Universidade do Vale do Rio dos Sinos - Unisinos Ciências Exatas e Tecnológicas – Desenvolvimento de Aplicações para Dispositivos Móveis

OPENMID: UM MIDDLEWARE ANDROID PARA BANCOS DE DADOS MÓVEIS

Luiz Fernando Duarte Júnior[1]
Prof. Msc. Gilberto Irajá Müller[2]

CONTEXTO: Os avanços da computação móvel permitiram aos dispositivos como *tablets* e *smartphones* atuarem como extensões dos ambientes de trabalho, usando *softwares* que integram-se com sistemas de gestão, de relacionamento com clientes e, consequentemente, que tem de lidar com um grande volume de dados remotos. PROBLEMA: O desenvolvimento de aplicações móveis que acessam SGBDs remotos representam desafios aos desenvolvedores de sistemas para estes dispositivos, uma vez que depende dos recursos limitados do aparelho, da rede de dados móveis que não está disponível em todos os territórios e de uma mudança na arquitetura das aplicações, devido à instabilidade das conexões móveis a SGBDs. SOLUÇÃO: Devido a estes problemas, foi desenvolvido um protótipo de um *middleware* de acesso móvel a SGBDs remotos, o qual irá intermediar requisições e respostas entre a plataforma móvel e o SGBD. MÉTODO PROPOSTO: Este trabalho descreve um estudo comparativo entre o protótipo de *middleware* para acesso móvel à SGBDs remotos e as principais soluções existentes no mercado que atendem à plataforma Android. CONCLUSÃO: A partir da implementação do protótipo e a análise do estudo comparativo, os resultados incentivam o uso do mesmo, assim como a continuidade através de trabalhos futuros.

Palavras-chave: *Middleware*; Banco de Dados Móveis; SGBD; Android; MySQL.

1 INTRODUÇÃO

O uso de celulares cada vez mais parecidos com computadores, os chamados *smartphones* (IPSOS, 2012), tem crescido muito nos últimos anos. Com isso, a demanda por sistemas operacionais e aplicativos móveis desencadeou uma proliferação de fabricantes e desenvolvedores, tentando suprir as necessidades do mercado. Entretanto, os profissionais envolvidos em transformar esta revolução

[1] Luiz Fernando Duarte Junior. Bacharel em Ciência da Computação, ULBRA. Pós-graduando em Desenvolvimento de Aplicações para Dispositivos Móveis, Unisinos. E-mail: luizfduartejr@gmail.com

[2] Prof. Msc. Gilberto Irajá Müller, Mestre em Computação Aplicada com atuação na área de Análise e Desenvolvimento de Aplicações para Dispositivos Móveis. E-mail: gimuller@unisinos.br

móvel em realidade se vêem com enormes desafios, uma vez que a computação móvel possui restrições adicionais se comparada à computação tradicional. Além da limitação de recursos, segurança e energia finita, o problema de conexão móvel de dados figura como um dos maiores entraves para os projetos de sistemas móveis (SATYANARAYANAN, 1996). Em conexões com alta latência, baixa largura de banda, perda de pacotes de dados ou intermitência de sinal, esbarra-se na questão de confiança no serviço, o que, para os desenvolvedores, implica em projetar suas aplicações de modo a sofrerem o mínimo possível com essas desvantagens, embora nem sempre isso seja possível.

O objetivo de um *middleware* de acesso à dados é prover uma camada intermediária para acesso confiável e consistente à SGBDs (Sistemas de Gerência de Banco de Dados) remotos, através de um dispositivo móvel. Desta forma, garante-se que os dados trafegados entre o dispositivo móvel e o banco de dados estejam protegidos logicamente contra ações maliciosas e que todas as operações serão efetivadas no banco de dados remoto. Além disso, os fabricantes costumam adicionar outras características que geralmente visam a otimização da comunicação entre as plataformas, como compressão de dados, *caching*[3] e *pipelining*[4].

1.1 Motivação

Devido ao fato de que as plataformas móveis, entre elas o Android, não suportam conexões diretas a bancos de dados remotos (devido à inúmeras características das redes móveis), um desenvolvedor que busque exemplos de conexões simples entre sua plataforma móvel e um SGBD achará a tarefa um tanto difícil. Embora os clientes sejam inúmeros para outros ambientes, quando o assunto é *mobile*, a orientação mais comum é a criação de *web services*[5] para disponibilizar os dados que se deseja consumir. Entretanto, esta abordagem leva os desenvolvedores a uma mudança arquitetural e aumento no tempo do projeto.

---

[3] *Cache* é um local que armazena acessos de memória e pode ter uma cópia dos dados que você está requisitando, reduzindo latência de referências para memória (BOTTOMLEY, 2004).

[4] *Pipeline* é uma série de estágios que cada instrução no fluxo de código deve passar quando o mesmo está sendo executado (STOKES, 2004).

[5] Segundo o W3C Working Group (2004), um *web service* é um sistema de *software* projetado para suportar interações máquina-para-máquina interoperáveis sobre uma rede.

Existem alguns fabricantes de bancos de dados que estão investindo na pesquisa e desenvolvimento de *middlewares* para acesso aos seus SGBDs como Oracle e SyBase. Entretanto todos existem para se utilizar com os bancos de dados destes mesmos fabricantes. Por outro lado, existe uma enorme demanda por *middlewares open-source*[6] que permitem aos desenvolvedores diminuírem o tempo de desenvolvimento de aplicações móveis que necessitam se comunicar com SGBDs remotos. O trabalho será conduzido de forma que o produto final seja de utilidade para estudos posteriores e implementações de *middlewares* para outras plataformas (iOS, Blackberry, etc) e bancos de dados (Oracle, SQL Server, etc).

1.2 Objetivos

Este trabalho possui como objetivo estudar, projetar e implementar um *middleware* Android para acesso remoto à Sistemas de Gerência de Banco de Dados. O *middleware* permitirá a conexão segura e eficiente entre aplicativos Android e servidores MySQL, previamente configurados através da instalação de agentes que usam a biblioteca de classes do *middleware*. De forma a atingir o objetivo deste artigo, foi realizado um estudo sobre acesso móvel à dados e sobre os *middlewares* existentes e suas arquiteturas, o que ofereceu uma visão ampla dos problemas enfrentados e algumas das melhores maneiras de solucioná-los. Para os testes do *middleware* foram utilizados um agente-cliente (no dispositivo móvel) e um agente-servidor (no servidor de banco de dados), que intermediarão o consumo de dados entre uma aplicação Android e um banco de dados MySQL.

1.3 Organização do Texto

Este artigo está estruturado da seguinte forma: na Seção 2 é apresentado o referencial teórico, abordando um panorama breve da situação atual da Computação Móvel, focado na visão do que ela é e os problemas de acesso à. Na mesma seção

[6] Segundo a Open-Source Initiative (2013),um programa de computador open-source (código-aberto) é aquele cuja redistribuição é livre, o acesso ao código-fonte é aberto, que permite trabalhos derivados e que não discrimina ou restringe-se à grupos ou pessoas.

é apresentada uma definição sobre acesso móvel à dados e os desafios ao desenvolvimento de aplicações móveis que consumam dados de SGBDs remotos.

Na Seção 3 fazem-se comentários sobre os trabalhos relacionados, incluindo os dois principais *middlewares* de acesso à SGBDs do mercado, explicando de forma sucinta a sua arquitetura e peculiaridades, além de outras soluções menores.

Na Seção 4 apresenta-se a metodologia de pesquisa aplicada, baseada em um estudo comparativo entre as soluções de mercado com o protótipo desenvolvido.

Na seção 5 a metodologia de desenvolvimento é apresentada, detalhando os diagramas do projeto e as partes que o compõem, bem como ilustrações dos testes, abordagem teórica dos problemas e da segurança envolvida no protótipo.

Na seção 6 tem-se a análise do estudo comparativo, cruzando os dados obtidos e traçando um panorama da situação atual do protótipo desenvolvido frente às soluções concorrentes. Por fim, encerra-se o trabalho com a conclusão.

2 REFERENCIAL TEÓRICO

A computação móvel nunca esteve tão presente em nossas vidas como na atualidade. Mesmo em países em desenvolvimento, milhões de pessoas usam *smartphones* para realizar tarefas que, até então, somente podiam ser feitas através dos seus computadores, tais como: ler notícias, acessar e-mails, verificar compromissos, navegar na Internet, etc. Em quase todas elas, o acesso à dados que não estão no próprio dispositivo se faz necessário. Não obstante, o meio mais utilizado atualmente para armazenamento de dados, independente de plataforma, são os SGBDs, fazendo-se necessário para a execução do trabalho proposto, um estudo sobre *middlewares* de acesso móvel a SGBDs.

2.1 Middleware e Acesso à Bancos de Dados

Segundo Krakowiak (2003), em um sistema de computação distribuída, *middleware* é definido como uma camada de *software* que reside entre o sistema operacional e as aplicações em cada ponto do sistema. Devido à heterogeneidade de arquiteturas permitidas em conjunto com a Internet, o desenvolvimento de *middlewares* vem crescendo com força, principalmente em ambientes empresariais em que é necessário integrar diversas tecnologias diferentes de forma que: a distribuição seja transparente, a heterogeneidade seja irrelevante (do ponto de vista

do usuário), o desenvolvimento de novas funcionalidades de *software* seja de alto nível e a comunicação entre os serviços computacionais seja facilitada como um todo, agindo como uma interface.

Um uso comum de *middlewares* é na programação de aplicações com bancos de dados, onde comumente conecta-se ao *middleware* do banco de dados para solicitar informações e executar comandos, e não ao mecanismo do banco em si. Para Linthicum (2001), um *middleware* orientado à banco de dados é qualquer *middleware* que facilite a comunicação com uma base de dados, seja a partir de uma aplicação ou entre diferentes bancos de dados. Exemplos citados pelo autor incluem o Microsoft ODBC (*Open Database Connectivity*) e o Oracle JDBC (*Java Database Conectivity*), padrões de acesso à dados comuns entre projetos de *software* Windows e Linux, agindo como interfaces de comunicação.

A vantagem no uso desses *middlewares* reside no fato de que através de instruções da própria linguagem de programação que se está utilizando é possível conectar-se e enviar comandos ao sistema na outra ponta (banco de dados, por exemplo) recebendo como resposta os dados em objetos da respectiva linguagem, mantendo uma programação de alto nível e extremamente simplificada. Como desvantagens, Campbell (1999) cita queda de desempenho e limitação no acesso aos recursos e funcionalidades do sistema, uma vez que o desenvolver tem acesso somente ao que o *middleware* expõe do sistema nativo.

2.2 Acesso Móvel à Bancos de Dados

Para que seja possível acessar dados remotos com dispositivos móveis sem o uso de cabos, uma intrincada trama de técnicas e tecnologias se faz necessária para que haja a conectividade visando a comunicação entre diferentes aparelhos, como *smartphones*, *notebooks*, computadores e dispositivos de rede. Com isso em mente, os desenvolvedores de aplicativos móveis, devem ser capazes de lidar com a troca de dados em ambientes heterogêneos e pouco confiáveis, o que gera inúmeros desafios.

