
# Desenvolvimento de uma aplicação móvel para Android usando replicação de objetos

**Julio Cesar da Silva Gomes[1], Jan Pierre André Lima de Souza[2], Fausto Veras Maranhão Ayres[3]**

[1]Aluno de Curso Superior de Tecnologia em Sistemas para Internet – IFPB. Bolsista do CNPq. e-mail: juliocesargba@gmail.com
[2]Aluno de Curso Superior de Tecnologia em Sistemas para Internet – IFPB. e-mail: janpierreandre@hotmail.com
[3]Professor do Curso de Tecnologia em Sistemas para Internet – IFPB. e-mail: fausto.ayres@ifpb.edu.br

**Resumo:** O surgimento da plataforma Android favoreceu o desenvolvimento de aplicações móveis para as mais diversas finalidades, tal como o seu uso integrado a um sistema (desktop/web) pré-existente, compartilhando um mesmo servidor de banco de dados. Neste cenário, a aplicação necessita processar os dados de forma *online*, usando uma conexão com a Internet, o que pode resultar em altos custos, dependendo da quantidade de dados recebidos/transmitidos, e/ou em possíveis problemas devido a falhas de conexão, impedindo a troca de dados e o funcionamento correto da aplicação. Este trabalho apresenta o desenvolvimento de aplicação móvel para Android que processa informações de forma *online* e *offline*, usando replicação de objetos bidirecional entre um banco de dados local e um remoto.

**Palavras–chave:** automação comercial, banco de dados orientado a objetos, dispositivos móveis, plataforma android, replicação de objetos

## 1. INTRODUÇÃO

A expansão dos dispositivos móveis (smartphone, tablet, etc) e da plataforma Android (ANDROID, 2012) favoreceu o surgimento de aplicações móveis para as mais diversas finalidades. Tais aplicações podem ser desenvolvidas de forma independente ou de forma compartilhada, necessitando, neste último caso, receber e/ou enviar dados de/para outras aplicações remotas, móveis ou não, utilizando protocolos de conexão de rede, em geral, os da pilha TCP/IP (TANENBAUM, 2003). No caso de aplicações móveis que usam a Internet, tal conexão pode ser feita através de tecnologias sem fio, como wi-fi (IEEE 802.11) e wi-max (IEEE 802.16), ou por meio de serviços de dados das operadoras de celular (3G/4G). Neste último caso, que é o mais comum, aplicações móveis que processam dados de forma online e precisam de conexão contínua podem gerar muitos custos, dependendo da quantidade de dados recebidos/transmitidos através das operadoras de celular. Além disso, é possível que ocorram falhas de conexão com a Internet, principalmente, em locais de baixa conectividade ou sem cobertura das operadoras, impedindo a troca de dados e o funcionamento correto da aplicação.

Para exemplificar este problema, considera-se um cenário na área comercial, onde um sistema de vendas necessita de uma aplicação móvel para ser executada nos dispositivos portáteis de seus vendedores (usuários do sistema), a partir da localidade dos clientes visitados. Tais vendedores visam realizar operações comerciais, como consultar o catálogo, contendo descrição, preço, foto, etc. dos produtos vendidos, e/ou criar pedidos de venda (ou pré-vendas) que serão convertidos em vendas, em um momento posterior, por uma aplicação desktop do sistema, que compartilha com a aplicação móvel o mesmo banco de dados servidor. Neste cenário, os custos da conexão podem ser elevados, considerando-se que um vendedor utiliza a aplicação móvel de forma online e que, em cada cliente visitado, as operações executadas necessitam dos dados do banco de dados servidor. Além disso, as falhas de conexão podem prejudicar tanto as consultas quanto os pedidos de venda, inviabilizando o uso da aplicação móvel.

Para reduzir esses custos, é necessário que a aplicação móvel opere tanto de forma *online* quanto de forma *offline*, dependendo o mínimo possível da conexão com o computador servidor. Para permitir isso, a aplicação deve operar com um banco de dados local e outro remoto (o servidor), que devem ser sincronizados, quando necessário, como esquematizado na Figura 1. É importante ressaltar que apenas uma parte dos dados do servidor é acessível pela aplicação móvel e, portanto, precisará ser sincronizada com o seu banco local.

