

**UNIVERSIDADE FEDERAL DO PAMPA**

**MARCEL DA SILVA CAMARGO**

**PROPOSTA DE UMA ABORDAGEM DE SINCRONIZAÇÃO MULTIMASTER  
UTILIZANDO DISPOSITIVOS MÓVEIS**

**Bagé  
2013**

**MARCEL DA SILVA CAMARGO**

**PROPOSTA DE UMA ABORDAGEM DE SINCRONIZAÇÃO MULTIMASTER  
UTILIZANDO DISPOSITIVOS MÓVEIS**

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Especialista.

Orientador: Sandro da Silva Camargo

**Bagé  
2013**

**MARCEL DA SILVA CAMARGO**

**PROPOSTA DE UMA ABORDAGEM DE SINCRONIZAÇÃO MULTIMASTER  
UTILIZANDO DISPOSITIVOS MÓVEIS**

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Especialista.

Trabalho de Conclusão de Curso defendido e aprovado em: 02 de Agosto de 2013.

Banca examinadora:

---

Prof. Dr. Sandro da Silva Camargo  
Orientador  
UNIPAMPA

---

Prof.<sup>a</sup> Dr.<sup>a</sup> Ana Paula Lüdtkke Ferreira  
UNIPAMPA

---

Prof. Dr. Carlos Emílio Padilha Severo  
IFSUL

Dedico este trabalho aos meus pais que sempre priorizaram minha educação e são meus exemplos de vida. Aos meus irmãos pelo constante apoio e incentivo. A minha esposa, pelo amor, carinho e compreensão. Ao orientador pela incansável ajuda e orientação, e a todos os professores do curso e funcionários da UNIPAMPA.

"O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis."

José de Alencar

## RESUMO

O uso de tecnologias sem fio e móveis pela atual sociedade é uma realidade. A capacidade de controlar as características associadas a mobilidade oferece grandes oportunidades para o desenvolvimento de soluções mais inteligentes. Atualmente na engenharia de uma aplicação de *software*, grande parte dos esforços em pesquisa é dedicada a modelos, conceitos e ferramentas que compreendam os dados móveis e os tornem viáveis em uma aplicação distribuída. Este trabalho tem como objetivo a criação de uma ferramenta de *software* para dispositivos móveis que permita o acesso em tempo real aos dados do sistema, mantendo-o consistente, visando alta disponibilidade, por meio de técnicas de sincronização de banco de dados *multimaster*, utilizando a ferramenta *SQLyog Job Agent* (SJA). Além disso, também será implementado uma aplicação de controle de credenciamento de participantes em eventos acadêmicos. Como resultado, foram realizados testes de interação com o banco de dados, analisando a sincronização do ambiente proposto e observando a persistência dos dados e a alta disponibilidade.

Palavras-chave: Sincronização, banco de dados, dispositivos móveis.

## **ABSTRACT**

The wireless and mobile technologies use by the current society is a reality. The ability to control the characteristics associated with mobility offer great opportunities for the smarter solutions development. Currently in software application engineering, most research effort is devoted to models, concepts and tools to understand mobile data and become viable in a distributed application. This work aims the software tool creation for mobile devices that allows access to real-time data from the system, keeping it consistent, aiming high availability through synchronization techniques for the multimaster database, using the tool SQLyog Job Agent (SJA). In addition, also development a control application for participants accreditation in academic events. As a result, were performed interact tests with the database, analyzing the proposed environment synchronization and observing the data persistence and high availability.

**Keywords:** Synchronization, database, mobile devices.

## LISTA DE FIGURAS

|   |    |
|---|----|
| Figura 1 - Assinatura mundial de serviços móveis.....                     | 17 |
| Figura 2 - Crescimento do mercado de smartphones. ....                    | 18 |
| Figura 3 - Quota de mercado por sistema operacional. ....                 | 20 |
| Figura 4 - Banco de dados distribuído.....                                | 22 |
| Figura 5 - Sincronização <i>one-way</i> . ....                            | 26 |
| Figura 6 - Sincronização <i>two-way</i> . ....                            | 26 |
| Figura 7 - Visão geral da aplicação. ....                                 | 27 |
| Figura 8 - Uso dos serviços de servidores web. ....                       | 30 |
| Figura 9 - SGDBs mais populares. ....                                     | 31 |
| Figura 10 - Uso de linguagens de programação web <i>server-side</i> ..... | 33 |
| Figura 11 - Estrutura do arquivo XML.....                                 | 34 |
| Figura 12 - Tela inicial do AndroPHP.....                                 | 36 |
| Figura 13 - Diagrama de sequência da sincronização. ....                  | 37 |
| Figura 14 - Modelo Entidade-Relacionamento. ....                          | 39 |
| Figura 15 - Interfaces iniciais da aplicação. ....                        | 40 |
| Figura 16 - Interfaces de cadastro.....                                   | 40 |
| Figura 17 - Interface de seleção de dispositivos.....                     | 41 |
| Figura 18 - <i>Log</i> da sincronização. ....                             | 42 |

## **LISTA DE TABELAS**

|   |    |
|---|----|
| Tabela 1 - Sistemas operacionais móveis.....                                | 19 |
| Tabela 2 - Comparativo entre as formas de banco de dados distribuídos. .... | 24 |
| Tabela 3 - Configuração do servidor. ....                                   | 29 |
| Tabela 4 - Configuração dos dispositivos móveis.....                        | 29 |

## LISTA DE SIGLAS

- API - *Application programming interface* (Interface de programação de aplicações)
- ASP - *Active server pages*
- BD - Banco de dados
- CGI - *Common Gateway Interface* (Interface de acesso comum)
- CSS - *Cascading Style Sheets*
- DBA - *Database administrator* (Administrador de banco de dados)
- DBMS - *Database management system* (Sistema de gerenciamento de banco de dados)
- DDB - *Distributed database* (Banco de dados distribuído)
- DER - Diagrama entidade relacionamento
- HTTP - *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)
- LDBMSs - *Local database management system* (Sistema de gerenciamento de banco de dados local)
- PHP - *Hypertext Preprocessor*
- PK - *Primary key* (Chave primária)
- SDK - *Software development kit* (Kit de desenvolvimento de *software*)
- SGBD - Sistema de gerenciamento de banco de dados
- SJA - *SQLyog job agent* (Agente de tarefas SQLyog)
- XML - *Extensible markup language*

## SUMÁRIO

|  |    |
|--|----|
| 1 INTRODUÇÃO .....                               | 12 |
| 1.1 Justificativa .....                          | 13 |
| 1.2 Problema de pesquisa.....                    | 13 |
| 1.3 Objetivos.....                               | 14 |
| 1.4 Metodologia.....                             | 14 |
| 1.5 Estrutura do trabalho.....                   | 14 |
| 2 FUNDAMENTAÇÃO TEÓRICA.....                     | 16 |
| 2.1 Computação móvel .....                       | 16 |
| 2.2 Sistemas operacionais móveis.....            | 18 |
| 2.2.1 Sistema operacional <i>Android</i> .....   | 19 |
| 2.3 Banco de dados .....                         | 20 |
| 2.3.1 Banco de dados distribuído.....            | 22 |
| 2.4 Sincronização de dados.....                  | 24 |
| 2.4.1 Tipos de Sincronização.....                | 25 |
| 3 PROPOSTA.....                                  | 27 |
| 3.1 Tecnologias utilizadas .....                 | 28 |
| 3.2 Ferramentas de implementação .....           | 29 |
| 3.2.1 Apache .....                               | 30 |
| 3.2.2 MySQL .....                                | 31 |
| 3.2.3 PHP.....                                   | 32 |
| 3.2.4 SJA.....                                   | 33 |
| 3.2.5 AndroPHP.....                              | 35 |
| 3.3 Projeto da aplicação .....                   | 36 |
| 3.3.1 Estudo de caso.....                        | 37 |
| 3.3.2 Problema.....                              | 38 |
| 3.3.3 Definição das fronteiras da aplicação..... | 38 |
| 3.4 Interfaces da aplicação.....                 | 39 |
| 3.5 Testes .....                                 | 41 |
| 4 CONCLUSÕES E CONSIDERAÇÕES FINAIS.....         | 43 |
| REFERÊNCIAS .....                                | 44 |

## 1 INTRODUÇÃO

O uso de dispositivos móveis cresce vertiginosamente nos últimos anos. Um de seus principais benefícios é a conveniência que é oferecida, permitindo aos usuários carregarem consigo as informações de que necessitam enquanto se deslocam ao longo do dia.

Os recentes avanços na tecnologia e equipamentos móveis têm levado ao surgimento de um novo ambiente de computação e a popularização de uma variedade de dispositivos de pequenos porte móveis, como PDAs (personal digital assistants), celulares inteligentes, HPCs (computadores portáteis), Pocket PCs e *Tablets*. Como várias tecnologias de rede são cada vez mais associadas a esses dispositivos móveis, o processamento de informações dos usuários pode estar disponível através dos dispositivos móveis. Como resultado, os modelos de negócios que dependem de tecnologias móveis estão crescendo (Ramya et al., 2012).

Os sistemas de computação móvel são sistemas computacionais que podem ser movidos fisicamente e cujas capacidades de computação necessitam ser utilizadas enquanto estão sendo movidos. Para distinguir os sistemas de computação móvel de outros sistemas de computação, podemos identificar as diferenças nas tarefas que eles são projetados para executar, a maneira que eles são projetados, e o modo em que são operados. Há muitas funções que um sistema de computação móvel pode fazer e um sistema de computação estacionário não pode fazer, estas funcionalidades agregadas são a razão para caracterizar separadamente os sistemas de computação móvel (B'Far, 2005).

Uma funcionalidade essencial é a capacidade de reconciliar facilmente os dados entre os dispositivos móveis e a fonte de dados que reside em um ponto central, ou seja, não móvel.

Os dispositivos móveis permanecem desconectados da rede a maior parte do tempo. Desta forma, as técnicas de transação não podem ser utilizadas para alcançar replicação consistente entre os dispositivos móveis e um servidor central. Em vez disso, normalmente é implementada uma forma mais fraca de controle de consistência, chamada de sincronização. Durante o tempo em que um dispositivo está desconectado da rede, os dados podem ser criados, modificados ou excluídos tanto no dispositivo quanto no servidor central. Quando a conexão de rede é reestabelecida entre os dois pontos, a sincronização pode ser invocada para reconciliar os dados entre o dispositivo e o servidor (Denny; Franklin, 2004).

Os dispositivos móveis não têm um grande poder de computação e dependem de baterias. Além disso, o acesso constante a rede é difícil devido a limitação de largura de banda. Por conseguinte, não é fácil de processar um grande volume de dados armazenados e manter uma conexão contínua com a base de dados localizada em um servidor. Por estas razões, os

dispositivos móveis precisam de bases de dados móveis, a fim de conseguir o processamento de dados estável. Os dispositivos móveis atualizam uma porção limitada de um banco de dados do servidor conectado através de uma técnica de sincronização, que tem uma função de comunicação estável. Assim os dispositivos móveis processam suas tarefas usando os dados atualizados em um estado *off-line*. O trabalho sobre as condições de falta de conexão com a rede é um ponto crucial para o apoio à mobilidade (Ramya et al., 2012).

