01

Padronizando a sincronização com o Firebase

Transcrição

Agora que a nossa aplicação está conseguindo dar prioridade para o servidor, cuidaremos de outro detalhe: o ***Firebase***.

O *Firebase* é uma camada a mais utilizada para que o cliente não fique pedindo manualmente as atualizações do servidor. Só que o *Firebase* funciona em paralelo com o que estamos fazendo atualmente, não respeitando a ordem que definimos.

Todos os teste que fizemos até agora, não utilizavam o *Firebase*, vamos habilitá-lo para ver como será o comportamento. No servidor, clicaremos no menu e depois, selecionaremos a opção *Firebase*. Em seguida, basta colocar a chave e clicar em "Salvar".

Novamente, ativaremos o modo avião do celular para realizarmos o teste. No servidor, colocaremos o nome do aluno como **Paulo**, e na aplicação, escreveremos **Paulo da Silva**. Para vermos o comportamento, abriremos o *Android Monitor*. Agora basta retirarmos o modo avião, porque o *Firebase* já vai mandar as informações quando detectar a conexão com nosso celular.

Antes de fazermos um *refresh* na aplicação, o *Firebase* já enviou os dados para o celular. Temos um caso no qual ficamos sem controle, que é justamente quando o *Firebase* nos envia informações.

O que precisamos fazer para ter um controle maior com o *Firebase*? Vamos olhar no código da classe `AgendaMessagingService`.

Esse código foi configurado para verificar se existe uma mensagem nova, além de salvá-la, atualizá-la e sincronizá-la com o banco de dados. Observe que ele não salva a versão desses dados, e isso é muito ruim, pois ele pode sobrescrever as informações com dados antigos. Tomaremos cuidado já que a versão dos dados importa.

O nosso desafio será fazer com que o código do *Firebase* tenha uma regra, como verificar a versão em vez de apenas sincronizá-la. O primeiro passo será fazer ele entrar no padrão de salvar a versão e depois sincronizar os alunos internamente.

Analisando a classe `AlunoSincronizador` dentro da classe `buscaAlunoCallback()`, teremos o código que faz a sincronização:

```
// ...

String versao = alunoSync.getMomentoDaUltimaModificacao();
preferences.salvaVersao(versao);

AlunoDAO dao = new AlundoDAO(context);
dao.sincroniza(alunoSync.getAlunos());
dao.close();

// ...
```

Vamos extraí-lo para um método com o atalho "Ctrl + Alt + m". Na janela de configuração do método, colocaremos a *Visibility* como `public`, e em *Name* preencheremos o nome do método como `sincroniza`.

Desta forma padronizada, garantiremos que sempre será salva uma versão. Temos agora o método `sincroniza()`:

```
// ...

public void sincroniza(AlunoSync alunoSync){

    String versao = alunoSync.getMomentoDaUltimaModificacao();

    preferences.salvaVersao(versao);

    AlunoDAO dao = new AlundoDAO(context);
    dao.sincroniza(alunoSync.getAlunos());
    dao.close();
}

// ...
```

Na classe `AgendaMessagingService`, dentro do método `converteParaAluno()`, retiraremos todo o código que trabalha com a instancia do `AlunoDAO` e, no lugar, colocaremos uma instancia de `AlunoSincronizador` passando no construtor o `AgendaMessagingService.this`. Depois, chamaremos o método `sincroniza()`, passando o `alunoSync` como argumento.

```
// ...

private void converteParaAluno(Map<String, String> mensagem) {
    String chaveDeAcesso = "alunoSync";
    if(mensagem.containsKey(chaveDeAcesso)){
        String json = mensagem.get(chaveDeAcesso);
        ObjectMapper mapper = new ObjectMapper();
        try {
            AlunoSync alunoSync = mapper.readValue(json, AlunoSync.class);
            new AlunoSincronizador(AgendaMessagingService.this).sincroniza(alunoSync);
            EventBus eventBus = EventBus.getDefault();
            eventBus.post(new AtualizaListaAlunoEvent());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

// ...
```

Agora, em vez de ficarmos preocupados em salvar as informações e a versão no banco de dados, nós delegaremos para o `AlunoSincronizador`. Mesmo que a informação venha do *Firebase*, ela já será atualizada.

Um detalhe é: o que fazer se a informação for antiga? O código irá salvar uma informação que já pegamos do servidor. Precisaremos colocar um critério com o qual verificaremos se a versão é mais antiga ou mais recente. Este será o próximo problema que iremos resolver.