Existem diversos problemas relacionados ao desenvolvimento de *middlewares* de acesso a dados, tais como: gerência de transações, controle de

concorrência, falhas, recuperação, segurança e integridade (Barbosa, 2001). No entanto, três deles foram alvos deste estudo: segurança dos dados, consistência das transações e desempenho.

2.2.1 Segurança dos Dados

Segundo Janssen (2013), a segurança dos dados refere-se às medidas de privacidade digital que são aplicadas para prevenir acesso não autorizado à computadores, bases de dados e *websites*, além de também proteger os dados contra corrupção. Em um *middleware* móvel para acesso a bancos remotos deve-se manter sigilo sobre as credenciais de acesso ao banco de dados, evitando que pessoas não autorizadas tenha acesso aos dados do mesmo e às mensagens trafegadas entre cliente e servidor.

Técnicas comuns para garantir a segurança de dados sobre meios inseguros, como a Internet, incluem o uso de criptografia que, segundo Katz (2007), é o estudo científico de técnicas para a segurança de informação digital, transações e computação distribuída. Existem inúmeros algoritmos de criptografia e formas de aplicá-los, devendo-se analisar as necessidades reais de segurança em comparação com o aumento do custo computacional que cada algoritmo traz consigo.

Basicamente, existem algumas famílias de algoritmos de criptografia e diferentes formas de catalogá-las. Segundo Kessler (1998), pode-se classificar os algoritmos de criptografia baseados no número de chaves que possuem para encriptação e decriptação: algoritmos de chave simétrica, onde somente uma chave permite cifrar e decifrar os dados; algoritmos de chave assimétrica, onde existe uma chave que somente cifra (chamada de chave pública) e outra que somente decifra os dados (chave privada); e algoritmos de *hashing*, onde não há meios de decifragem dos dados após cifrados. Exemplos de algoritmos usados atualmente incluem: RSA (assimétrico), AES (simétrico) e MD5 (*hashing*), entre muitos outros.

Além de proteger os dados contra leitura não autorizada através de criptografia, é necessário que se garanta que cada requisição seja executada somente uma vez e ainda assim, com um prazo de validade para que não seja usada futuramente para fins maliciosos, atividade conhecida como *Replay Attack,*

que segundo Barmala (2004) é o ato de usar mensagens copiadas para atacar um sistema computacional ou obter acesso não autorizado. Uma das maneiras para atender tais requisitos é usando o conceito de *timestamp* que determina que as requisições geradas dentro de um intervalo de tempo específico somente são válidas enquanto durar aquele intervalo, evitando que a captura de requisições que não foram processadas pelo servidor sejam usadas futuramente.

2.2.2 Consistência das Transações

Para um banco de dados, uma transação é uma unidade de trabalho lógica e atômica que contém uma ou mais diretivas SQL (ORACLE, 2011). Uma transação agrupa diretivas SQL para garantir que ou todas sejam executadas com sucesso, ou nenhuma, sendo desfeitas (*rollback*). Ainda segundo a Oracle (2011), as propriedades básicas de uma transação são popularmente referenciadas pelo acrônimo ACID (iniciais de Atomicidade, Consistência, Integridade e Durabilidade).

Em um *middleware* de acesso à dados, dado um grupo de comandos enviados pelo cliente a um servidor, deve-se garantir que todos os comandos sejam executados, ou nenhum deles deve permanecer na base. À primeira vista pode parecer óbvio e simples, uma vez que os SGBDs já endereçam esses problemas nativamente, mas considerando que a proposta é um *middleware* para acesso móvel a bancos de dados, é natural que a conexão de dados seja instável e é função do *middleware* (BILLARD, 1998) garantir que essa instabilidade não comprometa a confiança do usuário sobre o sistema.

2.2.3 Desempenho

E por último, outro grande desafio do acesso móvel à bases de dados remotas é o desempenho da conexão. Ainda que as novas gerações de redes de dados móveis tenham melhorado bastante nos últimos anos, o panorama geral ainda é abaixo da média esperada, segundo a Anatel (2013), quando o assunto é conexão móvel de dados. A cada requisição que é enviada ao SGBD tem-se como resposta o *download* de volumes de dados para o dispositivo, o que pode ocasionar queda de desempenho. Não obstante, para garantir a segurança e consistência das transações uma série de verificações adicionais serão efetuadas ao longo do ciclo de vida das requisições, o que servirá para aumentar o tempo de resposta a cada

ação do usuário que exija comunicação com o banco de dados. Com este cenário em mente, algumas técnicas incrementam consideravelmente o tempo de requisição e resposta, entre elas a compressão de dados, a paginação de resultados e *caching*, que serão discutidas a seguir e que são úteis em *middlewares* de acesso à dados.

Segundo Nelson e Gailly (1995), a compressão de dados busca reduzir o número de bits usados para armazenar ou transmitir informação. Isso é possível através de diversas técnicas como eliminação de redundância na representação dos dados e remoção de dados pouco significativos (cuja falta é quase imperceptível, como no caso de arquivos MP3). Embora a compressão de dados aumente o processamento das requisições (afinal, deve-se realizar as operações de compressão e descompressão a cada conjunto de dados), Shannon (1948) afirma que os ganhos em velocidade de tráfego na rede superam as perdas em processamento quando se utilizam grandes volumes de dados com altas taxas de compressão (como grandes quantidades de texto ou arquivos binários) e mais ainda quando as redes em questão são redes de baixa velocidade. Desta forma, é possível através da compressão de dados otimizar o desempenho de um *middleware* de acesso a dados.

A paginação de resultados é uma técnica muito comum na plataforma *web* para aumento do desempenho nas páginas, principalmente em sistemas com grandes quantidades de registros por tabela no banco de dados. A paginação de resultados baseia-se no princípio de que quando um grupo de dados possui muitos registros (dependendo do sistema), além dele ser oneroso para ser transportado pela rede de dados, é na maioria das vezes pouco prático para ser analisado ou até mesmo inútil se visualizado em sua totalidade. Dessa forma, trabalha-se com o conceito de páginas, onde cada página contém uma fração do total de registros. Com isso poupa-se a necessidade de trafegar uma carga muito grande de dados quando na maioria das vezes o usuário do sistema só prestará atenção em uma pequena fração dela por vez. De acordo com Brade (1989), técnicas de paginação também são padrões nos protocolos das redes de computadores, que dividem a informação que irá trafegar pela rede em *frames,* uma vez que seria pouco eficiente tentar transferir toda a informação de uma só vez. Aliado ao fato de que as redes móveis de dados são ainda mais instáveis que as redes tradicionais e que as telas dos dispositivos móveis permitem uma análise de dados mais limitada que nos

computadores, têm-se na paginação de resultados uma opção para aumento no desempenho do retorno das consultas realizadas ao banco de dados.

E por fim, técnicas de *caching* que, segundo Bottomley (2004), servem para armazenar dados muito utilizados ou que serão utilizados em breve em uma memória mais veloz do que a normal onde aquele dado costuma residir. Dados armazenados em bancos de dados, principalmente os relacionais, tendem a estar organizados em uma estrutura hierárquica que, entre outras características, prioriza a integridade referencial em detrimento ao desempenho, o que leva aos próprios fabricantes de sistemas de bancos de dados investirem em mecanismos de *caching* para armazenar dados em memória principalmente durante operações mais extensas e complexas. Outro uso para mecanismos de *caching* está na arquitetura de computadores, onde existem diversos tipos e níveis de memória *cache*, como a memória cache existente nos processadores, que evita o tráfego desnecessário (e lento) entre a unidade lógica aritmética (ULA) e o dado na memória RAM ou, pior ainda, no disco rígido. No caso de bancos de dados, é comum que as tabelas ou combinações de dados mais comuns sejam mantidas em estruturas organizacionais diferenciadas, como índices e visões, e as consultas mais comumente utilizadas sejam mantidas na memória RAM; tudo isso visando um aumento no desempenho do serviço de banco de dados como um todo. Ainda que nos dispositivos móveis as memórias mais velozes sejam de maior custo e devem ser usadas com parcimônia, no caso dos servidores de bancos de dados é possível utilizar-se de mecanismos de *cache* para diminuir o impacto causado pelo uso de um *middleware*.

Aplicando-se as três técnicas anteriores a um *middleware* de acesso à dados, têm-se um ganho de desempenho que embora não seja superior a implementações de acesso à banco de dados usando *web services*, é mais adequado e confiável ao cenário móvel.

3 TRABALHOS RELACIONADOS

Nesta seção serão apresentados alguns *middlewares* de acesso móvel à SGBDs existentes no mercado, para que possa se traçar um comparativo com a solução que está sendo proposta neste trabalho.

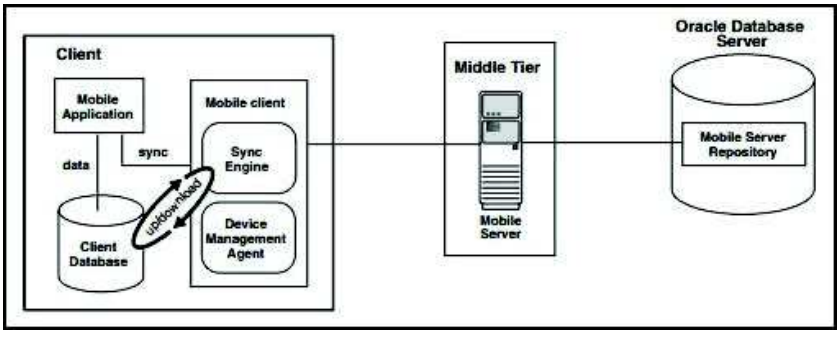
3.1 Oracle Database Mobile Server

Segundo a Oracle (2013), o Oracle Database Mobile Server é a melhor maneira de conectar com segurança dispositivos embarcados e aplicativos móveis à bases de dados Oracle. Seus principais diferenciais residem no seu motor de sincronização, que mantém diversas bases móveis em sincronia com o servidor de banco de dados, e em sua resiliência sobre conexões instáveis. Além disso, os dados armazenados e trafegados estão sob forte criptografia, garantindo a segurança dos mesmos contra interceptação.

Para que seja possível às aplicações móveis se conectarem ao servidor, é preciso utilizar um cliente móvel fornecido pela Oracle, que deve ser instalado no dispositivo e cuja arquitetura compreende (ORACLE, 2011): um banco de dados cliente (SQLite ou Berkeley DB), um cliente móvel (que contempla um motor de sincronização e um agente de gerenciamento para receber comandos remotos) e a aplicação móvel em si. Esse cliente móvel é suportado pelas plataformas Android, Blackberry, Microsoft Windows, Oracle Enterprise Linux, Windows Mobile, Windows Phone e iOS. Com exceção da sincronização automática, inexistente para a plataforma Blackberry, todos os recursos estão disponíveis em todos os sistemas operacionais citados anteriormente.