O presente trabalho tem como objetivo a implementação de uma aplicação de controle de credenciamento para eventos acadêmicos, utilizando dispositivos móveis para execução das tarefas de credenciamento dos participantes do evento e apresentando uma solução para a sincronização de dados entre o banco de dados (BD) dos dispositivos móveis e um banco de dados em um servidor central, visando uma maior velocidade na inserção dos dados dos participantes e na emissão de certificados após a conclusão do evento.

## **1.1 Justificativa**

Em muitos domínios de aplicação o trabalho precisa ser desenvolvido em situações onde a mobilidade é uma característica crucial. As aplicações de *software* tem por apelo, tipicamente, disporem de estruturas de BDs para armazenar os fatos. Estes BDs estão hospedados em servidores e acessados normalmente a partir de equipamentos instalados em pontos fixo da rede computacional da organização. Tal circunstância provoca inúmeras dificuldades operacionais aos usuários, forçando-os a fazer registros temporários manuais para posteriormente serem digitados nos sistemas.

Uma visão das tecnologias atuais permite considerar a possibilidade de contar com dispositivos móveis, muito populares, seguros e flexíveis, para satisfazer tais necessidades e fazerem a integração em tempo real dos dados inseridos nos dispositivos móveis com um BD localizado em um servidor central, usando múltiplas técnicas, destacando-se a sincronização.

A solução estudada neste trabalho compatibiliza o melhor dos dois mundos: a mobilidade do usuário e a alta disponibilidade dos dados.

## **1.2 Problema de pesquisa**

A eliminação de duplicidade de dados, a exigência de integração dos processos e a alta disponibilidade das informações em qualquer aplicação de *software* projetada para dispositivos móveis, que utilize banco de dados distribuído é de extrema importância para o bom funcionamento e eficiência da aplicação. Porém, pela dificuldade imposta por esse tipo de

solução, faz-se necessária uma evolução nas técnicas de sincronização *multimaster* aplicáveis a banco de dados distribuídos.

Elicitar os recursos e características necessários para implementar uma aplicação de *software* orientada a dispositivos móveis de interação com banco de dados distribuídos.

### 1.3 Objetivos

- Geral:
  - Implementar uma aplicação de *software* para dispositivos móveis, como um nó de um bando de dados distribuídos, permitindo acessá-lo em tempo real mantendo-o consistente, por meio de técnicas de sincronização.
- Específicos:
  - Revisão de literatura sobre o problema.
  - Implementar uma aplicação para utilização em dispositivos móveis (nós).
  - Conectar ao banco de dados do dispositivo móvel em tempo real.
  - Análise de abordagens de sincronização.
  - Sincronizar os nós, garantindo a persistência de dados.
  - Analisar a sincronização do ambiente.

### 1.4 Metodologia

Baseado nas publicações de autores da área de computação móvel, banco de dados e desenvolvimento web, a estrutura será apresentada da seguinte forma: será realizada uma pesquisa exploratória com busca, seleção e leitura de artigos e publicações das áreas acima mencionadas, um estudo de caso para a aplicação de *software*, a configuração do ambiente proposto e implementação da aplicação utilizando ferramentas para assegurarem a persistência de dados.

### 1.5 Estrutura do trabalho

Este trabalho está organizado da seguinte forma: o capítulo 2 apresenta uma revisão bibliográfica baseada em publicações da área, contextualizando uma arquitetura de computação móvel e suas principais características, descrevendo os principais sistemas operacionais da atualidade, focalizando sobre o sistema operacional *Android*, apresentando também definições sobre banco de dados e banco de dados distribuídos e exibindo as principais técnicas de sincronização de dados existentes. O capítulo 3 aborda a proposta do trabalho desenvolvido,

descrevendo a metodologia de desenvolvimento, a estrutura de um ambiente de desenvolvimento web juntamente com as ferramentas de *software* utilizadas na implementação da solução, declarando todos os processos envolvidos na implementação da aplicação de *software*, tais como, o estudo de caso, a definição do problema e das fronteiras do sistema, finalizando o capítulo com os resultados dos testes realizados. O capítulo 4 discute as conclusões finais sobre o trabalho desenvolvido e identifica alguns trabalhos futuros a serem realizados com base no conhecimento adquirido e em outros problemas em potencial a serem resolvidos.

## **2 FUNDAMENTAÇÃO TEÓRICA**

O objetivo deste capítulo é de apresentar a fundamentação teórica com base nos principais autores, argumentos e conceitos da área abordada, exibindo uma contextualização do ambiente que será proposto para que corroborem com a proposta do trabalho em questão.

Primeiramente são apresentadas as definições de computação móvel e suas principais características, posteriormente descreve os principais sistemas operacionais móveis, apresenta definições sobre banco de dados e banco de dados distribuídos e exibe as principais técnicas de sincronização de dados existentes. O capítulo se encerra com a apresentação da estrutura e ferramentas do ambiente de desenvolvimento web utilizado.

### **2.1 Computação móvel**

A grande evolução tecnológica dos últimos anos levou os usuários a um ponto onde é praticamente possível acessar suas informações em qualquer lugar e a qualquer momento. De acordo com pesquisas relacionadas, há um aumento diário no número de usuários que passam a utilizar computação móvel, tornando os dispositivos móveis parte de sua rotina. A computação móvel propicia uma ampliação do domínio da computação distribuída, pois elimina a limitação de conexão física à rede, possibilitando a mobilidade dos dispositivos.

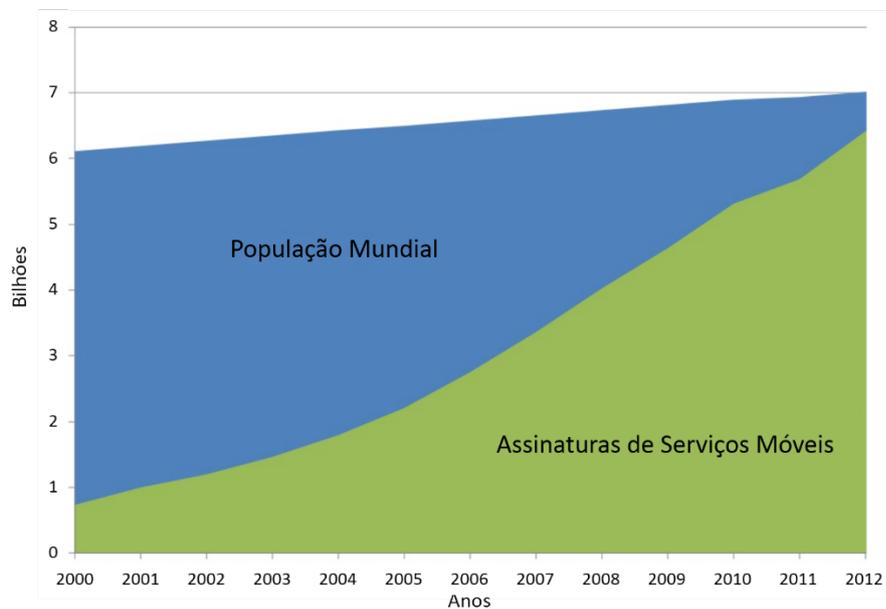
Em um ambiente de computação onde através de um dispositivo portátil seja possível se comunicar com a parte fixa da rede e com outros computadores móveis, dá-se o nome de computação móvel ou computação nômade (Tonin, 2012).

Um sistema de computação móvel, como com qualquer outro tipo de sistema de computação, pode ser conectado a uma rede. A conectividade com a rede, no entanto, não é um pré-requisito para ser um sistema de computação móvel. Datada do final dos anos 1960, a rede permitiu computadores conectar-se uns com os outros. Colocar em rede dois ou mais computadores requer algum meio que permite que os sinais sejam trocados entre eles. Este meio foi tipicamente alcançado através de redes cabeadas. Embora as redes com fio permaneçam como o método predominante de conexão de computadores, elas possuem limitações para a conexão de dispositivos de computação móvel. Não só a disponibilidade de portas de rede para a conexão, mas também a difusão de uma variedade de localizações físicas, o que dificultaria a conexão com a rede em tempo real, pois os dispositivos se movimentam. Portanto, fornecer conectividade através de um sistema com fio é praticamente um custo proibitivo (B'Far, 2005).

Uma palavra que define este novo paradigma é mobilidade. Em um ambiente onde usuários podem acessar serviços independentemente de onde estejam localizados, através de uma comunicação sem fio, que elimina a necessidade do usuário manter-se conectado a infraestrutura física.

De acordo com Google (2013), baseado nos dados públicos do Banco Mundial, foi constatado um aumento expressivo nos últimos anos da contratação de assinaturas de planos comerciais para utilização de dispositivos móveis, como se pode verificar na figura 1.

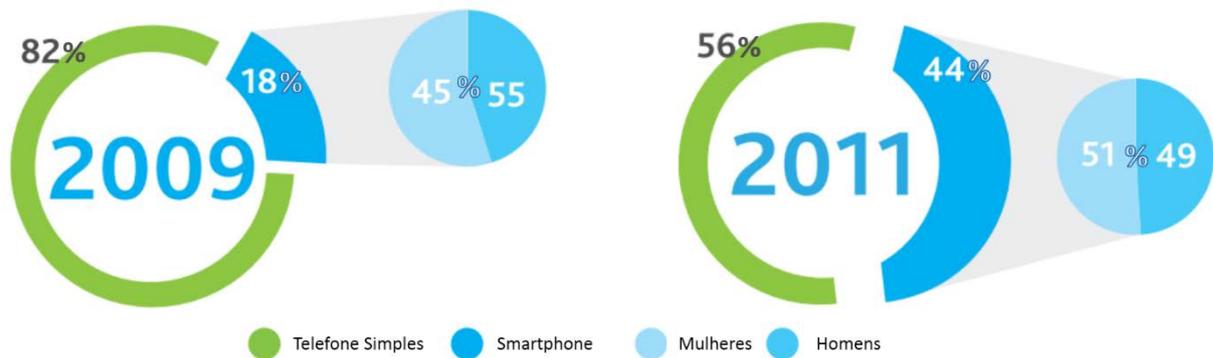
Figura 1 - Assinatura mundial de serviços móveis.



Fonte: Adaptado de Google (2013).

A figura 2 apresenta a pesquisa realizada por Nielsen (2011), exibindo um comparativo da situação do mercado de *smartphones* (telefone com funcionalidades avançadas de computação móvel) entre os anos de 2009 e 2011, onde se pode verificar o grande crescimento e popularização dos dispositivos móveis em um curto período.

Figura 2 - Crescimento do mercado de smartphones.



Fonte: Adaptado de Nielsen (2011).

A implantação e utilização de comunicação de dados sem fio faz com que seja necessário um sistema operacional para interligar os componentes que compõem uma aplicação móvel. O sistema operacional para computação móvel tem de lidar com a grande complexidade gerada pela alteração dinâmica de componentes dos aplicativos e também a reconfiguração dinâmica resultante das ligações entre esses componentes (Meier, 2002), o que será discutido no próximo capítulo.

## 2.2 Sistemas operacionais móveis

A necessidade de um sistema operacional (SO) especializado para suportar as tarefas executadas nos dispositivos móveis, oferece oportunidades na implementação de aplicações devido à proliferação de principalmente usuários de telefones celulares (*smartphones*) e *tablets*. Esses dispositivos se tornaram estruturas computacionais móveis e acessíveis, criando a necessidade de usuários utilizarem um ambiente de desenvolvimento que permitam a criação de aplicativos exclusivos e especializados.