**ISBN 978-85-62830-10-5**

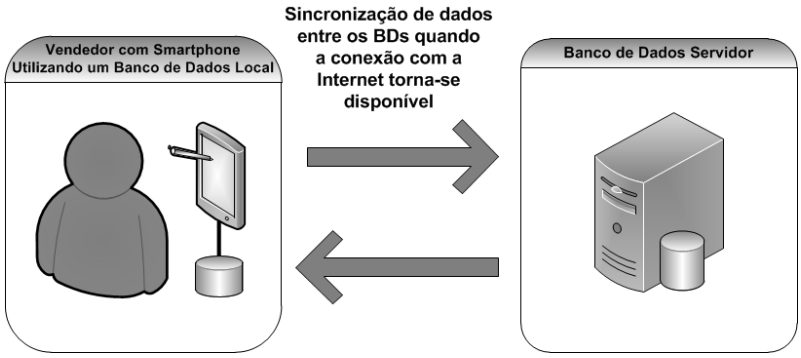


Figura 1 – Esquema para sincronização bidirecional entre o banco de dados local e o remoto

Neste contexto, uma questão importante é como persistir os objetos da aplicação, usando um banco de dados local, e como estabelecer a sincronização deste com o banco de dados remoto.

O objetivo deste trabalho é o desenvolvimento de uma aplicação móvel que realize, de forma *online* e *offline*, as operações comerciais descritas no cenário acima, envolvendo a replicação de objetos bidirecional entre um banco de dados local e um remoto.

## 2. MATERIAL E MÉTODOS

Para o desenvolvimento da aplicação móvel, foram utilizadas as seguintes tecnologias abertas:

- IDE Eclipse Indigo – ambiente de desenvolvimento Java
- Emulador Android – ambiente de simulação da plataforma *Android* no eclipse
- Db4o (*Database for Objects*) 8.0 – SGBD orientado a objetos
- DRS (*Db4o Replication System*) 8.0 – Módulo de sincronização de objetos do Db4o

As etapas de desenvolvimento do trabalho são:

- Desenvolvimento das classes de negócio relacionadas ao cenário proposto
- Desenvolvimento da aplicação acessando apenas o banco local
- Desenvolvimento do serviço de sincronização de objetos entre o banco local e o remoto

Durante todo o processo de desenvolvimento da aplicação, foi adotada a experimentação científica, através de testes práticos realizados em um ambiente composto por um *smartphone*, com o sistema operacional Android, e um computador *desktop*, com o sistema operacional Windows 7, ambos habilitados com serviço de Internet. O banco de dados local e o remoto são gerenciados pelo Db4o e a sincronização entre eles é feita pelo DRS.

## 3. RESULTADOS E DISCUSSÃO

A abordagem seguida neste trabalho foi a de desenvolver uma aplicação móvel integrada a um sistema comercial pré-existente[1], compartilhando o mesmo banco de dados servidor, exigindo a troca de informações entre ele e o banco local da aplicação, que é feita sob demanda, utilizando a replicação de objetos Java. A utilização de técnicas de replicação de dados (CHARRON-BOST, 2010) é algo presente na maioria dos sistemas utilizados em rede, visto que são bastante confiáveis para a consistência dos dados da instituição usuária. Por outro lado, para a plataforma móvel, existem poucos sistemas com a capacidade de replicação de seus dados. Isto se dá por conta de limitações inerentes a essa plataforma.

A aplicação móvel foi projetada para operar prioritariamente de forma *offline*, podendo ser usada também de forma *online*, quando necessário, contribuindo com a redução dos custos e eventuais

[1] O detalhamento desse sistema está fora do escopo deste trabalho.