Entre o cliente móvel e o servidor Oracle tem-se o Mobile Server em si. Com isto, entende-se que a arquitetura completa da solução da Oracle consiste em uma comunicação cliente-*middleware*, *middleware*-servidor, conforme mostrado na figura a seguir.

Figura 1: Arquitetura do Oracle Database Mobile Server



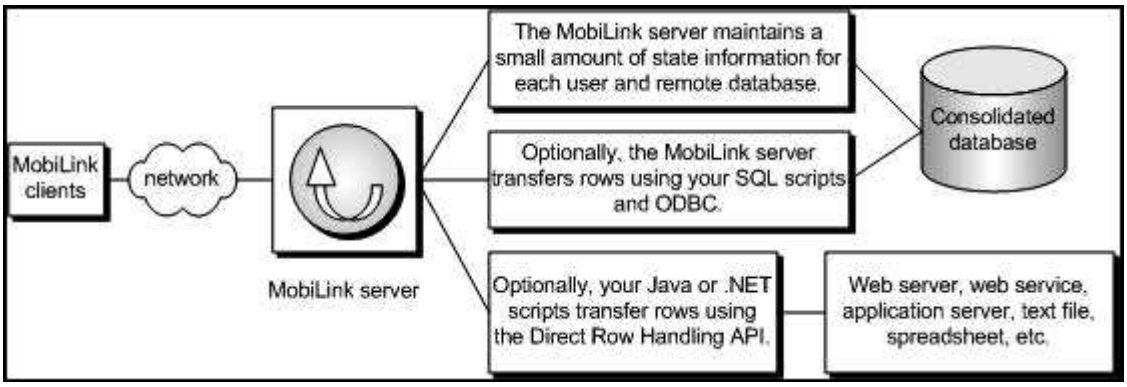
Fonte: Oracle (2011)

3.2 SAP Sybase SQL Anywhere

Segundo a Sybase (2013), o SAP Sybase SQL Anywhere é uma suíte abrangente de soluções que fornece tecnologias de gerenciamento, sincronização e troca de dados que permitem o desenvolvimento e a implantação rápida de aplicativos operados por bancos de dados em ambientes remotos e móveis. O SQL Anywhere Server executa na maioria dos ambientes de servidores da atualidade, entretanto sob o ponto de vista de acesso móvel à SGBDs, é o cliente móvel chamado UltraLite que interessa à este estudo.

O UltraLite é uma versão portada do SQL Anywhere que executa sobre as principais plataformas móveis do mercado como Android, iOS, Blackberry e Windows. Entre o cliente e o banco de dados consolidado fica o MobiLink (SYBASE, 2013): tecnologia de sincronização baseada em sessão, desenvolvida especialmente para sincronizar o UltraLite e o banco de dados SQL Anywhere entre eles e com outros bancos de dados empresariais, conforme a Figura 2, onde o MobiLink clients pode ser clientes UltraLite, SQL Anywhere Server ou ambos, em um ambiente completamente heterogêneo.

Figura 2: Arquitetura de uma aplicação usando MobiLink



Fonte: Sybase (2013)

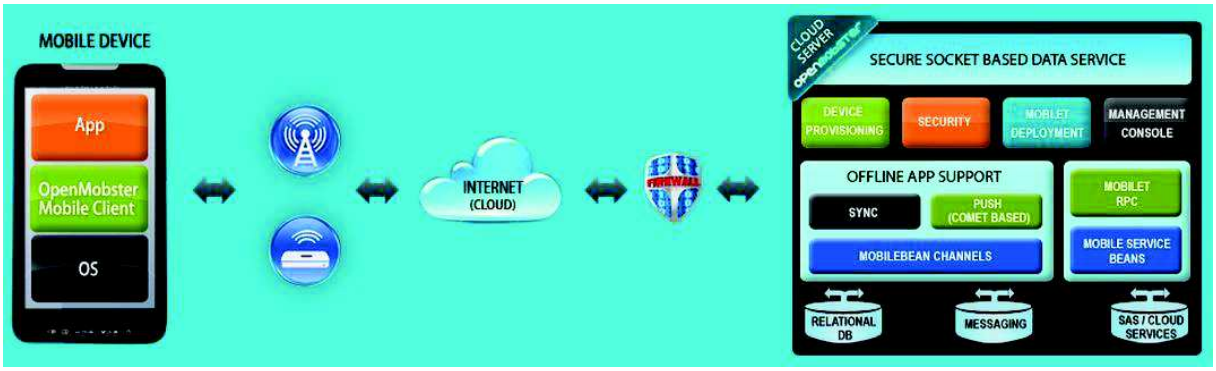
3.3 Outras Propostas

Além das soluções supracitadas, destacam-se ainda no mercado de *middlewares* móveis para acesso a dados remotos as soluções OpenMobster e MobiForms, descritas brevemente abaixo. A escolha destas soluções para serem

apresentadas deve-se à sua proposta de valor clara e focada em mobilidade, bem como suporte à plataforma Android, tal qual o proposto neste trabalho.

Segundo o OpenMobster (2012), o OpenMobster é uma plataforma *open source* para integração de aplicativos móveis com serviços na Internet, fornecendo principalmente sincronização de dados e notificações *push* (notificações no sentido servidor-cliente) em tempo real, fornecendo inclusive a infraestrutura como serviço para utilização da plataforma. Fortemente orientada para aplicativos híbridos[7] sobre a plataforma Phonegap[8], o OpenMobster tem como principal diferencial o seu sistema de atualização em modo *push*, onde os dados são sincronizados com os dispositivos automaticamente, sem a necessidade do usuário solicitar atualizações. A Figura 3 exemplifica a arquitetura em nuvem do OpenMobster.

Figura 3: Arquitetura em nuvem OpenMobster



Fonte: OpenMobster (2013)

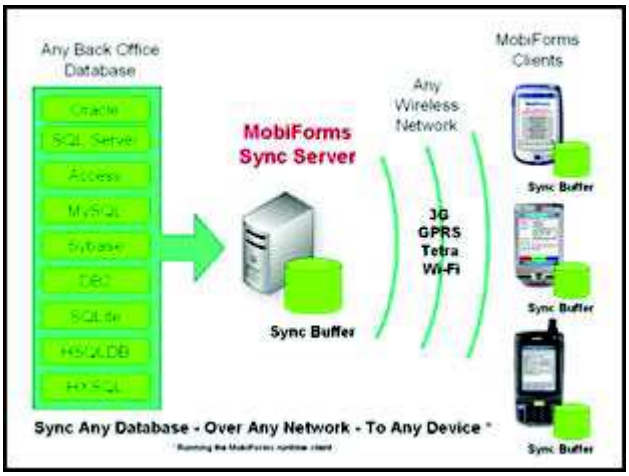
Já o MobiForms (2013) é uma suíte para desenvolvimento rápido de aplicações móveis, nativas ou híbridas, para inúmeras plataformas, sobre uma camada que abstrai as peculiaridades da plataforma do dispositivo. O recurso estudado que atende à necessidade de acesso móvel a bancos remotos é o MobiForms Sync Server, uma ferramenta complementar ao MobiForms Developer que oferece armazenamento *offline* e capacidade de sincronização *online*,

[7] Aplicativos híbridos são, segundo Rigo (2013), aquelas que usam o mesmo conjunto de ferramentas para desenvolvimento de código empregado nas aplicações web (HTML, CSS, etc), mas, além disso, utilizam também alguns recursos das aplicações nativas.

[8] Phonegap (2013) é um framework gratuito e open-source que permite a criação de aplicativos móveis usando APIs web padronizadas usando tecnologias como HTML5 e Javascript.

replicação bi-direcional e enfileiramento de dados *offline* para aplicações MobiForms. O MobiForms Sync Server pode ser hospedado em servidores próprios e oferece suporte a inúmeros bancos de dados relacionais famosos, incluindo MySQL. A Figura 4 ilustra a arquitetura do Sync Server.

Figura 4: Arquitetura do MobiForms Sync Server



Fonte: MobiForms (2013)

4 METODOLOGIA DE PESQUISA

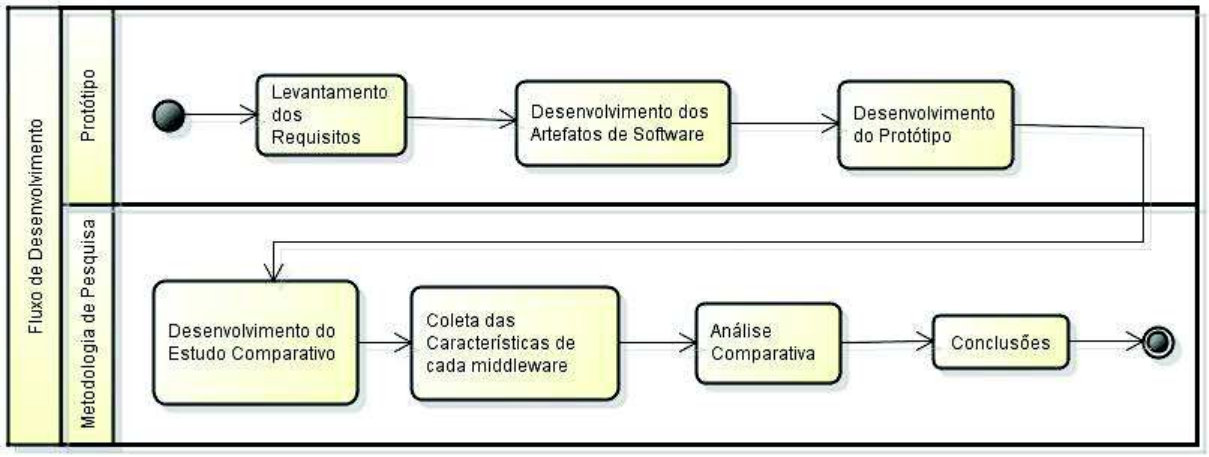
Esta seção tem como objetivo apresentar o método utilizado para realizar o estudo comparativo entre o protótipo proposto com as principais soluções existentes no mercado: Oracle Database Mobile Server e SAP Sybase SQL Anywhere. Neste cenário, procurou-se observar as especificações técnicas e características relatadas pelos fabricantes para seus produtos como requisitos funcionais. Para o estudo comparativo foi desenvolvida uma lista de características para análise se estão presentes em cada solução, com o objetivo de comparar os benefícios da adoção do *middleware* proposto, portanto, a abordagem da pesquisa é qualitativa. Com base nos objetivos do protótipo, seus recursos e as preocupações mais comuns ao desenvolvimento de aplicações móveis que consumam bancos de dados remotos, as características observadas foram:

- Suporte à plataforma Android;
- Suporte ao banco de dados MySQL;
- Motor de Sincronização integrado;

- Plataforma extensível para outros bancos e plataformas móveis;
- Código fonte aberto (*open source*);
- Suporte *offline* (sem acesso de rede);
- Servidor multiplataforma (Windows e Linux);

A seguir, é apresentado o fluxo da metodologia, assim como as etapas de desenvolvimento do protótipo.

Figura 5: Fluxo da metodologia



Fonte: Elaborado pelo autor.