O sistema operacional nada mais é que um *software* executado no dispositivo móvel, sua tarefa é gerenciar o *hardware* e os recursos do dispositivo. O sistema operacional também fornece uma interface chamada API (*Application Programming Interface*) permitindo a terceiros desenvolver aplicações nativas para serem executadas no dispositivo. A maioria dos sistemas operacionais móveis oferecem a possibilidade de instalar de forma dinâmica e fácil novas aplicações nativas em comparação com telefones convencionais, que só podem suportar aplicações pré-instaladas ou instaladas em modo seguro. Atualmente os fabricantes de dispositivos móveis estão lançando seus respectivos sistemas operacionais, por exemplo, o

*Symbian* pela *Nokia* ou *iPhone OS* da *Apple*, assim como os últimos lançamentos que entraram no mercado, como por exemplo, o *Android* da *Open Handset Alliance* (Palme et al., 2010).

A tabela 1, lista os principais sistemas operacionais utilizados atualmente em dispositivos móveis.

Tabela 1 - Sistemas operacionais móveis.

| <b>Sistemas Operacionais</b> | <b>Marcas</b>                              |
|------------------------------|--|
| <i>Android</i>               | <i>Google Inc. / Open Handset Alliance</i> |
| <i>iOS</i>                   | <i>Apple</i>                               |
| <i>Palm OS</i>               | <i>Palm, Inc</i>                           |
| <i>Windows Mobile</i>        | <i>Microsoft</i>                           |
| <i>Windows Phone</i>         | <i>Microsoft</i>                           |
| <i>Symbian OS</i>            | <i>Nokia</i>                               |
| <i>WebOS</i>                 | <i>Palm / HP</i>                           |
| <i>Meego</i>                 | <i>Intel / Nokia</i>                       |
| <i>Bada</i>                  | <i>Samsung</i>                             |
| <i>RIM</i>                   | <i>Blackberry</i>                          |
| <i>WinCE</i>                 | <i>Microsoft</i>                           |

Fonte: Primária (2013).

### 2.2.1 Sistema operacional *Android*

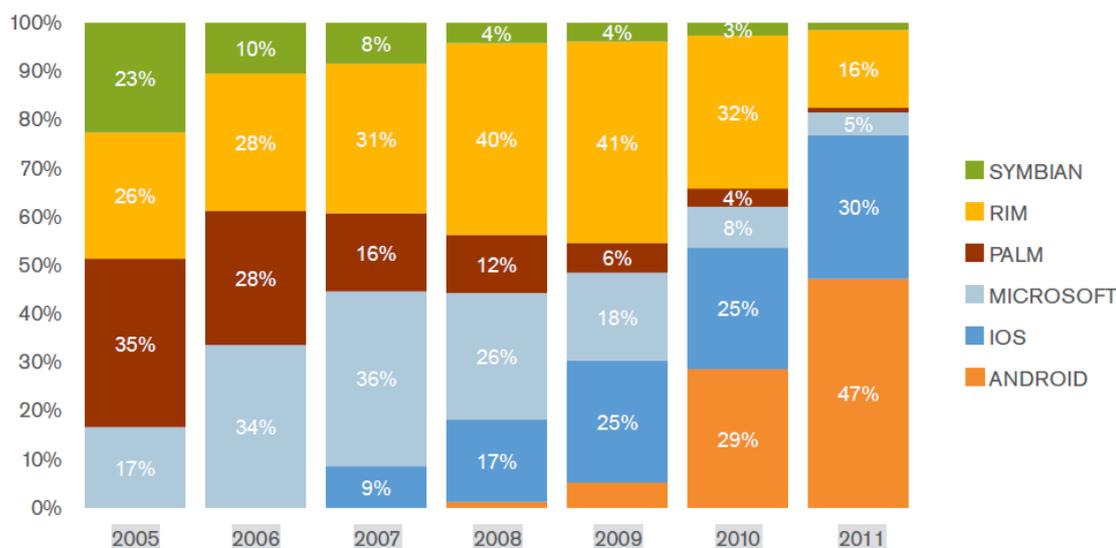
A principal característica do sistema operacional *Android* é de ser código aberto (*open source*), ou seja, é uma sistema gratuito, sendo assim qualquer fabricante de dispositivos móveis pode abertamente adotar o SO. Além disso, a *Open Handset Alliance* que suporta o sistema operacional fornece um kit de desenvolvimento de *software* livre (SDK) para desenvolvedores como uma maneira de incentivá-los a criar aplicativos para o *Android*. Enquanto em 2008, apenas um dispositivo estava rodando o SO, um ano depois, mais de cinco modelos de dispositivos móveis com o *Android* estavam disponíveis no mercado (Palme et al., 2010).

O *Android* é hoje uma das principais plataformas para dispositivos móveis. O fato de que é *open source* o distingue da maioria das outras plataformas móveis, como *iOS* e *Windows Phone*. A grande comunidade do *Android* usa este fato para portar a implementação original do para diferentes dispositivos, como por exemplo TVs, *E-Readers*, câmeras, ou conectá-lo a diferentes tipos de hardware periférico. Nós fazemos uso do fato de que o SO é construído em cima de um *kernel Linux* para estendê-lo aos recursos em tempo real, sem perder nenhuma funcionalidade específica do sistema operacional, preservando a compatibilidade com versões

anteriores. Um sistema *Android* com a capacidade de atender as solicitações em tempo real amplia o campo de aplicação para os dispositivos a domínios de tempo crítico. Por exemplo, o SO pode então ser utilizado como um dispositivo de monitoramento no campo, em instalações industriais, ou como uma plataforma de controle para automação de tarefas. As extensões de tempo real também poderiam melhorar as funcionalidades centrais do *Android* em si, assegurando a qualidade do processamento da fala ou de reprodução de vídeo (Kalkov et al., 2012).

De acordo com pesquisa realizada pela ComScore (2012), o *Android* lançado em 2008, vem crescendo rapidamente no mercado de sistemas operacionais. No período de 2005 a 2011, em 4 anos de existência conseguiu atingir quase metade do mercado de dispositivos móveis nos Estados Unidos da América (EUA) como mostra a figura 3.

Figura 3 - Quota de mercado por sistema operacional.



Fonte: Adaptado de ComScore (2012).

### 2.3 Banco de dados

Os bancos de dados e os sistemas gerenciadores de bancos de dados (SGBD) tornaram-se componentes essenciais na rotina da sociedade moderna. No cotidiano da sociedade atual, a grande maioria de usuários equipamentos computacionais se depara com atividades que envolvem alguma interação com os bancos de dados, como um depósito ou retirada de dinheiro em uma conta bancária, reservas em um hotel ou a compra de passagens aéreas, o acesso a uma biblioteca informatizada para consultar uma bibliografia, ou a compra de produtos de uma loja

por intermédio de sua página *Web*. Provavelmente estas tarefas envolverão a interação entre uma pessoa ou um programa de computador e um banco de dados.

Um sistema de banco de dados é constituído por um conjunto de dados associados a um conjunto de programas para acesso e controle desses dados. Um conjunto de dados, comumente chamado de banco de dados, contém informações sobre um domínio em particular. O principal objetivo de um SGBD é proporcionar um ambiente conveniente e eficiente para a recuperação, armazenamento e manipulação das informações do banco de dados (Silberschatz et al., 1999).

Um banco de dados é uma coleção de dados relacionados, na qual os dados significam fatos formalmente registrados. Um banco de dados típico representa aspectos do cotidiano do mundo real e pode ser utilizado por um ou vários grupos de usuários para propostas específicas. Um SGBD é um pacote de *software* para a implementação e manutenção de bancos de dados computadorizados. Juntos, o banco de dados e o *software* formam um sistema de banco de dados. Existem algumas funcionalidades que devem ser oferecidas por um SGBD para o DBA (*database administrator*), projetistas de bancos de dados e os usuários, para ajudá-los a projetar, administrar e utilizar um banco de dados (Elmasri; Navathe, 2005).

Os sistemas de banco de dados permitiram, a partir de um paradigma de processamento de dados, em que cada aplicação mantinha os seus próprios dados, para outro que os dados são definidos e administrados centralmente. Estes novos resultados causaram a independência de dados, no qual os programas de aplicação são imunes a variações na organização física ou lógica dos dados, eliminando sua redundância de dados.

Uma das principais motivações para a utilização de sistemas de banco de dados é o desejo de integrar os dados operacionais dentro de um domínio e para proporcionar a centralização, controlando assim o acesso a esses dados. A tecnologia de redes de computadores, por outro lado, promove um modo de trabalho que vai de encontro a todos os esforços de centralização. À primeira vista, pode ser difícil entender como estas duas abordagens contrastantes podem, eventualmente, ser sintetizadas para produzir uma tecnologia que é mais poderosa e mais promissora do que qualquer uma delas sozinha. A chave para esse entendimento é a compreensão de que o objetivo mais importante da tecnologia de banco de dados é a integração, e não a centralização. É importante perceber que qualquer um desses termos não implica necessariamente no outro. É possível alcançar a integração, sem centralização, e isso é exatamente o que a tecnologia de banco de dados distribuídos tenta alcançar (Ozsu; Valduriez, 2011).

A seguir serão abordados os conceitos fundamentais de um banco de dados distribuído, iniciando pela análise dos sistemas distribuídos em geral, a fim de esclarecer o papel da tecnologia dentro do processamento de dados distribuído.

### 2.3.1 Banco de dados distribuído

Diferentemente dos sistemas convencionais, onde estruturas de banco de dados são fortemente acopladas e visualizadas como um sistema de banco de dados único, um sistema distribuído consiste em uma estrutura fracamente acoplada que compartilha um recurso físico, onde cada sistema pode possuir um alto grau de independência.

Num sistema de banco de dados distribuído, a base de dados é armazenada em vários computadores. Os computadores em um sistema distribuído comunicam-se uns com os outros através de diversos meios de comunicação, como as redes privadas de alta velocidade ou a internet. Eles não compartilham memória principal ou discos. Os computadores em um sistema distribuído podem variar em tamanho e função, desde estações de trabalho até sistemas de *mainframe*. Os computadores de um sistema distribuído são representados por diversos nomes, tais como *sites* ou nós, dependendo do contexto em que são mencionados (Silberschatz et al., 2011). No presente trabalho será utilizado o termo nó, para enfatizar a distribuição física destes sistemas. A estrutura de um sistema de distribuído está representado na figura 4.

Figura 4 - Banco de dados distribuído.



Fonte: Primária (2013).

O projetista de um banco de dados distribuído (DDB - *distributed database*) vai decidir qual alternativa de distribuição é melhor para uma determinada situação. Eles podem decidir manter cada tabela intacta (todas as linhas e todas as colunas de cada tabela são armazenados no mesmo banco de dados e no mesmo nó), ou para dividir algumas das tabelas em pedaços menores, chamados fragmentos de dados ou partições. Em um banco de dados distribuído, os projetistas podem decidir armazenar esses fragmentos localmente, de forma centralizada, ou armazenar esses fragmentos em vários LDBMSs (*local database management system*) através dos nós distribuídos.

De acordo com a definição de Kumar (2006), em sistemas distribuídos, o banco de dados é distribuído em três formas: particionado, parcialmente replicado, e totalmente replicado para melhorar a disponibilidade de dados, qualidade de serviço e desempenho.