ISBN 978-85-62830-10-5

problemas apresentados na introdução deste artigo. Outra decisão de projeto foi a de utilizar o Sistema Gerenciador de Banco de Dados (SGBD) orientado a objetos Db4o (DB4O, 2012), para o gerenciamento dos bancos de dados local e remoto, e utilizar a sua API (*Application Programming Interface*) DRS, para replicação de objetos entre os bancos de dados, a qual encapsula operações de mais baixo nível, como *sockets.*

A seguir serão descritos aspectos do funcionamento da aplicação e de sua implementação.

A Figura 2 apresenta um diagrama UML (*Unified Modeling Language*), simplificado, com as principais classes de negócio utilizadas para o desenvolvimento da aplicação móvel. Observa-se neste diagrama que um vendedor possui vários pedidos que contem vários itens de produto e cada produto pode ter vários estoques (em diferentes datas).

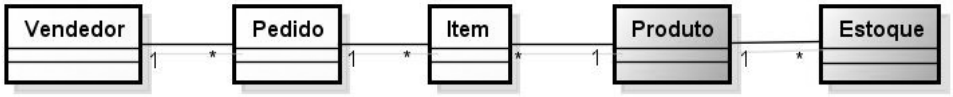


Figura 2 – Classes de negócio da aplicação

Após a inicialização da aplicação móvel, uma tela de *login* (Figura 3a) é apresentada ao usuário[2] que fornece o seu nome e senha para serem validados, usando o banco de dados local. A Figura 3b mostra a segunda tela da aplicação, contendo um *menu* de opções, descrito a seguir:

- **Listar Produtos**: Exibe a tela que lista todos os produtos do catálogo (Figura 4a), não permitindo a edição dos mesmos;
- **Novo Pedido**: Exibe a tela que cria um novo pedido (Figura 4b) com os itens do pedido, selecionando-se o produto e a quantidade de cada item (similar a um "carrinho de compras");
- **Listar Pedidos**: Exibe a tela que lista todos os pedidos em aberto (Figura 4c) do usuário, os quais podem ser excluídos ou editados com a inclusão ou remoção de itens e a alteração da quantidade de um item;
- **Enviar Atualizações:** Executa a ação de enviar para o banco de dados remoto todos os pedidos criados e alterados pelo usuário desde o último envio;
- **Logout:** sair da conta de usuário da aplicação.

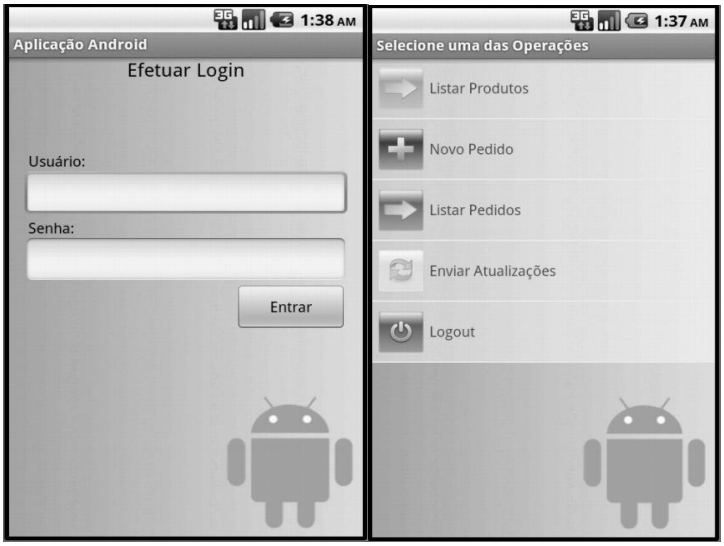


Figura 3 - Telas de *login* (a) e de *menu* (b) da aplicação

[1]Considera-se, como usuário, todo vendedor previamente cadastrado no banco de dados remoto

ISBN 978-85-62830-10-5

Cada novo pedido é identificado por um número gerado pela aplicação, o qual combina o código do usuário com o numero sequencial do pedido, que é iniciado em "1" para cada usuário.

