ao atualizarmos o servidor, notaremos que ele manteve a atualização da aplicação, porque ela foi mandada por último. Isso era esperado, afinal estamos conectados... No entanto, o que acontecerá quando estivermos offline?

Fazendo a mesma simulação, porém, com o celular no modo avião. Em seguida, vamos recuperar a conexão e fazer o *swipe*. Notamos que foi mantida a versão do servidor. Isso aconteceu justamente pelo fato de não termos uma ordem que indique qual versão deve ser mantida.

Os nossos métodos de busca e sincronização são executados em paralelo, ou seja, não é possível ter certeza de qual deles terminará a execução primeiro.

Tentaremos entender tudo o que pode acontecer por debaixo dos panos para identificarmos as medidas que podemos tomar.

Definindo prioridades

Imagine que perdemos a conexão entre a aplicação com o servidor. Durante este período, o nome do aluno **Alex** foi alterado para **Alex Felipe** na aplicação, e no servidor foi alterado para **Alex Vieira**.

Quando a aplicação ficar online, quem terá prioridade? Atualmente não possuímos nenhuma regra. Existem dois casos, prioridade para o servidor ou para a aplicação Android.

Priorizando o servidor, quando a aplicação Android voltar a ficar online, ela irá solicitar todas as informações atualizadas no servidor.

Com as novas informações, a aplicação vai atualizar tudo internamente, mantendo os dados que foram definidos no servidor.

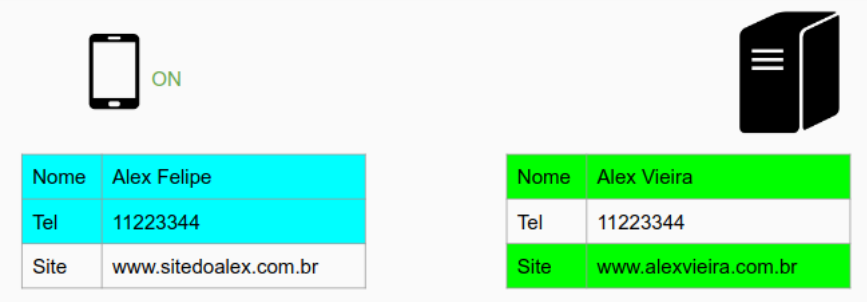
Resumindo, a aplicação vai pedir primeiro as informações do servidor, atualizar localmente para depois enviar os seus dados para o servidor.

E se a prioridade fosse do cliente? Neste caso, o cliente mandará primeiro as informações para o servidor.

O servidor vai receber e editar as suas informações, mandando as informações atualizadas de volta para a aplicação.

Além das definições de prioridades, temos o ***merge personalizado***, que nada mais é que a junção das informações. Em vez de sobrescrever as informações automaticamente, o cliente sempre vai ser perguntando sobre qual versão ele pode manter.

Porém isso não é tão simples assim, é necessário alguns cuidados. Quando alteramos alguma informação e perguntamos ao cliente o que ele quer manter, é necessário verificar todos os campos, por exemplo, nome, telefone, site e assim por diante.



Isso é algo bem difícil de ser implementado, e talvez em alguns casos se torne um esforço desnecessário. É importante que os cenários da aplicação sejam analisadas. No caso, faz mais sentido darmos prioridade para o servidor, considerando que estamos trabalhando em um sistema no qual várias aplicações estão mandando informações para o servidor.

Se uma aplicação ficar editando informações offline e simultaneamente, outras estiverem editando online, ao ser feito o envio de dados para o servidor, a aplicação sem conexão receberá as informações atualizadas - em vez de mandar as antigas, devido à questão de prioridade.

Mas existe situações nas quais seria melhor a prioridade no cliente? Sim, nos casos em que apenas a aplicação mande informações para o servidor e não tenha outros clientes editando.

Mais adiante, veremos como implementar esses passos de prioridade.