- **Particionado:** no âmbito desta forma todo o banco de dados é claramente dividido em um número de partições. Normalmente, o número de partições é equivalente ao número de nós (servidores). Essas partições de banco de dados são atribuídas aos servidores sob um conjunto de critérios, dentre os mais importantes estão: a maioria das consultas devem ser processadas pela partição no servidor local; deve ajudar a minimizar o custo de comunicação de dados; deve ajudar a minimizar o custo de manutenção da coerência dos dados; deve ajudar a localizar a serialização de transações simultâneas; e deve minimizar o custo da recuperação.
- **Replicação parcial:** sob esta forma, o banco de dados é dividido e um subconjunto de partições é replicado em mais de um servidor. A replicação parcial tem todas as propriedades do esquema de partição e melhora ainda mais a localização de dados e confiabilidade. Se um servidor falhar, então a cópia replicadas da partição ainda estará disponível para processar as consultas. A recuperação é mais fácil porque a partição replicada pode ser copiada integralmente para o servidor depois que ele se recupera da falha. É relativamente mais demorado para manter a consistência global porque qualquer alteração numa partição deve ser instalada para todas as suas replicações localizadas em vários servidores. Para aumentar ainda mais a disponibilidade local, o banco de dados é totalmente replicado.
- **Totalmente replicado:** nesta forma todo o banco de dados é replicado em todos os servidores. Assim, é possível fornecer disponibilidade local máxima e também minimizando o custo da comunicação de dados durante o processamento da consulta.

O esquema totalmente replicado oferece a mais alta confiabilidade e disponibilidade, mas com o custo de manter a consistência global.

Com base nas definições acima, a tabela 2 resume os méritos das três abordagens de distribuição de dados.

Tabela 2 - Comparativo entre as formas de banco de dados distribuídos.

| <b>Atividade</b>      | <b>Totalmente replicado</b> | <b>Replicação parcial</b> | <b>Particionado</b> |
|-----------------------|-----------------------------|---------------------------|---------------------|
| Disponibilidade Local | Muito alta                  | Média                     | Baixa               |
| Serialização          | Difícil                     | Moderada                  | Fácil               |
| Consistência          | Difícil                     | Moderada                  | Fácil               |
| Confiabilidade        | Muito alta                  | Moderada                  | Baixa               |
| Custo de recuperação  | Muito alta                  | Moderada                  | Baixa               |

Fonte: Adaptado de Kumar (2006).

## 2.4 Sincronização de dados

Sob o paradigma da computação móvel, os usuários têm acesso aos seus dados onde quer que estejam, muitas vezes fracamente ou intermitentemente ligados a uma rede. Sistemas que fornecem esse acesso a dados de forma onipresente devem lidar com o problema inerente de sincronização de dados, isto é, as alterações feitas remotamente devem ser refletidas de volta no banco de dados do servidor e vice-versa. A tarefa da sincronização de dados é identificar rapidamente essas mudanças, resolver possíveis conflitos e propagar as atualizações para os vários dispositivos de sincronização (Agarwal et al., 2002).

Segundo Rahimi; Haug (2010), os algoritmos de controle de sincronização devem manter a coerência entre as cópias (nós) do banco de dados. Uma maneira de categorizar as abordagens se baseia nas cópias, se são ou são idênticas em todos os momentos. A partir desta perspectiva, há duas abordagens para o controle da sincronização: o controle da replicação síncrona e controle da replicação assíncrona.

Na replicação síncrona, as cópias são mantidas em sincronia em todos os momentos. Nesta abordagem, a transação pode acessar qualquer cópia de um registro no banco de dados com a garantia de que, o registro que está acessando tem o mesmo valor que todas as suas outras cópias. Obviamente, é fisicamente impossível alterar os valores de todas as cópias de registro, exatamente ao mesmo tempo. Portanto, para fornecer uma visão consistente em todas as cópias, enquanto uma cópia do registro está sendo alterado, o algoritmo de controle de replicação tem de esconder o registro das outras cópias que estão fora de sincronia com ele. Em outras palavras,

nenhuma transação será sempre capaz de ver valores diferentes para diferentes cópias do mesmo registro. Na replicação assíncrona, em oposição a replicação síncrona, as réplicas não são mantidos em sincronia em todos os momentos. Duas ou mais réplicas do mesmo registro podem as vezes ter diferentes valores, e qualquer transação pode ver esses valores diferentes. Esta característica passa a ser aceitável em algumas aplicações, como o estoque e o ponto-de-venda de uma empresa (Rahimi; Haug, 2010).

De acordo com B´Far (2005), existe uma diferença entre replicação de dados e sincronização de dados, onde a replicação de dados, em seu sentido mais amplo, refere-se simplesmente a cópia de um banco de dados de um ou mais locais de armazenamento, para um ou mais locais de armazenamento. Já a sincronização de dados, garante que não há conflitos ou discrepâncias entre duas ou mais instâncias dos mesmos dados. A sincronização de dados, quando utilizada em aplicações móveis, permite que o dispositivo do usuário mantenha uma cópia local consistente de seus dados, a partir de uma central de armazenamento de dados ou um provedor de serviços.

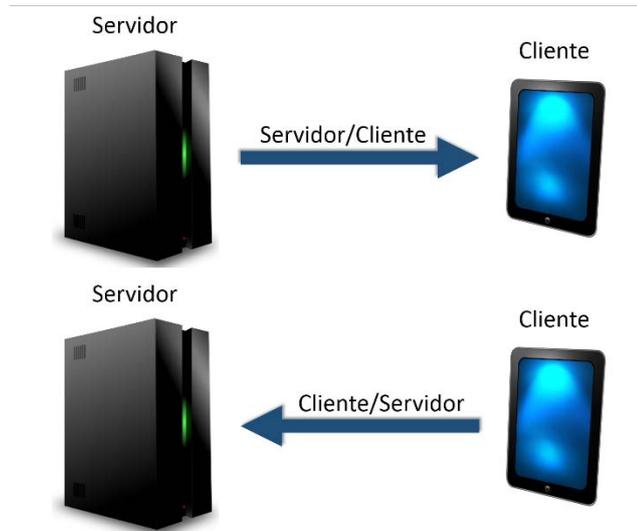
Uma das características da sincronização é que o dispositivo transmissor dos dados da sincronização não recebe nenhuma mensagem de aceite ou erro das modificações enviadas ao dispositivo receptor, visto que se ocorrer algum erro em um processo de sincronização, ao executar novamente um novo processo, serão enviados os dados que não foram sincronizados nos processos anteriores.

#### 2.4.1 Tipos de Sincronização

Os tipos de sincronização podem ser divididas em *one-way*, onde a transferência de dados é transmitida em um único sentido, e *two-way*, onde as modificações ocorrem nos dois sentidos (Palazzo et al., 2000).

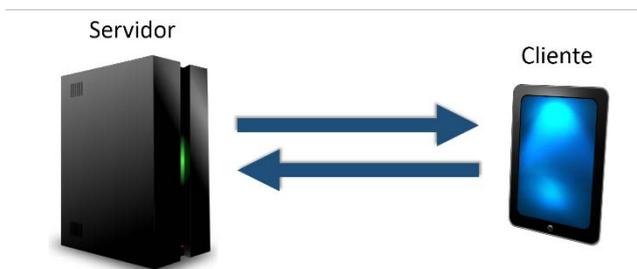
Como ilustrado na figura 5 a sincronização *one-way* pode ser efetuada em duas formas:

- **Cliente/Servidor:** nessa forma o dispositivo cliente é responsável por enviar as modificações nas tabelas do BD ao dispositivo servidor, que irá atualizar seus registros conforme o enviado pelo transmissor. Caso o registro possua a mesma chave primária do servidor, a sincronização irá subscrever o registro do servidor.
- **Servidor/Cliente:** o dispositivo servidor envia todos os registros das tabelas atualizadas no BD consolidado ao dispositivo cliente. Ao final da sincronização o dispositivo cliente possui a mesma versão do BD do servidor. As alterações executadas no dispositivo cliente não serão capturadas pelo servidor.

Figura 5 - Sincronização *one-way*.

Fonte: Primária (2013).

A sincronização *two-way* ocorre nos dois sentidos (Palazzo et al., 2000), as modificações são efetuadas tanto no dispositivo cliente quanto no servidor. O processo de sincronização inicia-se pelo cliente, enviando ao servidor os registros, o servidor por sua vez identifica e trata os conflitos existentes retornando ao dispositivos cliente os dados consolidados. Ao final do processo os dois dispositivos possuem a mesma versão do BD, sem perder nenhum registro. A figura 6 ilustra o processo de sincronização *two-way*.

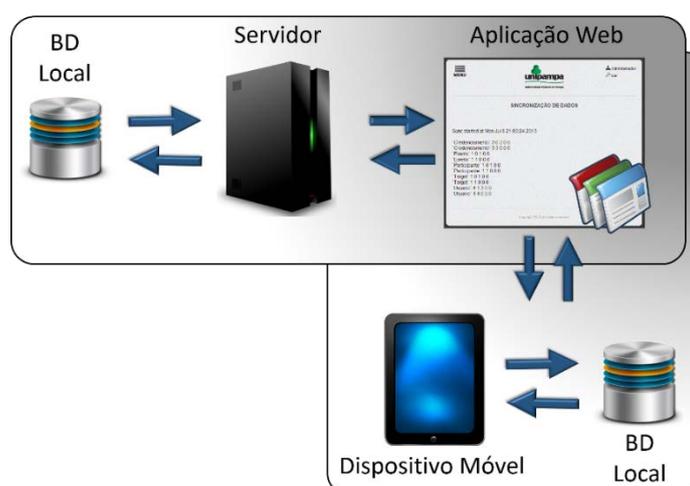
Figura 6 - Sincronização *two-way*.

Fonte: Primária (2013).

### 3 PROPOSTA

Neste capítulo é feita a descrição do ambiente e das técnicas de sincronização de dados proposta neste trabalho. Inicialmente, são apresentadas as ferramentas utilizadas no ambiente proposto e suas principais características. Em seguida é efetuada a descrição dos conceitos de projeto de *software*. Então é feita uma abordagem do problema proposto e a definição do escopo da aplicação. Por fim são demonstrados os testes realizados a fim de atender os objetivos do trabalho realizado.

Figura 7 - Visão geral da aplicação.



Fonte: Primária (2013).

Como visto na figura 7, a aplicação foi dividida em três conceitos:

- **Servidor:** equipamento que servirá como ponto central para concentrar todos os dados que serão criados a partir dos dispositivos móveis.
- **Dispositivos móveis:** serão utilizados para efetuar consultas e inserir dados no BD local dos dispositivos, que posteriormente serão sincronizados com o servidor.
- **Aplicação web:** aplicação de software desenvolvida para executar a sincronização dos dados entre os dispositivos móveis e o servidor.

O fluxo de dados é criado principalmente pelo usuário do dispositivo móvel, o qual irá registrar as informações no dispositivo, e posteriormente serão sincronizadas. A sincronização permite que os dados sejam gravados no servidor através da funcionalidade de sincronização disponível na aplicação *web*. Após a sincronização os BDs possuem as mesmas informações, podendo ser utilizado para gerenciar, acessar e classificar os dados pela aplicação *web*.

Por problemas de disponibilidade de rede, a sincronização não pode ser feita em tempo real. Desta forma, a abordagem suporta que a sincronização ocorra em modo *batch*. Outro

grande problema que as funcionalidades nativas do BD não suportam é o fato de que a sincronização deve ser nos dois sentidos (*multimaster*), onde todas as alterações de registros executadas no servidor sejam atualizadas no dispositivo móvel, e todas as alterações realizadas no dispositivo móvel sejam atualizadas no servidor. Destaca-se uma observação importante da terminologia usada neste trabalho, que os termos *two-way* e *multimaster* são tratados como sinônimos. A razão disto vem do fato de que na literatura de banco de dados distribuídos, a sincronização de dados entre vários dispositivos considerados como *master*, caracteriza-se como *multimaster*.