5 METODOLOGIA DE DESENVOLVIMENTO

Nesta seção é apresentada a metodologia de desenvolvimento do protótipo de *middleware* Android, com o objetivo de permitir acesso móvel ao SGBD MySQL. São apresentadas as ferramentas utilizadas, a estrutura da biblioteca de classes criada, a descrição do desenvolvimento e os diagramas do protótipo, usando notação UML 2.0.

5.1 Ferramentas e Tecnologias Utilizadas

Para o desenvolvimento da biblioteca de classes, do agente móvel e da aplicação Android, foi utilizada a linguagem orientada a objetos Java e a plataforma Android, versão 2.2 (codinome Froyo). O ambiente de programação utilizado foi a IDE Eclipse versão Juno, com ADT Bundle instalado (*Android Development Toolkit*).

Para o desenvolvimento do agente servidor e das aplicações console cliente e servidor, também utilizada para testes, foi utilizada a linguagem Java, versão 1.7, atuando diretamente com o SGBD MySQL, versão 5, hospedado em servidor Linux compartilhado. O ambiente de programação utilizado foi a IDE Eclipse com a biblioteca JDBC de conexão MySQL referenciadas ao projeto.

Apesar do desenvolvimento ter sido focado nas tecnologias Android e MySQL, a utilização dos agentes é independente e o *middleware* pode ter qualquer uma de suas partes facilmente substituídas por outra implementação (usar o cliente com banco SQL Server, ou um cliente iOS com MySQL, por exemplo), uma vez que ambos agentes trabalham trocando requisições em formato JSON[9] via *sockets*[10] e com tecnologia Java (suportada pela maioria das plataformas móveis). Bastaria, portanto, construir o novo agente respeitando as características do *middleware* para que o funcionamento do mesmo se mantivesse. A escolha das tecnologias adotadas foi devido à experiência do desenvolvedor e a adoção do mercado.

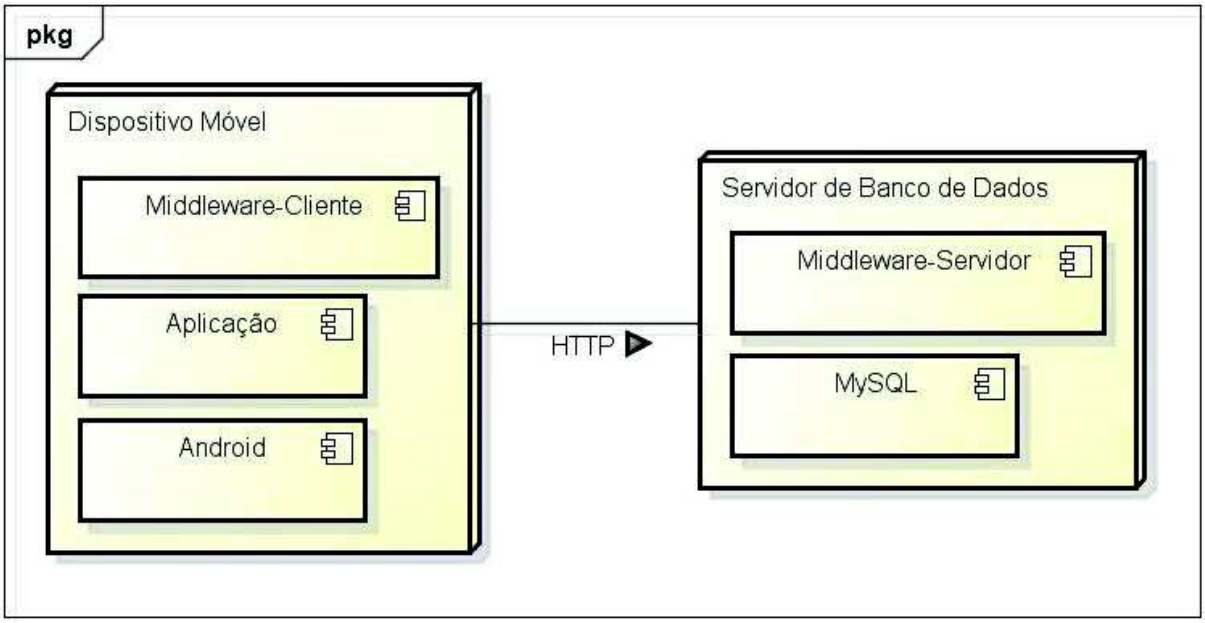
5.2 O Protótipo

Com base na arquitetura cliente-agente, agente-servidor para acesso móvel a bancos de dados e fortemente inspirado pelas soluções comerciais com a mesma finalidade, foi construído o *middleware open-source* OpenMid. Entretanto, ao contrário de suas contra-partes proprietárias e comerciais, o *middleware* prototipado é muito simples e contém somente os recursos essenciais, devendo ser expandido e customizado conforme as necessidades de cada configuração de cliente/servidor, como será visto mais adiante. A Figura 6 mostra a arquitetura final do protótipo, com as partes que o compõem.

[9] JSON (2013) ou JavaScript Object Notation (Notação de Objetos Javascript) é um formato de troca de dados leve, fácil para humanos máquinas entenderem.

[10] Segundo IBM (2013), um socket é um ponto de conexão para comunicações (endpoint) que se pode nomear e endereçar em uma rede de computadores.

Figura 6: Arquitetura do protótipo



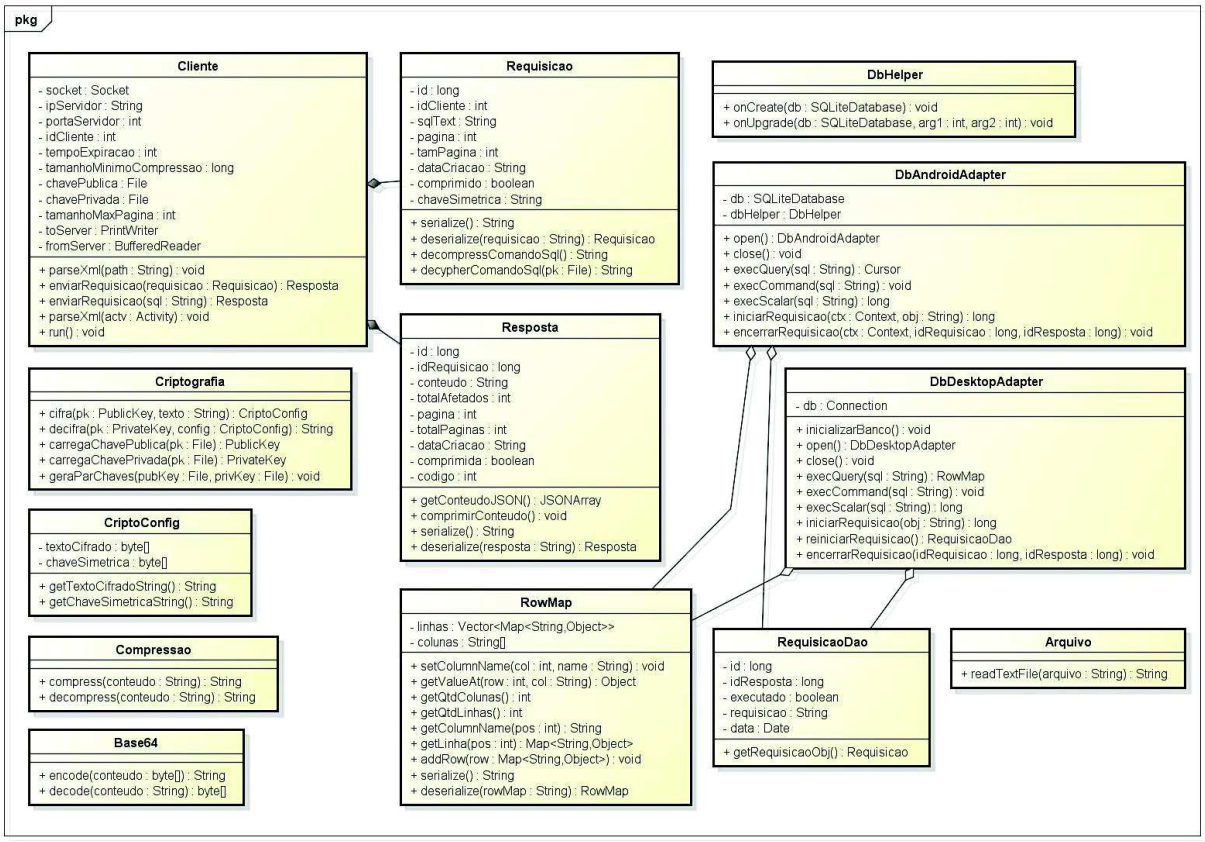
Fonte: Elaborado pelo autor.

O projeto do *middleware* é composto por bibliotecas de classes e exemplos de clientes Android e Windows, bem como de um servidor que integra-se ao SGBD MySQL. Dessa forma, atinge-se uma utilidade ainda maior para o trabalho, que não restringe-se à execução dos objetivos propostos como deixa brecha para trabalhos futuros e implementações de aplicativos móveis de mercado que necessitem se conectar com seus bancos de dados remotos. Nas subseções seguintes, são explorados os referidos componentes que compõem o protótipo.

5.2.1 Biblioteca de Classes

A biblioteca OpenMid.Framework contém as classes básicas do *middleware*, sendo utilizada como referência tanto para o desenvolvimento de clientes quanto para servidores, permitindo a extensão e utilização de suas classes, em aplicações Android. O desenvolvimento de um cliente usando esta biblioteca é detalhado mais adiante. A Figura 7 mostra o diagrama de classes desta biblioteca:

Figura 7: Diagrama de classes da biblioteca OpenMid.Framework

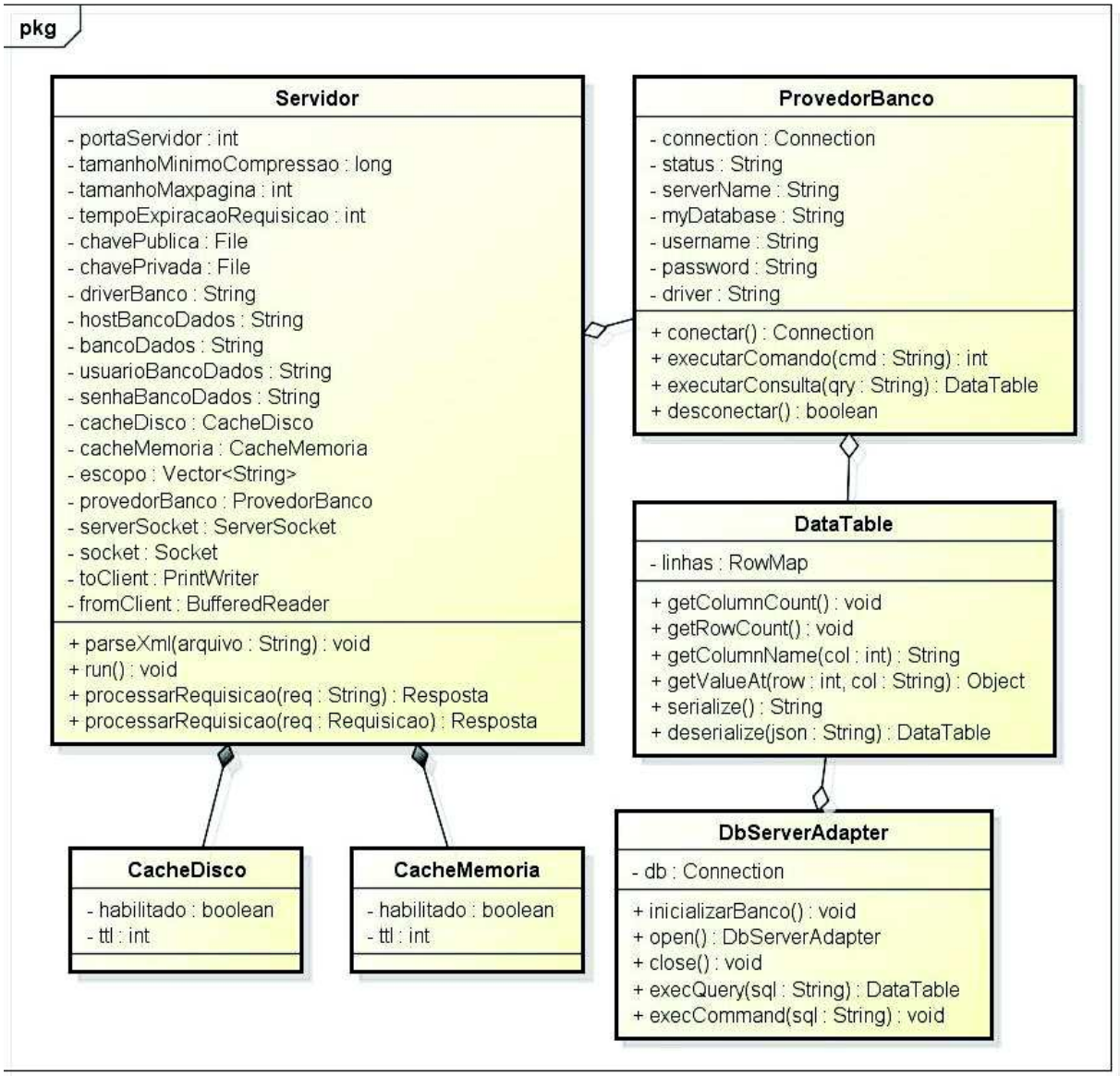


Fonte: Elaborado pelo autor.

A biblioteca apresentada na Figura 7 é divida em três partes: dados, utilitários e o framework. Na camada de dados tem-se a classe DbHelper, responsável pela criação e atualização do banco de dados local, usados nos dispositivos móveis para registro das requisições realizadas e não realizadas, enquanto que as classes DbDesktopAdapter e DbAndroidAdapter são responsáveis por manipular o mesmo banco de dados em ambiente desktop Windows (usado em testes) e Android, respectivamente. Na camada de utilitários tem-se classes para manipular arquivos (Arquivo), serialização de binários (Base64), criptografia (CriptoConfig e Criptografia), compressão de dados (Compresso) e armazenamento de tabelas em memória (RowMap), úteis tanto a clientes quanto aos servidores construídos com o *middleware*. E por fim, o framework contém as classes de Requisicao, Resposta e a classe Cliente, que é *interface* de comunicação dos aplicativos Android com o *middleware* e, consequentemente, o SGBD.

Para os servidores, foi desenvolvida uma extensão da biblioteca padrão, com algumas classes exclusivas para tal finalidade, que inclusive exigem referências adicionais aos componentes de terceiros, para o acesso a dados propriamente dito, direto no banco relacional. Este desacoplamento entre o framework central e esta biblioteca mais específica visa obter maior flexibilidade em implementações futuras, bem como a diminuição do tamanho do cliente, característica importante considerando o cenário móvel estudado anteriormente. Um exemplo de servidor prototipado com esta biblioteca é citado mais adiante, que deve ser instalado e devidamente configurado no servidor de banco de dados (a princípio MySQL), para ficar recebendo as requisições de sua contra-parte. A Figura 8 mostra o diagrama de classes desta biblioteca, mostrando as entidades que a compõem.

Figura 8: Diagrama de classes da biblioteca OpenMid.Server



Fonte: Elaborado pelo autor.

A biblioteca usada para a construção de servidores contém basicamente a classe Servidor, responsável por receber e processar as requisições enviadas pelos clientes. Para realizar tal função é necessário de classes adicionais, como uma classe para manipular o banco de dados local do servidor (DbServerAdapter), usado para registrar as requisições e respostas trocadas com os clientes Android, bem como os dados dos próprios clientes autorizados. Além disso, a classe ProvedorBanco para consumir os dados do SGBD e a classe DataTable para transferir os mesmos dados entre as camadas do *middleware*. E por fim, tem-se o

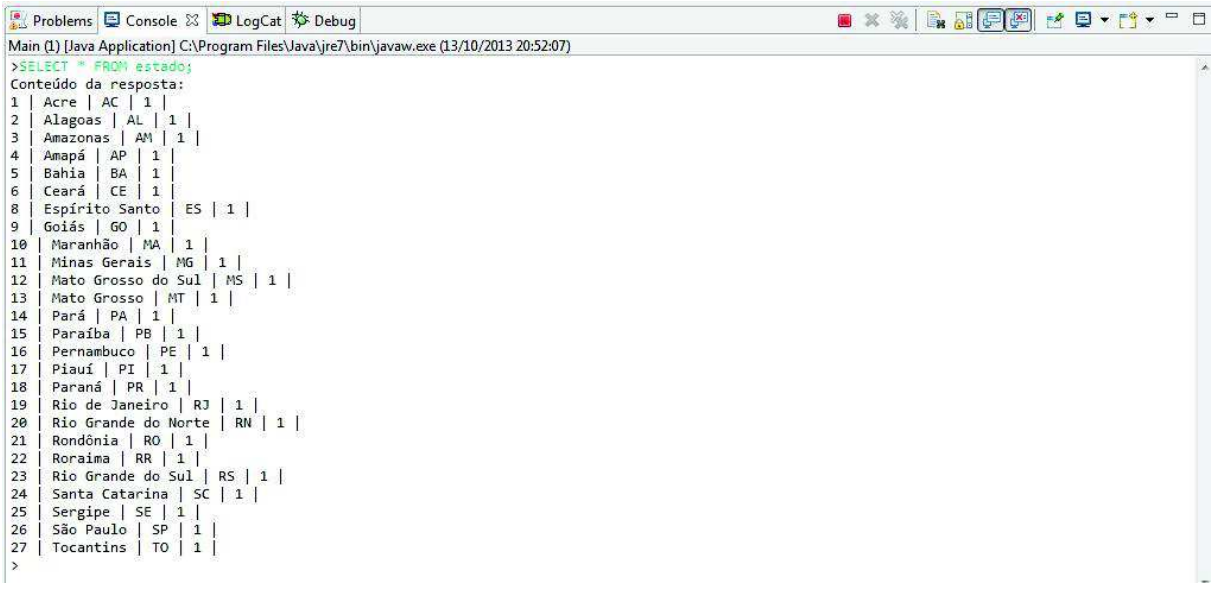
esboço de duas classes de manipulação de cache, que ainda encontram-se em desenvolvimento ao término deste artigo.

5.2.2 Agente Cliente

Ao todo foram desenvolvidos dois *softwares* clientes que estão inclusos no protótipo, inicialmente um Windows, para ser utilizado como prova de conceito, e outro para Android, mais consolidado e completo, representando o uso do *middleware* em uma aplicação rotineira que manipula dados de um banco de dados.

O cliente Windows é um aplicativo de linha de comando que, dadas as configurações presentes em um arquivo XML, conecta-se ao agente-servidor do *middleware* e permite que o usuário digite comandos SQL que são devidamente tratados e enviados ao banco de dados, recebendo em seguida a resposta em modo texto novamente, diretamente na tela de comandos. Não há tratamentos para as informações, e todos os comportamentos são descritos em modo texto para que o usuário possa acompanhar os resultados (erros, número de linhas afetadas, entre outros). A Figura 9 exibe o uso do cliente Windows dentro da IDE Eclipse para enviar uma consulta SQL para o servidor, com o retorno sendo impresso no próprio console.

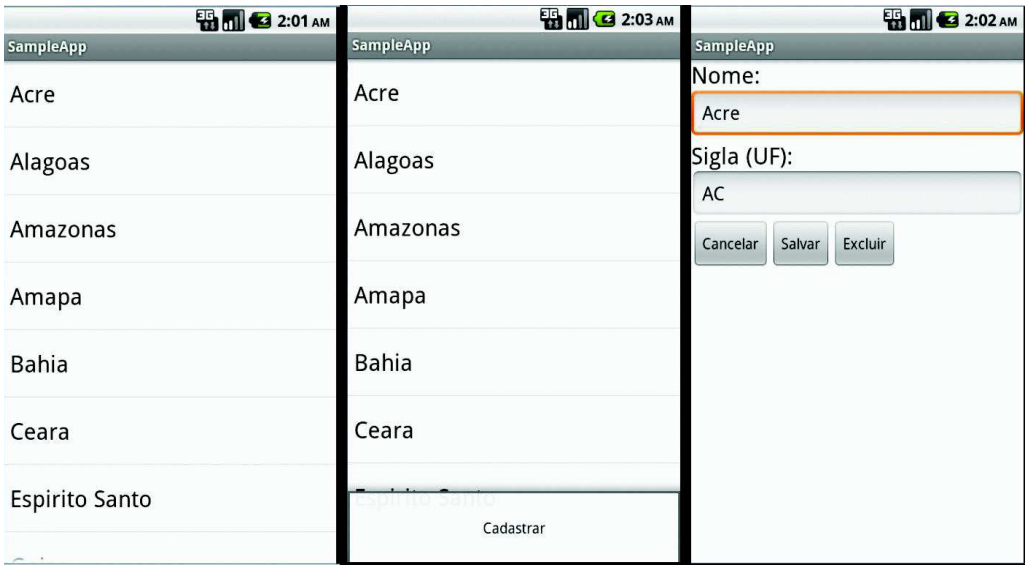
Figura 9: Cliente Windows



Fonte: Elaborado pelo autor.

O cliente Android é um aplicativo nativo para a versão 2.2 da plataforma, simulando o uso da biblioteca de classes em um aplicativo real para smartphones. O aplicativo realiza as operações básicas no banco de dados, popularmente chamadas de CRUD (acrônimo para Create, Retrieve, Update e Delete), iniciando com uma tela de listagem de uma tabela qualquer do banco de dados. Nesta tela inicial o cliente já mostra sua utilidade, sendo manipulado para retornar uma listagem de registros provenientes de um banco de dados MySQL remoto, previamente criado e populado com dados. Como funcionalidade tem-se a possibilidade de selecionar um item da lista para manipulá-lo ou escolher uma única opção de cadastrar um novo registro. Em ambas as opções abre-se um formulário que permite através do preenchimento dos respectivos campos criar, editar ou excluir registros do banco de dados. Ao término de qualquer operação, o fluxo volta para a tela de listagem que é atualizada. A Figura 10 mostra a interface da aplicação:

Figura 10: Telas do cliente Android



Fonte: Elaborado pelo autor.