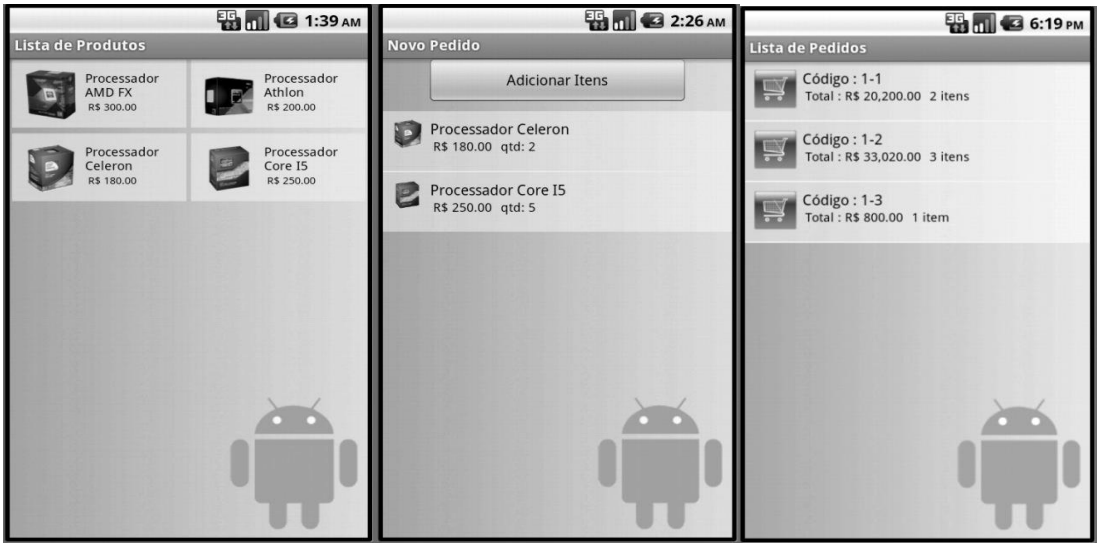

Figura 4 - Telas de (a) listar produtos, (b) novo pedido e (c) listar pedidos

A opção "Enviar Atualizações" do *menu* é necessária, pois os pedidos são salvos apenas no banco de dados local, possibilitando que a aplicação seja executada de forma *offline*. O envio dos objetos atualizados ocorrerá até o final da execução da aplicação ou logo após o seu reinício, dependendo da disponibilidade de conexão com o servidor. Após o envio, uma notificação será exibida na barra de notificações do aparelho, como mostra a Figura 5.

Figura 5 – Tela de notificação

Quando a aplicação é iniciada, um Serviço da Aplicação (SA) é iniciado em *background* e tem como função verificar continuamente a existência de conexão do dispositivo com a Internet, sendo que podem ocorrer duas situações distintas:

- Quando existir a conexão, esse serviço dispara um processo de atualização do banco de dados local a partir do remoto, recebendo os novos objetos relacionados aos cadastros de produtos e de usuários do sistema.

**ISBN 978-85-62830-10-5**

- Enquanto não existir a conexão, a aplicação usará os dados previamente carregados para o banco de dados local desde o último processo de atualização. Isso exige que haja uma carga inicial dos objetos remotos na primeira inicialização da aplicação no dispositivo, permitindo que o usuário faça o login e acesse o catálogo de produtos, usando esses objetos.

Após o *login* do usuário, o SA dispara um novo processo de atualização do banco de dados local, a partir do remoto, recebendo as alterações dos produtos, como, por exemplo, um reajuste de preço. As imagens recebidas dos produtos ficam armazenadas no diretório padrão de arquivos do Android e suas referências ficam armazenadas no banco local. Acaso algum dos produtos não possua imagem cadastrada, será usada uma imagem padrão.

Caso um usuário execute a aplicação pela primeira vez em outro dispositivo, o SA atualizará, localmente, os pedidos em aberto dele que estão no servidor, permitindo que ele possa continuar o seu trabalho nesse dispositivo. É importante ressaltar que os pedidos finalizados do usuário são excluídos do banco local e ficam disponíveis somente no banco de dados remoto, para a concretização de suas vendas através de outra aplicação do sistema, liberando espaço no banco de dados local para novos pedidos.