Grande parte dos SGBDs possuem a funcionalidade de replicação de dados *master-slave*, ou seja, possui um servidor atuando como principal (*master*), e os outros dispositivos como secundários (*slave*), onde todas as alterações ocorridas no servidor principal são replicadas para todos os dispositivos secundários em um único sentido (*one-way*). A implicação desta fato restringe alterações no dispositivo secundário, as quais não ocorrem no dispositivo principal.

Uma das grandes contribuições desta pesquisa avança sobre a literatura de referência demonstrando o grau de inovação obtido no tocante a permissibilidade de efetuar a sincronização *multimaster* com suporte do banco de dados MySQL, pois a própria literatura destaca em um quadro à página 364 (Schwartz et al., 2008) a restrição do uso desta tecnologia nos resultados obtidos nos testes de sincronização do presente trabalho.

### 3.1 Tecnologias utilizadas

No desenvolvimento da solução proposta foi empregado o modelo de ciclo de vida evolucionário, permitindo que o usuário visualize os resultados parciais rapidamente, efetuando comentários sobre as falhas a serem corrigidas e os requisitos que devem ser atendidos nas próximas versões da aplicação. O modelo foi escolhido por ser o mais apropriado, pois as funcionalidades necessitavam de velocidade na implementação das soluções pelo motivo do projeto apresentar um curto prazo para finalização.

A partir das necessidades do domínio escolhido e dos requisitos de software elicitados, foi iniciada a modelagem da aplicação, construindo o diagrama sequência para descrever com propriedade as características da sincronização e o diagrama entidade relacionamento (DER) utilizado para representar o modelo conceitual do BD. O ambiente de implementação e testes é composto por dois dispositivos móveis, e um servidor com suas principais configurações descritas na tabela 3 e na tabela 4.

Tabela 3 - Configuração do servidor.

| <b>Hardware</b>            | <b>Especificação</b>               |
|----------------------------|------------------------------------|
| <i>Modelo</i>              | <i>Lenovo EDGE 72 3484</i>         |
| <i>CPU</i>                 | <i>Intel® Core™ I3 3220 3.3Ghz</i> |
| <i>Memória RAM</i>         | <i>4 Gb</i>                        |
| <i>Hard Disk</i>           | <i>500 Gb</i>                      |
| <i>Rede</i>                | <i>LAN 10/100/1000</i>             |
| <i>Sistema Operacional</i> | <i>Debian 6.0.7 (squeeze)</i>      |

Fonte: Primária (2013).

Tabela 4 - Configuração dos dispositivos móveis.

| <b>Hardware</b>            | <b>Especificação</b>                  |
|----------------------------|---------------------------------------|
| <i>Modelo</i>              | <i>Samsung Galaxy Tab P3100</i>       |
| <i>CPU</i>                 | <i>ARMv7 Processor Dual-Core 1Ghz</i> |
| <i>Memória RAM</i>         | <i>1 Gb</i>                           |
| <i>Hard Disk</i>           | <i>16 Gb + 8 Gb microSD</i>           |
| <i>Rede</i>                | <i>WLAN Wi-Fi 802.11 a/b/g/n</i>      |
| <i>Sistema Operacional</i> | <i>Android™ 4.0.4</i>                 |

Fonte: Primária (2013).

No ambiente experimental foram utilizados os dispositivos móveis com conexão sem fio e um servidor fixo conectado fisicamente à rede. A seguir serão descritas todas as ferramentas de software utilizadas na implementação da proposta de sincronização *multimaster*, no projeto da aplicação e nos testes executados.

### 3.2 Ferramentas de implementação

A aplicação foi desenvolvida utilizando somente ferramentas *open source*, permitindo a troca aberta de informações, contribuindo para que a aplicação possa tornar-se uma ferramenta de software verdadeiramente poderosa e eficiente, disponível para todos. Isso permite que o código fonte seja disponível publicamente, podendo ter seus conceitos aprimorados.

Para a modelagem do sistema foi utilizada a ferramenta *ArgoUML*, versão 0.34, e na modelagem do BD a ferramenta *MySQL Workbench*, versão 5.2.45 CE.

No que diz respeito ao ambiente de desenvolvimento, é importante entender o papel que cada um dos programas (*Apache*, *MySQL* e *PHP*) desempenha na criação de uma aplicação *web*. Na implementação da funcionalidade de sincronização foi utilizada a ferramenta *SQLyog Job Agent* (SJA), que mostrou-se uma ferramenta versátil e que atendeu os requisitos funcionais

das técnicas de sincronização. Todas as ferramentas utilizadas podem ser obtidas gratuitamente nos *websites* de seus desenvolvedores.

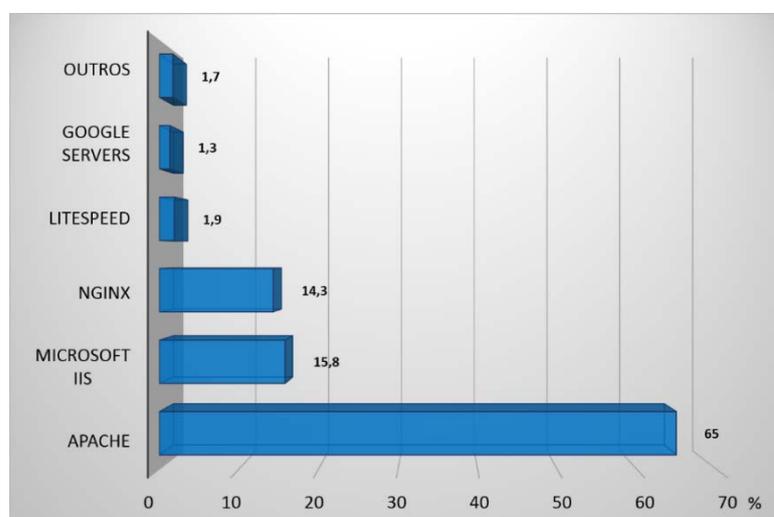
### 3.2.1 Apache

O Projeto Apache *HTTP Server* é um aplicação de desenvolvimento de software colaborativo que visa a criação de um servidor *HTTP (web)* robusto, comercial, com mais recursos, e de código fonte livre (*open source*). O projeto é gerido em conjunto por um grupo de voluntários distribuídos em todo o mundo, que utilizam a internet e a *web* para se comunicar, planejar e desenvolver o servidor e sua respectiva documentação. Este projeto faz parte da *Apache Software Foundation*. Além disso, centenas de usuários têm contribuído ideias, códigos e documentação para o projeto (Apache, 2012).

O Apache atua como um servidor web. Sua principal função é analisar qualquer arquivo solicitado por um navegador e exibir os resultados corretos de acordo com o código dentro desse arquivo. O servidor Apache é bastante poderoso e pode fazer virtualmente qualquer tarefa que um *Webmaster* exige (Naramore et al., 2005).

De acordo com W3techs (2013a), no mês de julho de 2013 o servidor Apache está em execução em mais de 34 milhões de servidores, correspondente a 65% de todos os servidores de internet em uso atualmente, conforme ilustra a figura 8. Segundo Naramore et al. (2005), sua flexibilidade, poder e baixo custo operacional fazem dele uma escolha popular, podendo ser usado para hospedar um site para o público em geral, de uma grande empresa de intranet, ou simplesmente para testar suas páginas antes de serem enviadas para um servidor seguro em outra máquina.

Figura 8 - Uso dos serviços de servidores web.



Fonte: Adaptado de W3techs (2013a).

### 3.2.2 MySQL

Para adicionar, acessar, e processar dados armazenados em um banco de dados, é necessário um sistema de gerenciamento de banco de dados (SGBD) como o servidor MySQL. Como os computadores são muito bons em lidar com grandes quantidades de dados, o SGBD desempenha um papel central na computação, seja como utilitários independentes ou como partes de outras aplicações.

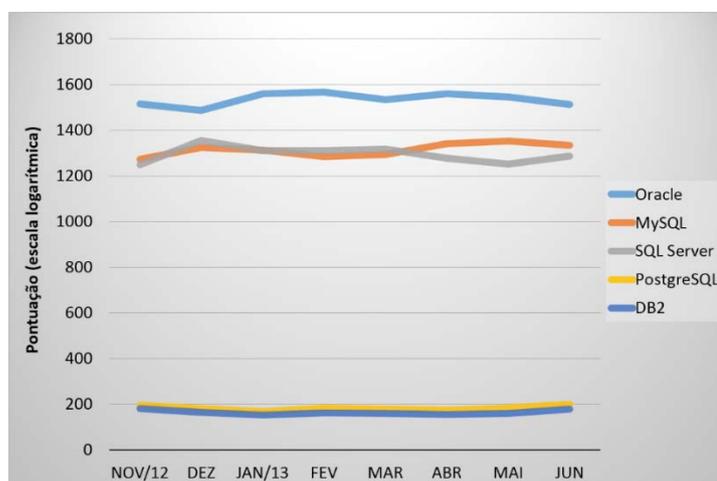
O MySQL é um SGBD que permite ao PHP e Apache trabalharem em conjunto para acessar e exibir dados em um formato legível para um navegador de internet. Segundo Naramore et al. (2005), o MySQL é um servidor SQL (*Structured Query Language*) projetado para cargas pesadas e de processamento de consultas complexas. Como um sistema de banco de dados relacional, o MySQL permite que muitas tabelas diferentes possam ser agrupadas para a máxima eficiência e rapidez.

Dentre as principais características do MySQL podemos destacar (MySQL, 2013):

- Facilidade de uso;
- portabilidade (Suporta praticamente qualquer sistema operacional da atualidade);
- excelente desempenho e estabilidade;
- aplicativos complementares, como o *MySQL Workbench*;
- é um *software* livre;
- exige poucos recursos de hardware.

Atualmente o MySQL está entre os SGBDs mais populares de acordo com pesquisa realizada pela DB-Engines (2013), que classifica os SGBDs pela utilização de alguns parâmetros para calcular sua popularidade. A figura 9 ilustra os cinco primeiros da pesquisa.

Figura 9 - SGBDs mais populares.



Fonte: Adaptado de DB-Engines (2013).