Ambos os clientes possuem as mesmas funcionalidades do ponto de vista de recursos do *middleware*, incluindo o suporte às transações *offline*. Embora limitado pelo fato do *middleware* não possuir um motor de sincronização e nem mesmo uma cópia da base consolidada localmente no lado do cliente, o suporte às transações *offline* permite que, mesmo sem o dispositivo cliente estar conectado a uma rede de dados, as operações que o cliente deseja realizar no SGBD sejam agendadas

automaticamente, para serem realizadas na próxima vez que a conexão estiver disponível. Este é um recurso especialmente útil, principalmente para *softwares* comerciais em dispositivos móveis, onde permitir que a venda seja fechada mesmo sem conexão com a Internet é algo importante e que pode ser efetivado no banco de dados mais tarde, na maioria dos casos. Para registrar esses agendamentos de requisição, o *software* cria localmente um banco de dados SQLite junto ao aplicativo, que registra informações sobre as requisições. Desta forma, assim que o cliente detecta conexão com a Internet após um período sem estar *online*, busca-se no banco de dados por requisições não realizadas, dando prioridade a elas antes que novas requisições sejam feitas. Caso uma requisição não possa ser realizada, devido a um erro no comando enviado ou diferença de integridade do banco no momento em que a requisição foi agendada em comparação ao momento atual, o cliente notifica o aplicativo através de uma exceção, cabendo ao desenvolvedor adotar as medidas que julgar adequadas ao seu *software*.

5.2.3 Agente Servidor

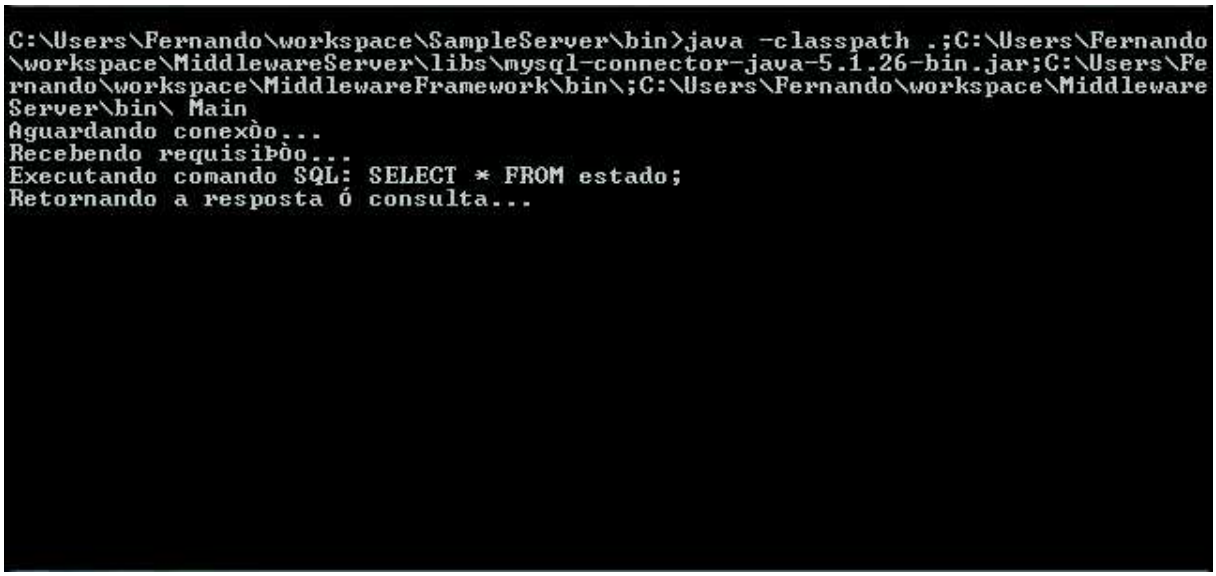
O agente servidor desenvolvido para os testes é um *software* de linha de comando que deve ser configurado previamente com as informações de acesso ao banco de dados. Desenvolvido com a linguagem Java, permite que seja executado em diversas plataformas como Linux e Windows, desde que tenha uma máquina virtual Java (JVM) instalada.

O agente servidor por si só não oferece utilidades práticas, devendo ser usado em conjunto com um servidor de banco de dados, sendo o Oracle MySQL o SGBD escolhido como alvo do protótipo por sua adoção no mercado. As configurações do servidor devem ser realizadas através de um arquivo XML auto-documentado que se encontra junto ao projeto e inclui informações de *host*, porta e credenciais de acesso ao banco de dados, entre outras configurações possíveis como o *driver* JDBC que será utilizado. Dessa forma, os clientes móveis jamais terão problemas de segurança de cada um ter as informações de acesso ao banco em sua memória, o que facilita políticas de BYOD (*Bring Your Own Device*) em empresas, bem como um controle centralizado do acesso ao banco consolidado do sistema.

O agente servidor pode ser implantado em um ambiente separado do SGBD, embora não seja recomendado tendo em vista o desempenho. Entretanto, é uma alternativa interessante caso esteja utilizando servidores em nuvem e *on-premise* ao mesmo tempo ou qualquer outro ambiente heterogêneo de servidores.

Uma vez configurado e sendo executado no servidor, o agente passará a aguardar requisições do software cliente, para processá-las. A Figura 11 exemplifica a *interface* do *software* sendo executado em ambiente Windows, no momento em que está processando uma consulta SQL.

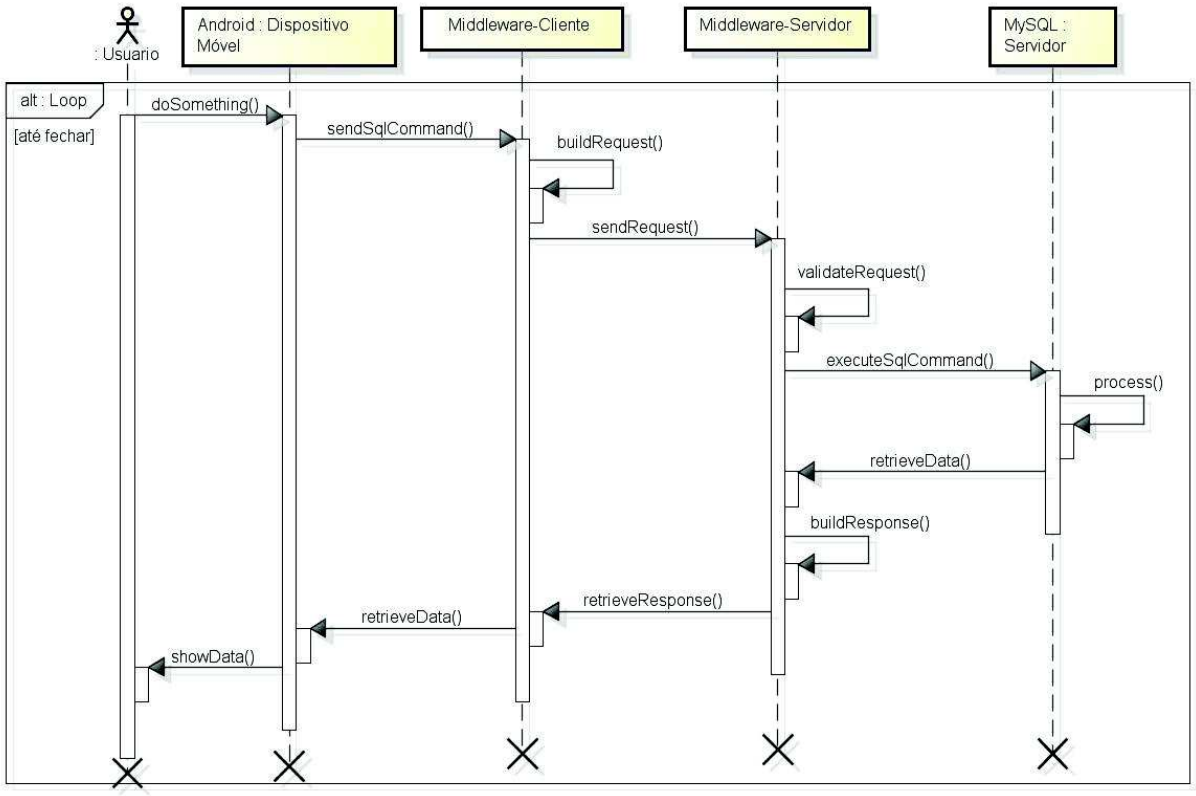
Figura 11: Imagem do agente servidor em ambiente Windows



Fonte: Elaborado pelo autor.

Uma vez que uma requisição seja disparada, o agente servidor seguirá um fluxo de processos que incluem identificação e autorização do cliente, processamento da requisição junto ao SGBD, construção da resposta e finalmente seu retorno. A Figura 12 mostra o Diagrama de Sequência de uma requisição, do cliente ao servidor e as etapas que a compõem, sendo executada com sucesso.

Figura 12: Diagrama de Sequência de uma requisição ao servidor



Fonte: Elaborado pelo autor.

Devido a uma interação do usuário com o aplicativo que está usando o *middleware*, o aplicativo dispara um comando SQL para o *middleware*, que o encapsula em um pacote criptografado que é enviado ao servidor. Uma vez que esta requisição chega no lado do servidor, os algoritmos de segurança do OpenMid validam a requisição, verificando se o destinatário é válido, se foi criptografada corretamente, se esta requisição nunca foi realizada antes e se a requisição não está velha demais para estar sendo executada. Se tudo estiver de acordo, o código SQL que veio junto com a requisição é processado junto ao SGBD e inicia-se o processo de empacotamento da resposta, incluindo a compressão de respostas muito grandes (conforme configuração do servidor) e a criptografia do pacote, para que somente o cliente que solicitou a resposta possa ler seu conteúdo. E por fim, chegando a resposta no cliente, desempacota-se os dados e exibe-se os mesmos no aplicativo, para que o usuário possa consumir a informação.

Cenários alternativos à Figura 12 incluem falha na validação da requisição, falha no processamento do comando SQL ou até mesmo falha na comunicação. Nos

dois primeiros casos, o servidor retornará uma resposta com descrição da falha e um código do tipo de erro. No terceiro caso, não há comunicação entre cliente e servidor, independente do motivo, e neste caso o cliente passa a operar em modo *offline*, o que significa que todas as requisições que o aplicativo fizer, serão enfileiradas em um *pipeline* de execução, que era executado tão logo a comunicação volte a operar normalmente.

Dentre as responsabilidades do servidor que não estão relacionadas à segurança (que serão vistas mais adiante), está o controle sobre o *cache*, a paginação e a compressão das respostas aos clientes, todas visando aumento de desempenho no uso do *middleware* pelos clientes móveis. Estes recursos são nativos do *middleware* e devem ser configurados no arquivo XML que está no servidor, podendo ser utilizados ou não, à critério do desenvolvedor.