A aplicação móvel é implementada com o uso das *Activities* do Android, para criar e gerenciar as telas responsáveis pelas operações realizadas pelo usuário, e o uso das classes *Service* nas operações de atualização executadas em segundo plano, permitindo que o serviço continue ativo mesmo após a aplicação ser fechada com o uso do botão "voltar" padrão do Android.

A API DRS efetua a replicação de objetos de maneira direcional ou bidirecional. A Figura 6 mostra um método da aplicação, utilizando o DRS para a replicação bidirecional de objetos, executando as seguintes tarefas.

- Linhas 37: instanciar uma conexão com o banco de dados local
- Linhas 38: instanciar uma conexão com o banco de dados servidor
- Linha 39: instanciar um provedor de replicação para o banco de dados local
- Linha 40: instanciar um provedor de replicação para o banco de dados servidor
- Linha 41: preparar a replicação entre os bancos conectados
- Linha 42: localizar os objetos alterados no banco local desde a última atualização
- Linhas 43-44: replicar as atualizações do banco local para o remoto
- Linha 45: localizar os objetos alterados no banco remoto desde a última atualização
- Linhas 46-47: replicar as atualizações do banco remoto para o local
- Linha 48-49: efetivar as replicações e encerrar as conexões.

Desta forma, todos os objetos da classe Vendedor e de suas classes relacionadas, que sofreram algum tipo de atualização, serão replicados entre os dois bancos de dados.

```
36      public void replicate( ){
37          ObjectContainer localmanager = getManager();
38          ObjectContainer externalmanager = getServerManager();
39          Db4oEmbeddedReplicationProvider providerA = new Db4oEmbeddedReplicationProvider(localmanager);
40          Db4oEmbeddedReplicationProvider providerB = new Db4oEmbeddedReplicationProvider(externalmanager);
41          ReplicationSession session = Replication.begin(providerA, providerB);
42          ObjectSet changedInA = session.providerA().objectsChangedSinceLastReplication(Vendedor.class);
43          while (changedInA.hasNext())
44              session.replicate(changedInA.next());
45          ObjectSet changedInB = session.providerB().objectsChangedSinceLastReplication(Vendedor.class);
46          while (changedInB.hasNext())
47              session.replicate(changedInB.next());
48          session.commit();
49          session.close();
50      }
```

Figura 6 – Método responsável pela replicação bidirecional

## 4. CONSIDERAÇÕES FINAIS

**ISBN 978-85-62830-10-5**

Com base nos resultados, pode-se concluir que a aplicação móvel desenvolvida efetua as operações comerciais de consulta a produtos e de pedido de vendas, tanto de forma *online* como de forma *offline*, utilizando a replicação de objetos entre um banco de dados local e um remoto de forma bidirecional. O emprego da tecnologia DB4o resultou em várias vantagens, como o baixo custo de desenvolvimento da aplicação, devido ao seu modelo orientado a objetos que elimina a necessidade de mapeamento de objetos para tabelas (comumente encontrado em aplicações feitas em Java que persistem seus objetos em banco de dados relacionais) e ao uso de poucos recursos computacionais do dispositivo móvel, tal como a pouca quantidade de memória RAM consumida, além da facilidade de replicação de objetos.

Pretende-se, num futuro próximo, adotar outros tipos de SGBD, tanto para o banco de dados local quanto para o remoto, para efetuar a replicação de dados com diferentes modelos de banco de dados.

## AGRADECIMENTOS

Este trabalho foi realizado com o apoio do Programa de Incentivo em Desenvolvimento Tecnológico e Inovação (PIBITI) do CNPq.

## REFERÊNCIAS

ANDROID. **Android Developers.** Disponível em <http://www.android.com>. Acesso em 05/07/2012.

CHARRON-BOST, B. et al. **Replication - Theory and Practice Series: Lecture Notes in Computer Science** , Vol.5959 1st Edition., 2010.

DB4O. **Database For Objects.** Disponível em <http://www.db4o.com>. Acesso em 05/07/2012.

TANENBAUM, A.S. **Redes de Computadores.** 4. Ed. Rio de Janeiro: Campus, 2003.