### 3.2.3 PHP

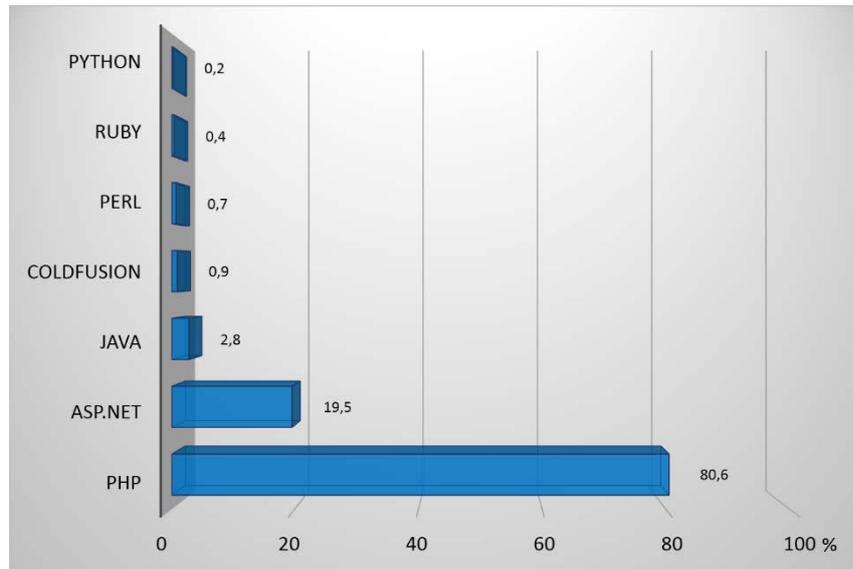
O PHP é uma linguagem de script do lado do servidor (*server-side*) que permite que seu site seja verdadeiramente dinâmico. Sua flexibilidade e curva de aprendizagem relativamente pequena (especialmente para os programadores que têm experiência em C, *Java* ou *Perl*) torna-a uma das linguagens de programação mais populares. A popularidade do PHP continua a aumentar à medida que as empresas e os usuários em todos os lugares o utilizaram como uma alternativa à linguagem ASP da *Microsoft* e perceberam que os benefícios do PHP certamente superam os custos de uma linguagem licenciada (Naramore et al., 2005).

O PHP foi projetado para ser uma linguagem de script *server-side*, portanto, é possível fazer qualquer função que outro programa CGI (*Common gateway interface*) possa fazer, como: coletar dados de formulários, gerar páginas com conteúdo dinâmico ou enviar e receber cookies. Como as maiores áreas de atuação do PHP pode se destacar as seguintes (PHP, 2013):

- Script no lado do servidor (*server-side*): este é o mais tradicional e principal campo de atuação do PHP. São necessárias três ferramentas para executar essa característica, são elas: o interpretador do PHP (como CGI ou módulo), um servidor web e um navegador de internet. Basta rodar o servidor *web* conectado a um PHP instalado, para poder acessar os resultados do programa PHP com o navegador, visualizando a página PHP através do servidor *web*. Todos os serviços podem rodar em um único equipamento.
- Script de linha de comando: possibilita a criação de um script PHP para funcionar sem um servidor *web* ou um navegador. A única ferramenta necessária é o interpretador. Esse tipo de uso é ideal para script executados usando o *cron* ou o Agendador de Tarefas (no *Windows*). Esses scripts podem ser usados também para rotinas de processamento de texto
- Criação de aplicações desktop: O PHP provavelmente não é a melhor linguagem para criação de aplicações desktop com interfaces gráficas, mas pode usar alguns dos seus recursos avançados nas aplicações do lado do cliente, tendo como principal atrativo a possibilidade de escrever aplicações multi-plataformas.

De acordo com W3techs (2013b), o código PHP pode atualmente ser encontrado em aproximadamente 16 milhões de servidores *web*, correspondente a mais de 80%, como ilustra a figura 10.

Figura 10 - Uso de linguagens de programação web *server-side*.



Fonte: Adaptado de W3techs (2013b).

A linguagem PHP é uma linguagem de programação de domínio específico, ou seja, seu escopo se estende a um campo de atuação que é o desenvolvimento web, embora tenha variantes como o já mencionado PHP-GTK. Seu propósito principal é de implementar soluções web velozes, simples, eficientes e portáteis, independentemente da plataforma utilizada.

### 3.2.4 SJA

O *SQLyog Job Agent* (SJA) é uma aplicação de alta performance, *multithreaded* e multi-plataforma que permite executar scripts de manutenção com bancos de dados MySQL, possui logs de resultados da sincronização, importação de dados e metadados de fontes ODBC, sincroniza BDs e tabelas do MySQL e faz backups agendados de alto desempenho. É disponibilizado para plataformas *Windows* e *Linux*. No *Windows*, ele está incluído no *SQLyog Enterprise* e *SQLyog Ultimate*. Para o *Linux* é gratuito para uso pessoal e comercial (Laursen, 2009).

O SJA é uma ferramenta executada por linha de comando para o MySQL, pode ser executado em *Windows* e *Linux*, mas irá se conectar a qualquer servidor MySQL rodando em qualquer sistema operacional. Esse é o significado do termo cliente. Uma das características mais importantes do programa SJA é a de sincronizar os dados (bancos de dados completos ou tabelas individuais) em dois servidores. Ele utiliza algoritmos de *checksum* avançados para comparação dos dados e pode ser muito eficaz em "descobrir" que os dados devem ser inseridos,

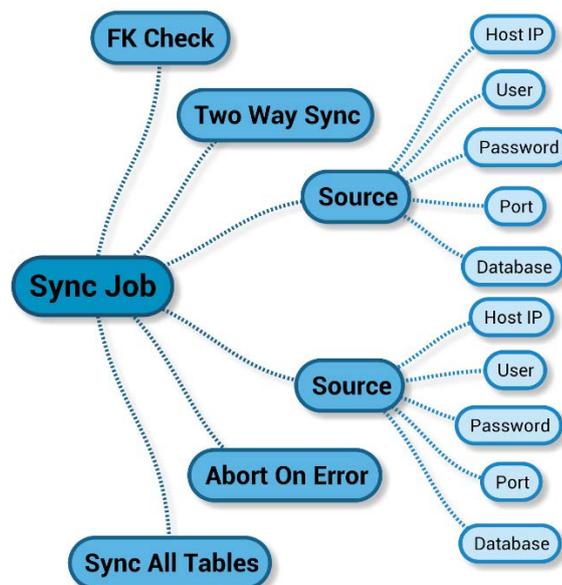
atualizados ou excluídos (*insert*, *update* e *delete*) em qualquer um dos servidores para executar a sincronia. (Laursen; Ton, 2011).

Existem diferentes opções disponíveis com a ferramenta de sincronização, as mais importantes são as de sincronização disponíveis, divididas em *one way* ou *two way*, que são tipo de sincronização como já mencionado nesse trabalho no capítulo 2.4.1. Outra opção importante é de escolher para que os dados não sejam excluídos pelo SJA. Pode-se optar também por deixar o programa abortar em caso de algum erro ou seguir em frente com a próxima linha de dados. Existem também algumas opções de otimização (Laursen; Ton, 2011).

A ferramenta executado por linha de comando lê um arquivo de definição de tarefa (*job definition*) codificado em XML como parâmetros para a execução (Laursen, 2009). Esses parâmetros são lidos pelo SJA do arquivo XML, que pode ser criado a partir de qualquer aplicação ou até mesmo um simples editor de textos. O SJA também pode ser iniciado por qualquer aplicação capaz de executar programas externos e tais programas podem até mesmo gerar ou modificar o arquivo XML. Essa é uma das ideias básicas de utilização do formato XML (Laursen; Ton, 2011).

A figura 11 ilustra a estrutura do arquivo XML utilizada na sincronização dos dados no presente trabalho.

Figura 11 - Estrutura do arquivo XML.



Fonte: Primária (2013).

Abaixo serão descritas algumas definições muito importantes para compreender o funcionamento do SJA de acordo com Laursen; Ton (2011):

- A funcionalidade de sincronização do SJA exige que sempre existam dois *hosts* definidos, um como *source* e o outro como *target*. O *source* geralmente é definido para o servidor de banco de dados, e o *target* o equipamento que deverá ser o objetivo da sincronização.
- É necessário identificar exclusivamente cada registro de dados, de modo que os registros correspondentes em ambos os *hosts* possam ser identificados.
- Geralmente, recomenda-se usar uma chave primária (*PK*) das tabelas para os registros serem sincronizados por esta identificação única. Quando um *PK* existe, o SJA simplesmente identifica cada registro através da *PK*. Dois registros em cada *host* com o mesmo *PK* serão considerados como idênticos.
- No caso de conflito de chaves primárias (que é o mesmo valor da *PK* usada por registros diferentes em cada *host*), os dados o *source* irão sobrescrever os dados no *target*. Portanto, a importância da criação da *PK* com cuidado não pode ser enfatizado o suficiente se você quiser usar o SJA com seus dados.
- Não são utilizados *timestamps* ou qualquer tipo de variável de data para a sincronização, a menos que a *PK* seja construída com tal variável tempo. No entanto, a precisão de um *timestamp* com MySQL é de apenas um segundo. E dependendo do tamanho da quantidade de utilização do banco de dados, um segundo pode ser um tempo muito longo. Assim, um atributo do tipo *timestamp* geralmente não será utilizado sozinho como uma chave primária.

### 3.2.5 AndroPHP

Quando for necessário criar aplicação *web*, é necessário possuir um ambiente de desenvolvimento que possua suporte para as tecnologias nas quais serão implementados os projetos. Como já apresentado anteriormente as tecnologias *web* mais utilizadas nesse tipo de ambiente são compostas de um servidor *web* com PHP (baseado no Apache ou *Lighttpd*) e um servidor MySQL.

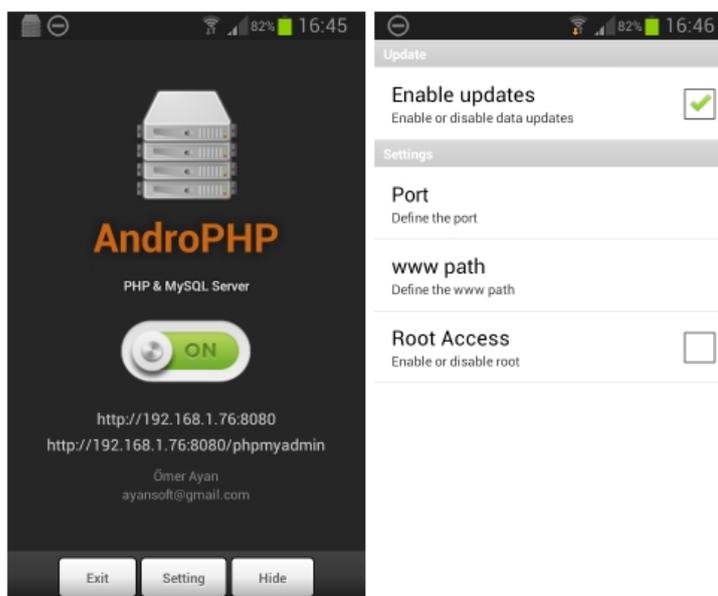
Como este ambiente de desenvolvimento não requer grande capacidade de processamento e de armazenamento, as características de *hardware* dos dispositivos móveis são suficientes para atender esta demanda. Então, para implementar a solução, foi utilizado o ambiente AndroPHP, um servidor *web* desenvolvido para sistemas operacionais *Android*,

semelhante aos já disponíveis em outros sistemas operacionais. O AndroPHP é uma aplicação que reúne os seguintes componentes:

- Apache HTTPD 1.4.29;
- PHP 5.4.8;
- MySQL 5.1.62;
- phpMyAdmin 3.5.3;
- FTP.

A figura 12 apresenta a tela inicial e a de configuração do servidor *web*, onde é possível configurar algumas opções como a porta de conexão ao servidor e o diretório onde são armazenados os arquivos visíveis ao acessar o endereço *localhost*.

Figura 12 - Tela inicial do AndroPHP.



Fonte: Primária (2013).