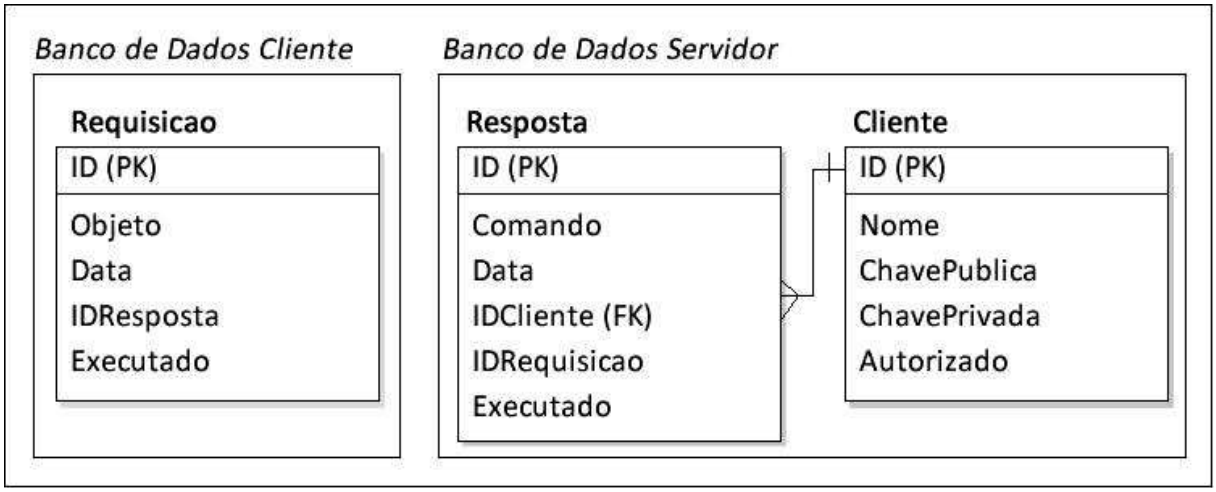
O *cache* é uma área de memória do agente servidor utilizada para armazenar o resultado de uma consulta feita ao banco de dados, para que, em uma segunda requisição aos mesmos dados, não seja necessário processar novamente os registros a serem retornados, aumentando significativamente o desempenho das requisições subsequentes. É comum em sistemas que utilizam *caching* deixar tabelas de uso comum e corriqueiro em memória, para acesso mais rápido.

E, por último, o recurso de compressão de dados que é utilizado sobre os resultados de consultas muito grandes, que poderiam causar um atraso na entrega da resposta ao cliente. Este valor é relativo à realidade de cada *software* e deve ser configurado apropriadamente, para garantir que os recursos sejam bem utilizados. Uma vez que o retorno do SGBD para uma requisição ultrapasse esse valor mínimo (por exemplo, 500KB) o servidor irá usar seu algoritmo interno de compressão para redução do tamanho total da resposta e, consequente, otimização do tempo de entrega. Uma resposta comprimida é marcada para que, quando chegar ao cliente, seja descomprimida pelo mesmo para uso no aplicativo.

5.3 Modelo de Dados

Nesta seção é apresentado o Diagrama de Entidade-Relacionamento (DER)[11], que tem como propósito apresentar o modelo de dados utilizado tanto no software cliente como no agente servidor. Para o DER foi utilizado o modelo lógico com notação IE (*Information Engineering*). Na Figura 13 está ilustrado o modelo de dados completo, separando visualmente o modelo do cliente do modelo servidor. Uma vez que os bancos de dados estão em dispositivos distintos, não há relacionamentos para garantir integridade referencial, cabendo ao *middleware* garantir a mesma.

Figura 13: Diagrama de Entidade e Relacionamento



Fonte: Elaborado pelo autor.

No *middleware* cliente é criado um banco de dados local com somente uma tabela chamada Requisicao. Cabe a esta tabela registrar as requisições efetuadas pelo aplicativo ao servidor, bem como registrar o estado da mesma. Em caso de impossibilidade de se comunicar com o servidor, nesta tabela são armazenadas as requisições que deverão ser executadas tão logo a conexão seja possível novamente. Já no lado do servidor, outro banco de dados local é criado, este com duas tabelas, sendo uma (Resposta) para armazenar as respostas enviadas às requisições dos clientes e garantir que requisições repetidas não sejam executadas

[11] Segundo Chapple (2013), um Diagrama Entidade-Relacionamento é um gráfico especializado que ilustra os relacionamentos entre entidades em uma base de dados.

e a outra, chamada Cliente, com informação sobre os clientes que possuem ou não autorização para se comunicar com o servidor, bem como o par de chaves RSA que é utilizado na criptografia entre este cliente específico e o servidor. Esta tabela é utilizada na etapa de validação da requisição recebida.

5.4 Aspectos de Segurança

Uma vez que os *middlewares* lidam com dados sensíveis dos SGBDs, deve ser dada atenção especial à segurança dos servidores que terão seus dados expostos. Este item trata exclusivamente dos aspectos levados em conta para garantir a segurança do *middleware* desenvolvido. Em especial, dois tipos de ataques foram levados em consideração, tendo em vista a probabilidade de que seriam as principais brechas a serem exploradas por atacantes: o Ataque Homem no Meio (*Man in the Middle Attack*) e o Ataque de Repetição (*Replay Attack*).

Segundo Glynn (2013), o Ataque Homem no Meio é um tipo de ataque cibernético onde um ator malicioso se insere em uma comunicação entre duas partes, se fazendo passar por ambas e ganhando acesso às informações que se tentava enviar de um lado a outro. É possível através do uso de criptografia garantir que somente o destinatário de uma mensagem trafegada na Internet possa ler o seu conteúdo, inutilizando desta forma a mensagem para o atacante. No protótipo foram utilizados dois algoritmos de criptografia juntos para garantir que a comunicação cliente-servidor não fosse explorada: um algoritmo de chave simétrica, o AES, e outro de chave assimétrica, o RSA. Enquanto que o primeiro possui bom nível de criptografia e desempenho (SMITH, 2000), há o problema do envio da chave para decrifrar a mensagem. Desta forma, usa-se o algoritmo RSA para criptografar a chave simétrica do AES, utilizada para cifrar os dados a serem enviados com o melhor desempenho possível. O servidor, ao receber a mensagem usa sua chave privada RSA para decifrar a chave simétrica do AES que por sua vez é utilizada na sequência para descriptografar a requisição em si. Embora um pouco confuso, este método é extremamente seguro, sendo empregado nas conexões seguras de navegadores de Internet (SSL) e no serviço de chamadas telefônicas privadas, como o oferecido pela empresa Kryptotel (2010).

Enquanto isso, o Ataque de Repetição, segundo Barmala (2004), é o uso de requisições previamente capturadas para atacar um sistema e obter acesso ao mesmo, uma forma de roubo de identidade. Ainda que as mensagens trafegadas estejam criptografadas, não é seguro permitir que uma mesma mensagem possa ser utilizada mais de uma vez, o que provocaria efeitos indesejados principalmente com comandos SQL que alteram os registros ou a estrutura do banco de dados. Para minimizar o risco de tais ataques, todas as requisições possuem identificadores únicos e prazo de validade. Com isso, uma requisição ao servidor não será atendida em uma segunda utilização, inutilizada para uso posterior.

Independente dos esforços para garantir o máximo de segurança aos dados confiados ao middleware vale ressaltar que algumas medidas adicionais devem ser levadas em consideração durante a implantação e uso do *middleware* sobre bancos de dados de produção. O primeiro ponto a ser observado é quanto ao uso de Redes Privadas Virtuais (VPN – *Virtual Private Network*) operando com criptografia (SSL), o que ajuda a eliminar a maioria das brechas de segurança relacionadas ao tráfego de dados sobre a Internet. O segundo ponto a ser levado em consideração é quanto à implantação de cada cliente do *middleware* com chave única (vide arquivo de configuração do cliente), o que garante não somente a fácil identificação de cada cliente no lado do servidor, quanto a facilidade em se bloquear um cliente que tenha sido comprometido (com o roubo do dispositivo do usuário, por exemplo). O terceiro e último ponto é o uso de chaves fortes, a sua troca periódica e o segredo sobre elas, tanto no lado dos clientes quanto no lado do SGBD, uma vez que o *middleware* não é capaz de bloquear requisições que estejam utilizando credenciais válidas que tenham sido fornecidas pelos usuários ou descobertas por força bruta.

6 ANÁLISE E RESULTADOS

O protótipo de middleware para acesso móvel a SGBDs foi comparado com duas soluções do mercado, a Oracle Database Mobile Server e a SAP Sybase SQL Anywhere. Para coletar os dados referentes às características de cada um foram utilizados os websites dos fabricantes e os manuais digitais dos produtos, também disponíveis nos mesmos sites (ORACLE, 2013; SYBASE, 2013). Uma vez que o estudo comparativo é puramente conceitual, o resultado da análise deve ser levado

em conta somente para fins de estudo e para que se possa situar o protótipo desenvolvido perante os trabalhos relacionados.

6.1 Análise das Características

O Quadro 1 apresenta as características utilizadas na análise em referência à existência ou não de tal recurso em cada uma das soluções.

Quadro 1: Análise das Características das Soluções

Característica	Oracle Database Mobile Server	SAP Sybase SQL Anywhere	OpenMid
Suporte à plataforma Android	Sim	Sim	Sim
Suporte à diferentes SGBDs	Não	Sim	Sim
Motor de Sincronização de Dados	Sim	Sim	Não
***Middleware* extensível para outras plataformas e SGBDs**	Não	Não	Sim
Código-fonte aberto	Não	Não	Sim
Suporte à requisições *offline*	Sim	Sim	Sim
Servidor Multiplataforma	Sim	Sim	Sim

Fonte: elaborado pelo autor.

As características analisadas refletem os objetivos que se pretendia alcançar com a implementação de um *middleware* de acesso móvel a SGBDs remotos. O suporte à plataforma Android e a SGBDs heterogêneos, iniciando com o MySQL, eram duas premissas em virtude de sua popularidade e também como diferencial por ser uma combinação de baixo custo e baixa complexidade frente aos concorrentes. O fato de ser um *software* de código aberto e que usa tecnologias igualmente abertas, permite que a solução possa ser implantada em diferentes plataformas, e ser ajustada às necessidades específicas dos desenvolvedores, além de estendida para outras plataformas móveis e SGBDs. E por fim, os suportes à sincronização de dados e a requisições *offline* foi parcialmente alcançado pelo protótipo, em virtude da complexidade do desenvolvimento de uma solução robusta e completa e também pelo próprio fato do objetivo primário do protótipo realmente não ser algo deste porte.

