José Adeljan Marinho da Silva

Residência em Robótica e Inteligência Artificial

Orientação a Objetos com Java e Android

Descrição das etapas implementadas no projeto

1. Na classe **ContasActivity**, as informações são extraídas do banco de dados usando uma *Thread*, o que implica que a operação é executada em uma *thread* separada para evitar bloqueios na interface do usuário. A lista de contas é mantida atualizada usando um Runnable, que é executado periodicamente para obter os dados atualizados.

2. Na classe **ContaViewHolder**, os parâmetros da classe conta são recebidos através do método “bindTo”. O método bindTo é responsável por associar os dados de uma conta específica aos elementos da visualização do item. Ele recebe um objeto Conta como parâmetro e define os valores dos campos nomeCliente e infoConta com base nos dados da conta. Além disso, ele chama o método addListener para adicionar um listener de clique ao item do *RecyclerView*.

3. No item anterior, o método *BindTo* da classe **ContaViewHolder** foi alterado para receber todas as informações das contas, logo são criados quatro *Intents*, um com cada uma das informações das contas, que as enviam para a classe **EditarContaActivity**.

4. Na classe **AdicionarContaActivity**, foi implementado um “*try*/*catch*” para adicionar a conta ao banco de dados. No caso de o usuário deixar os campos vazios, a aplicação chama o erro. Também foi implementado o método de inserir, cujo código cria um novo objeto Conta com os dados fornecidos e chama o método inserir do **ContaViewModel** para inserir a nova conta.

5. Na classe **ContaDAO**, foram implementados métodos para: Inserir, atualizar, deletar contas a partir de *Queries* do banco de dados. Também foram implementados métodos para buscar todas as contas e, por último, um método para fazer uma busca específica pelo número da conta. Um método extra foi implementado que soma o saldo total do banco, que será utilizado em outra etapa do projeto.

6. Na classe **ContaRepository** foram implementados métodos para inserir, atualizar e remover uma conta do banco de dados esses métodos usam “Conta” como parâmetro. Também foram implementados métodos de buscar uma conta pelo

nome, pelo CPF e pelo número no banco de dados. Estes métodos usam uma “String” contendo o parâmetro solicitado. E o método de buscar o saldo total do banco no banco de dados. Todos esses métodos foram criados utilizando os métodos da classe **ContaDAO**, criados na etapa anterior.

7. Na classe **ContaViewModel**, foram implementados métodos para inserir, atualizar as informações de uma conta, remover uma conta e buscar uma conta pelo número no banco de dados. Todos os métodos foram criados usando a classe **ContaRepository.**

8. Na classe **EditarContaActivity** as contas são editadas utilizando os valores passados via *Intent* e, assim, os campos de número da conta, nome do cliente, CPF do cliente e saldo são preenchidos.

9. Analogamente, para editar uma conta nesta classe foi implementado um “*try*/*catch*” para o caso no qual os campos estão vazios. Nessa mesma classe também foi implementado o método “atualizar”, da classe **ContaViewModel**, utilizando Conta como parâmetro.

10. Semelhante à etapa anterior, mas, para remover a conta foi utilizado o método “remover”, da classe **ContaViewModel**.

11. Na classe **BancoViewModel** são implementados os métodos de transferir, creditar e debitar. A classe **Conta** e os métodos “transferir”, “creditar” e “debitar” são utilizados para fazer a movimentação. Posteriormente, a classe **ContaReposytory** é utilizada para salvar a movimentação no banco de dados. Também foram implementados os métodos para pesquisar por nome, CPF e número usando os métodos “buscarPeloNome”, “buscarPeloCPF” e “buscarPeloNumero” da classe **ContaRepository**. Ainda nessa mesma classe existem um método que busca a soma de saldo do banco usando o método “saldoTotal” da classe “contaDao” e um para retornar esse valor.

12. Nas classes **DebitarActivity**, **CreditarActivity**, e **TransferirActivity** foramimplementados “*try*/*catch*” para o caso no qual os campos estão vazios. Na classe **DebitarActivity** o valor debitado é removido de uma conta usando o método “debitar” da classe **BancoViewModel**. Analogamente, na classe **CreditarActivity** o valor creditado é adicionado a uma conta usando o método “creditar” da classe **BancoViewModel**. Na classe **TransferirActivity** é removido um valor de uma conta

(origem) e esse mesmo valor é adicionado em outra conta (destino) usando o método “transferir” da classe **BancoViewModel**.

13. Na classe **PesquisarActivity** são utlizados métodos de pesquisa do banco de dados para realizar pesquisas por nome, CPF e número da conta. A barra de pesquisa captura o termo de pesquisa e o insere no método “getListContas” que realiza uma pesquisa no banco de dados e atualiza os resultados em uma *RecyclerView* por 20 segundos, em seguida o método “getAllContas” é chamado para exibir todas as contas no banco.

14. Explicado no item anterior.

15. Na classe MainActivity, uma “Thread” é utilizada para buscar e manter atualizado o saldo total do banco.O o método “getSaldoTotal” da classe **BancoViewModel** é utilizado para retornar o saldo total no banco.