### 3.3 Projeto da aplicação

A estrutura do projeto da aplicação foi dividida da seguinte forma:

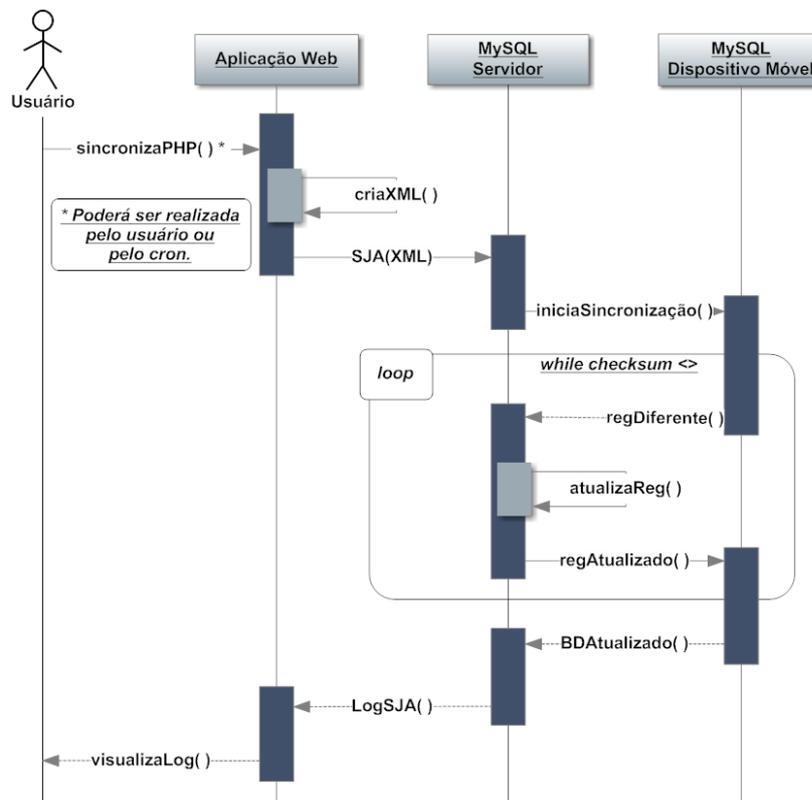
- Páginas *web*: diretório que contém os arquivos PHP, no mesmo diretório encontram-se todos os recursos utilizados separados em diretórios específicos, para que fiquem visualmente melhores e mais dinâmicos, visando uma facilidade na manutenção do código fonte.
- Imagens: contém os arquivos gráficos da aplicação, tais como: botões de navegação, ícones, logotipos e plano de fundo da aplicação.

- Estilo: diretório que armazena todos os estilos de formatação (CSS) das páginas *web*.
- SJA: contém o arquivo executável da aplicação utiliza na sincronização, o arquivo de parâmetros no formato XML e o arquivo de *log*.

A estrutura mencionada é encontrada em sua totalidade no servidor. Os dispositivos móveis não possuem o módulo do SJA, responsável pela sincronização, que será executada somente através do servidor.

A figura 13 apresenta o diagrama de sequência gerado com base no processo de sincronização efetuado com a interação da aplicação *web* e da ferramenta SJA.

Figura 13 - Diagrama de sequência da sincronização.



Fonte: Primária (2013).

### 3.3.1 Estudo de caso

A técnica de sincronização proposta neste trabalho foi aplicada sobre uma questão real no controle de credenciamento de participantes em eventos acadêmicos, onde se mostra sua importância no processo de distribuição de uma aplicação que utiliza BD local em dispositivos móveis. Nas próximas seções neste capítulo serão descritos os processos modelados baseados nos requisitos, com um formalismo suportado pelas ferramentas de desenvolvimento e projeto de *software*.

### 3.3.2 Problema

Um evento acadêmico envolve vários processos referentes à inscrições *online*, submissão de trabalhos, programação de palestras, controle financeiro, emissão e controle de certificados, notas e avaliações de trabalhos apresentados, entre outros. Para a criação da proposta apresentada, fez-se necessária a escolha de um subdomínio dentro da área escolhida, que fosse responsável pela maior necessidade de uma aplicação móvel.

Para a implementação da aplicação móvel foi escolhido o controle de credenciamento de participantes, que envolve os processos de criação e controle de eventos, participantes, usuários da aplicação responsáveis pelo credenciamento e por fim o controle dos dispositivos móveis utilizados, para controlar a sincronização dos dados.

### 3.3.3 Definição das fronteiras da aplicação

A observação informal em algumas instituições de ensino na região, autoriza assegurar que a maioria das instituições não possuem uma aplicação de controle de participantes credenciados, que tenha uma flexibilidade para que possa ser utilizada nos mais diversos ambientes, independente de conexões de rede ou dependência de conexão com um servidor central. Por isso optou-se pelo uso deste domínio como exemplo no uso de uma solução móvel, para expor a contribuição deste trabalho de um bem sistematizado processo de sincronização de dados.

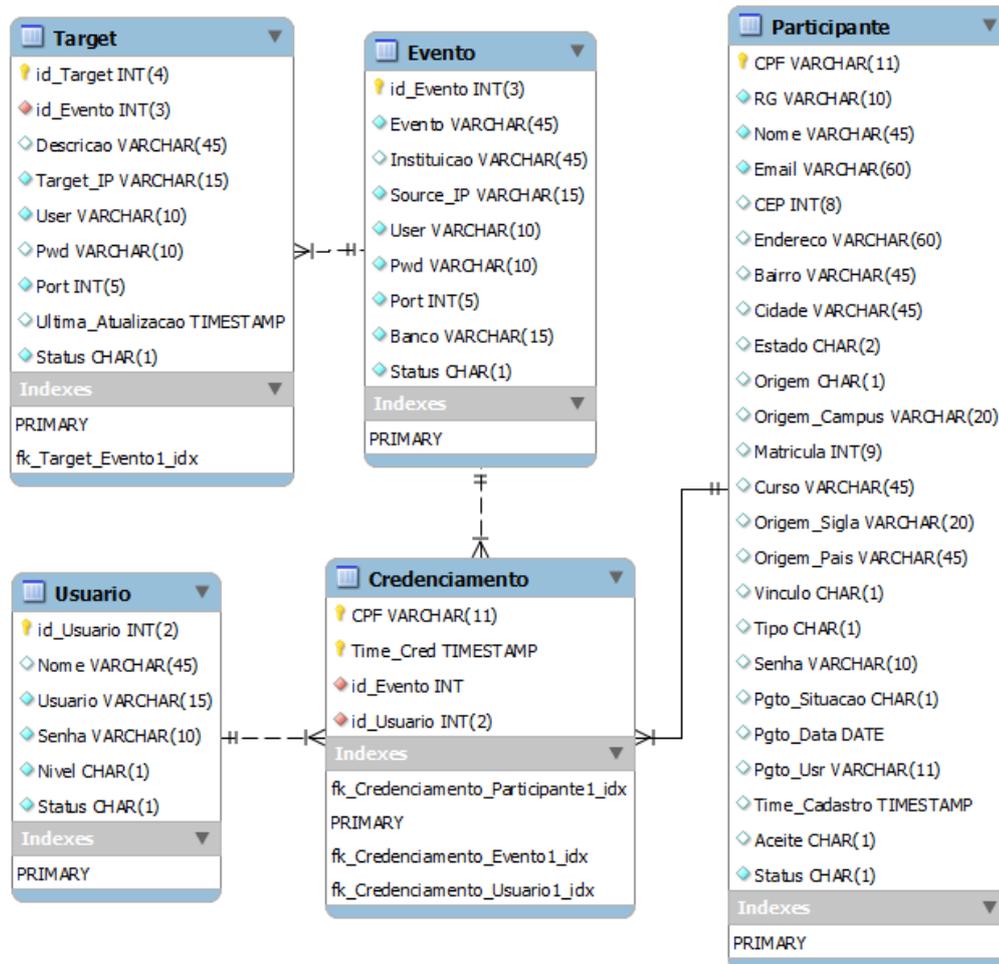
O grande nível de tempo perdido com retrabalho decorrente do uso de controles manuais, implica na atualização dos dados em uma etapa posterior em uma estação de trabalho conectada ao servidor de banco de dados. Este fato contribui para a demora na emissão de certificados de participação.

Para o bom funcionamento e entendimento dos processos abordados, a aplicação foi dividida em quatro grupos de operações:

- Cadastro: inserção de usuários, participantes, eventos e dispositivos.
- Edição: função para editar informações dos usuários, participantes, eventos e dispositivos.
- Consulta: funcionalidade para consultar os registros inseridos na aplicação. Na opção de consulta de dispositivos possui um ícone que realiza a sincronização do disposto com o servidor de banco de dados do evento.
- Credenciamento: efetua o credenciamento do participante em determinado evento, mediante a informação do CPF do participante.

A modelagem do banco de dados foi realizada utilizando a ferramenta *MySQL Workbench*, como já mencionado. A figura 14 ilustra o modelo entidade-relacionamento do banco de dados utilizado na aplicação desenvolvida.

Figura 14 - Modelo Entidade-Relacionamento.

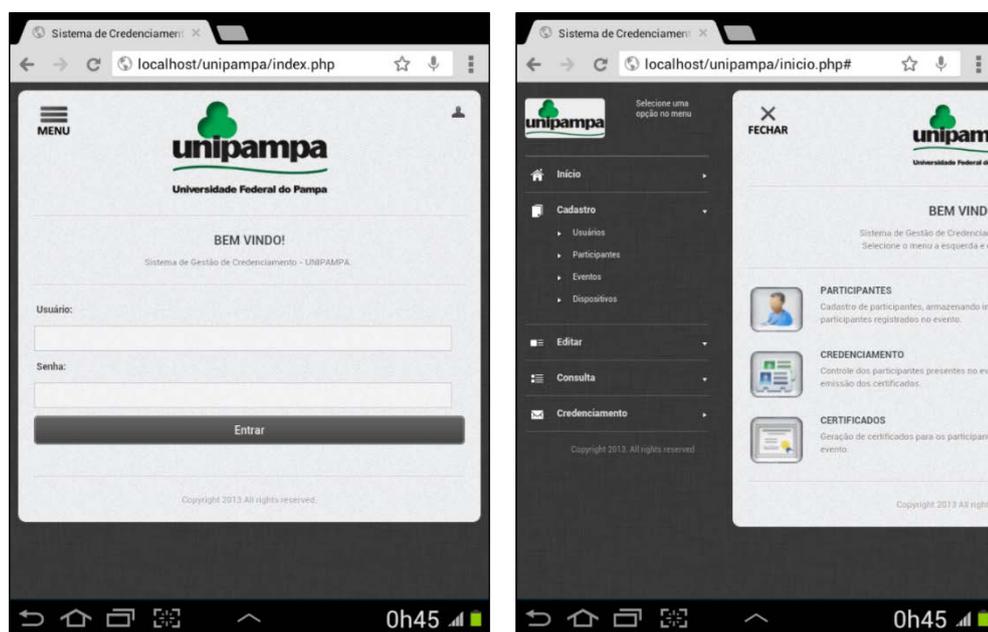


Fonte: Primária (2013).

### 3.4 Interfaces da aplicação

As interfaces da aplicação de controle de credenciamento ilustradas nessa seção foram capturadas através do dispositivo móvel. A figura 15 exibe as interfaces iniciais da aplicação, que necessita que o usuário efetue a autenticação com usuário e senha para que consiga acessar a aplicação. O controle de permissões feito pelo aplicativo utilizou o recurso de sessões do PHP, que permite que usuário interaja com as páginas e também que os credenciamentos efetuados pelo usuário sejam identificados pelo código de seu cadastro.