6.2 Conclusão da Análise

Como conclusões da análise do Quadro 1 supracitado têm-se que o OpenMid oferece boa parte das funcionalidades tradicionais dos *middlewares* para acesso móvel à SGBDs, com as vantagens de ser extensível e suportar um SGBD de baixo custo de implantação. As soluções concorrentes possuem funcionalidades adicionais que lhes proporcionam maior robustez e integridade, como o motor de sincronização de dados, inexistente no OpenMid, o que lhes dá uma vantagem em um cenário de aplicações empresariais mais robustas e/ou de missão crítica. Entretanto, uma vez que o OpenMid é *open source* e extensível, é possível que estas deficiências venham a ser sanadas no futuro em virtude de uma possível demanda em utilizá-lo.

E por fim, com base nos estudos prévios também, nota-se uma clara preocupação dos fabricantes em criar soluções adequadas às suas necessidades em particular, para facilitar o acesso de dispositivos móveis aos seus bancos proprietários e não necessariamente em fornecer uma solução mais democrática que seja heterogênea, por exemplo.

7 CONCLUSÃO

Este artigo apresentou os aspectos básicos do estudo sobre o acesso móvel a banco de dados, iniciando com uma rápida visão do cenário atual e da necessidade dos aplicativos de se conectar a bancos de dados remotos via dispositivos móveis.

Também foram vistos alguns motivos de segurança e desempenho para utilização de um *middleware* de acesso a dados e alguns problemas referentes às ferramentas proprietárias existentes, onde ficou claro que o acesso móvel a SGBDs é um assunto merecedor de atenção que está longe de ter sido esgotado.

Um estudo abrangente foi feito sobre as propostas para *middlewares* de acesso a dados existentes no mercado, culminando no desenvolvimento de protótipos de agentes cliente e servidor, bem como das aplicações de teste, elucidando o que foi estudado de forma prática. Como método de pesquisa proposto utilizou-se um estudo comparativo entre as características das principais soluções existentes com o *middleware* desenvolvido, chegando à conclusão de que a

flexibilidade e simplicidade do protótipo atendem a pequenas aplicações, enquanto que a robustez dos *middlewares* mais tradicionais são indicados para soluções de missão crítica e uso mais intenso de dados.

Além do conhecimento obtido, tem-se como produto uma biblioteca *open source* contendo todas as classes necessárias para implantação do *middleware*, tanto no lado do cliente quanto do servidor. Não obstante, a documentação criada durante o desenvolvimento possibilita trabalhos futuros utilizando acesso móvel a banco de dados como tema principal ou como ferramenta para suas aplicações, se tornando uma referência para o assunto. Dentre possíveis trabalhos futuros, pode-se estender a biblioteca para se adequar às necessidades de outras plataformas e SGBDs, bem como a características específicas de sistemas em produção.

OPENMID: AN ANDROID MIDDLEWARE FOR MOBILE DATABASES

CONTEXT: The mobile computing advances made possible to tablets and smartphones act like work environment extensions, using softwares that connect to ERPs, CRMs and generally have to deal with big databases. PROBLEM: The mobile applications development that have to access remote DBMSs show many challenges to the mobile developers, once that it depends of the device limited resources, mobile data network that isn't cover all the regions and about an architectural change in the application, because the mobile connections are unstable. SOLUTION: Because of these problems, was developed a middleware prototype for remote DBMS mobile access, that will intermediate requests and responses between the mobile platform and the DBMS. PROPOSED METHOD: This paper details the comparative study between the middleware prototype for remote DBMS mobile access and the mainstream solutions in the market that have support to the Android platform. CONCLUSION: From the prototype development and comparative study analysis, the results incentive the use of him, as future studies on this platform.

Keywords - Middleware; Mobile Databases; DBMS; Android; MySQL.

REFERÊNCIAS

ANATEL – Agência Nacional de Telecomunicações. **Avaliação Trimestral do Plano Nacional de Ação de Melhoria da Prestação do Serviço Móvel Pessoal**. Julho de 2013. Disponível em: <http://www.anatel.gov.br/Portal/documentos/sala_imprensa/26-7-2013--10h37min43s-COQL_SMP_Avaliacao.ppt>. Acesso em 13/10/2013.

BARBOSA, Alvaro C.. **Middleware para Integração de Dados Heterogêneos**. Maio de 2001. Disponível em: <ftp://ftp.inf.puc-rio.br/pub/docs/theses/01_PhD_barbosa.pdf>. Acesso em 13/10/2013.

BARMALA, Christian. **Replay Attack – Computer Definition**. 2004. Disponível em: <http://www.yourdictionary.com/replay-attack>. Acesso em 17/09/2013.

BILLARD, David. **Transactional Services for the Internet**. Março de 1998. Disponível em: <http://books.google.com.br/books?id=pSgPg1Na51UC&pg=PA14>. Acesso em 13/10/2013.

BOTTOMLEY, James. **Understanding Caching**. Janeiro de 2004. Disponível em: <http://www.linuxjournal.com/article/7105>. Acesso em 19/08/2013.

BRADEN, Robert. **Requirements for Internet Hosts – Comunication Layers**. Outubro de 1989. Disponível em: <https://tools.ietf.org/html/rfc1122#page-18>. Acesso em 01/09/2013.

CAMPBELL, Andrew T.; COULSON, Geoff; KOUNAVIS, Michael. **Managing Complexity: Middleware Explained**. Outubro de 1999. Disponível em: <http://www.cin.ufpe.br/~redis/intranet/bibliography/middleware/campbell-managing-1999.pdf>. Acesso em 13/10/2013.

CHAPPLE, Mike. **Entity-Relationship Diagram**. 2013. Disponível em: <http://databases.about.com/cs/specificproducts/g/er.htm>. Acesso em 07/10/2013.

GLYNN, Fergal. **Man in the Middle Attack**. 2013. Disponível em: <http://www.veracode.com/security/man-in-the-middle-attack>. Acesso em 13/10/2013.

IBM – International Business Machines. **Socket programming**. 2013. Disponível em: <http://pic.dhe.ibm.com/infocenter/iseries/v7r1m0/index.jsp?topic=%2Frzab6%2Fhowdosockets.htm>. Acesso em: 18/09/2013.

IPSOS OTX MEDIACT. **Nosso Planeta Mobile: Brasil**. Maio de 2012. Disponível em: < http://services.google.com/fh/files/blogs/our_mobile_planet_brazil_pt_BR.pdf>. Acesso em 19/08/2013.

JANSSEN, Cory. **Data Security**. 2013. Disponível em: <http://www.techopedia.com/definition/26464/data-security>. Acesso em 18/09/2013.

JSON. **Introducing JSON**. 2013. Disponível em: <http://www.json.org/>. Acesso em 18/09/2013.

KATZ, Jonathan; LINDELL, Yehuda. **Introduction to Modern Cryptography**. Agosto de 2007. Disponível em: <http://www.cs.umd.edu/~jkatz/imc/chap1.pdf>. Acesso em 18/08/2013.

KESSLER, Gary. **Encryption and Decryption Algorithm**. Maio de 1998. Disponível em: <http://homepages.uel.ac.uk/u0430614/Encryption%20index.htm>. Acesso em 18/08/2013.

KRAKOWIAK, Sacha. **What is Middleware**. 2003. Disponível em: <http://middleware.objectweb.org/>. Acesso em 18/08/2013.

KRYPTOTEL. **RSA Algorithm (Rivest Shamir Adleman)**. 2010. Disponível em: <http://en.kryptotel.net/rsa.html>. Acesso em 14/10/2013.

LINTHICUN, David. **Database-oriented middleware**. Junho de 2001. Disponível em: <http://searchsoa.techtarget.com/answer/Database-oriented-middleware>. Acesso em 18/08/2013.

MOBIFORMS. **MobiForms Mobile Application Development Tools**. 2013. Disponível em: <http://www.mobiforms.com/>. Acesso em: 18/09/2013.

NELSON, Mark; GAILLY, Jean-Loup. **The Data Compression Book**. Dezembro de 1995. Disponível em: <http://staff.uob.edu.bh/files/781231507_files/The-Data-Compression-Book-2nd-edition.pdf>. Acesso em 01/09/2013.

OPENMOBSTER. **OpenMobster – Open Source Mobile Cloud Platform**. 2012. Disponível em: <http://www.openmobster.com/product.html>. Acesso em: 18/09/2013.

ORACLE. **Introduction to Transactions**. 2011. Disponível em: <http://docs.oracle.com/cd/E11882_01/server.112/e10713/transact.htm>. Acesso em 19/08/2013.

ORACLE. **Mobile Client Overview**. 2011. Disponível em: <http://docs.oracle.com/cd/E22663_01/doc.11100/e22681/soverview.htm>. Acesso em 22/09/2013.

ORACLE. **Oracle Database Mobile Server 11g**. 2013. Disponível em: <http://www.oracle.com/technetwork/products/database-mobile-server/overview/index.html>. Acesso em 02/09/2013.

ORACLE. **Oracle Database Mobile Server: Mobile Client Guide**. Setembro de 2011. Disponível em: <http://docs.oracle.com/cd/E22663_01/doc.11100/e22681.pdf>. Acesso em 02/09/2013.

OSI - Open Source Initiative.**The Open Source Definition**. Disponível em: <http://opensource.org/osd>. Acesso em 17/09/2013.

PHONEGAP. **Phonegap**. Disponível em: <http://phonegap.com/>. Acesso em 18/09/2013.

RIGO, Sandro José. **Introdução à Plataforma Web para Dispositivos Móveis**. Disponível em: <http://www.moodle.unisinos.br/file.php/4783/INTRODUCAO_A_PLATAFORMA_WEB_PARA_DISPOSITIVOS_MOVEIS/Sandro_Intro_plataforma_WEB_texto.pdf>. Acesso em 18/09/2013.

SATYANARAYANAN, Mahadev. **Fundamental Challenges in Mobile Computing**. 1995. Disponível em: <http://www.cs.cmu.edu/~coda/docdir/podc95.pdf>. Acesso em 19/08/2013.

SHANNON, Claude. **A Mathematical Theory of Communication**. Outubro de 1948. Disponível em: <http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf>. Acesso em 01/09/2013.

SMITH, Kaleigh. **Rijndael: Advanced Encryption Standard**. Novembro de 2000. Disponível em: <http://www.cs.mcgill.ca/~kaleigh/computers/crypto_rijndael.html>. Acesso em 14/10/2013.

STOKES, Jon. **Pipelining: An Overview**. Setembro de 2004. Disponível em: <http://arstechnica.com/features/2004/09/pipelining-1/>. Acesso em 19/08/2013.

SYBASE**. MobiLink synchronization**. 2013. Disponível em: <http://dcx.sybase.com/index.html#sa160/en/mlstart/ml-basics.html>. Acesso em 03/09/2013.

SYBASE. **Parts of a MobiLink application**. 2013. Disponível em: <http://dcx.sybase.com/index.html#sa160/en/mlstart/parts-ml-basics.html>. Acesso em 23/09/2013.

SYBASE. **SAP Sybase SQL Anywhere**. 2013. Disponível em: <http://www.sybase.com.br/products/databasemanagement/sqlanywhere>. Acesso em 03/09/2013.

W3C Working Group. **Web Services Glossary**. Fevereiro de 2004. Disponível em: <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>. Acesso em 20/08/2013.