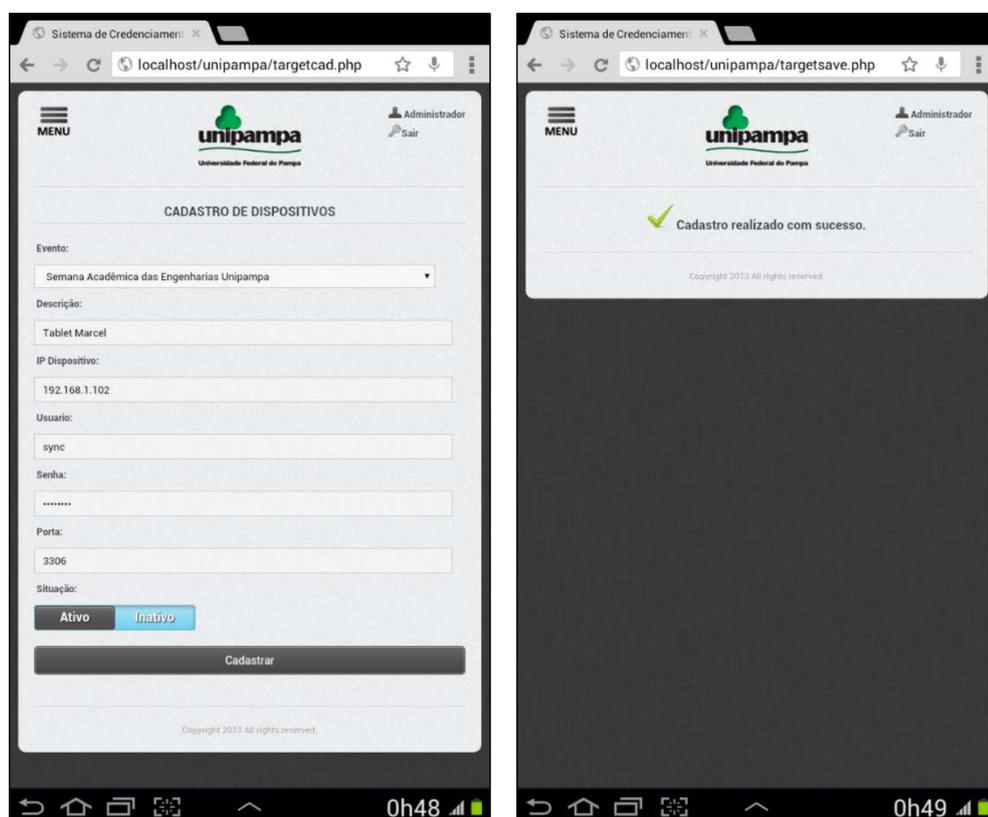
Figura 15 - Interfaces iniciais da aplicação.



Fonte: Primária (2013).

A figura 16 apresenta uma interface de cadastro de dispositivos, e uma mensagem de cadastro realizado. É importante ressaltar que somente o usuário com permissões de administração poderá acessar as interfaces de cadastro.

Figura 16 - Interfaces de cadastro.

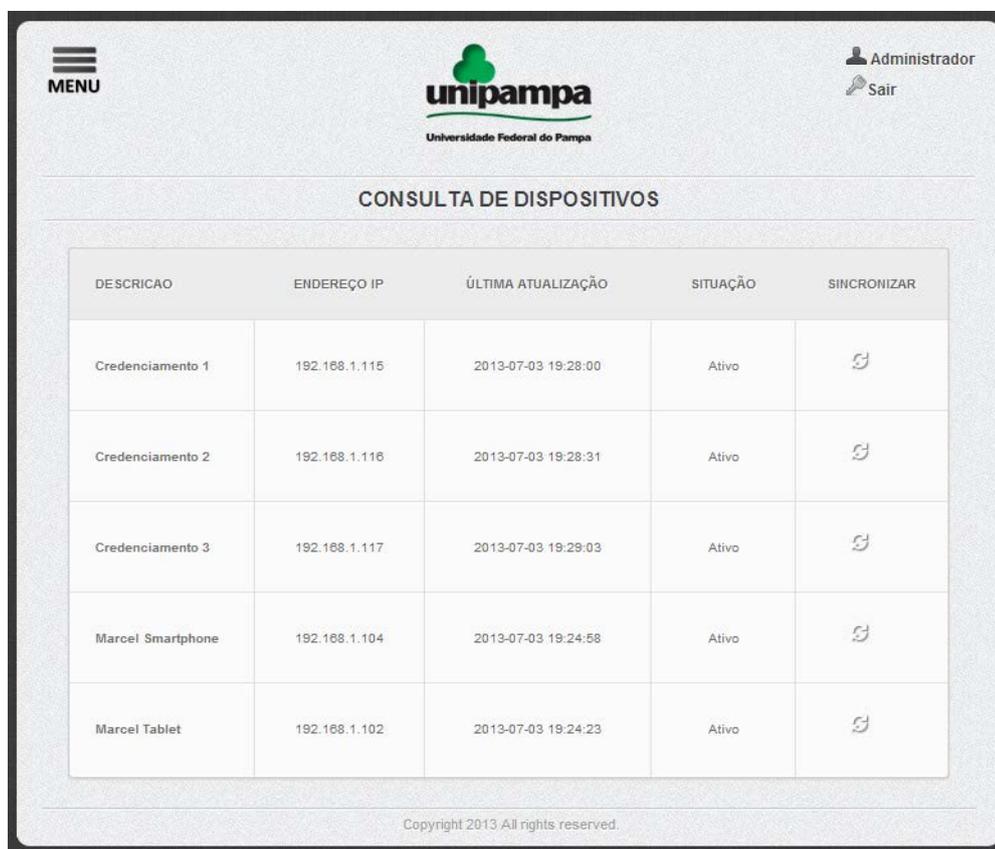


Fonte: Primária (2013).

### 3.5 Testes

Na funcionalidade de sincronização, foi implementada uma página de consulta de dispositivos cadastrados, onde é possível selecionar o dispositivo a ser sincronizado por meio de um ícone que representa a função, como mostra a figura 17. Após a seleção do dispositivo, a função ativa o arquivo PHP responsável por gerar o XML de parâmetros, o qual gera o arquivo e executa uma função para a execução da ferramenta SJA, passando como parâmetro o arquivo XML criado. Todo o processo mencionado acima está descrito também no diagrama de sequência da sincronização, ilustrado na figura 13.

Figura 17 - Interface de seleção de dispositivos.



| DESCRICAO         | ENDEREÇO IP   | ÚLTIMA ATUALIZAÇÃO  | SITUAÇÃO | SINCRONIZAR   |
|-------------------|---------------|---------------------|----------|---|
| Credenciamento 1  | 192.168.1.115 | 2013-07-03 19:28:00 | Ativo    |  |
| Credenciamento 2  | 192.168.1.116 | 2013-07-03 19:28:31 | Ativo    |  |
| Credenciamento 3  | 192.168.1.117 | 2013-07-03 19:29:03 | Ativo    |  |
| Marcel Smartphone | 192.168.1.104 | 2013-07-03 19:24:58 | Ativo    |  |
| Marcel Tablet     | 192.168.1.102 | 2013-07-03 19:24:23 | Ativo    |  |

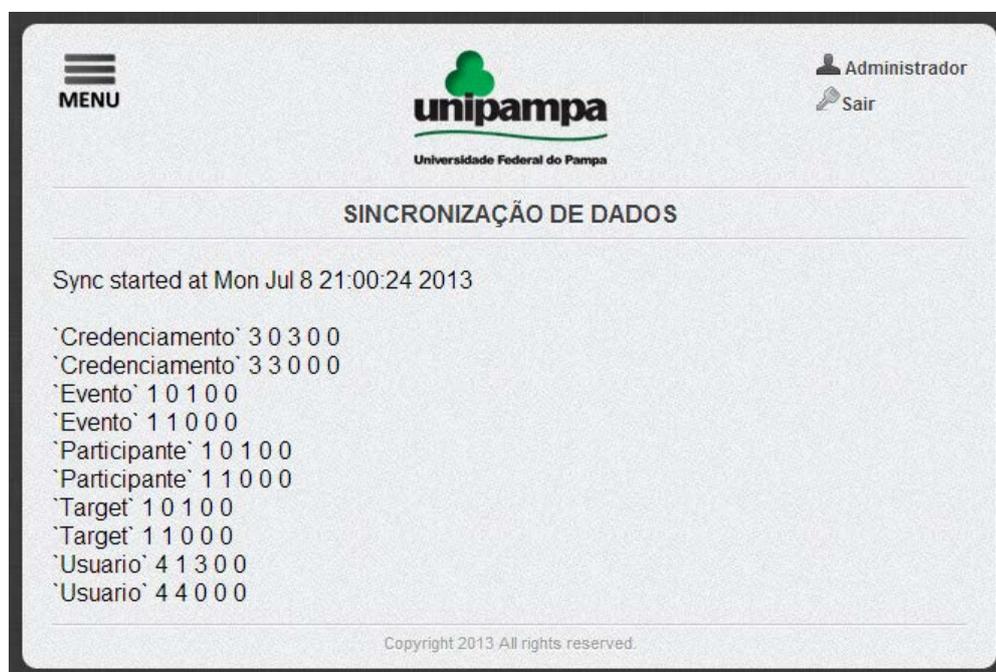
Fonte: Primária (2013).

O mesmo processo pode ser executado por meio do utilitário *cron*, criando a tarefa de sincronização de forma automatizada, observando que deve ser criada uma instrução no *cron* para cada dispositivo que deseja sincronizar.

Após a execução da ferramenta SJA, o aplicativo mostra o arquivo de *log* do SJA, retornando para o usuário a informação de como transcorreu a sincronização, como apresenta a figura 18. O arquivo de *log* exibe o nome da tabela, e as quantidades de registros na tabela local,

na tabela externa e registros inseridos, atualizados e excluídos, para cada tabela do banco de dados definido como *source* e *target*.

Figura 18 - Log da sincronização.



Fonte: Primária (2013).

Na etapa de execução dos testes foram identificados alguns erros de implementação de funções e algumas operações necessárias que não haviam sido planejadas inicialmente. Cada problema constatado foi solucionado após sua identificação, permitindo que o aplicativo esteja em condições de ser utilizado para o fim proposto.

#### 4 CONCLUSÕES E CONSIDERAÇÕES FINAIS

No domínio deste trabalho, a mobilidade é uma característica principal. Uma visão das tecnologias atuais permite considerar a possibilidade de contar com dispositivos móveis, para efetuarem a integração em tempo real dos dados inseridos nos dispositivos móveis com um BD localizado em um servidor central.

A alta disponibilidade e consistência de dados são de grande importância no desenvolvimento de soluções com banco de dados distribuídos para dispositivos móveis. A funcionalidade de replicação de dados nativa do MySQL e outras soluções existentes no mercado, apesar de apresentarem um conjunto bastante completo de funcionalidades, requerem a permissão de acesso aos arquivos de configuração do sistema, o que não é permitido por definição pelo sistema operacional *Android*, e adicionalmente impedem o suporte à sincronização *multimaster*.

A inovação tecnológica advinda dos esforços de pesquisa deste trabalho apresentaram uma contribuição permitindo a sincronização *multimaster* sob o banco de dados MySQL, até então considerada como não suportada. Com efeito aplica-se com sucesso as operações propostas, sendo bastante útil para cenários onde se deseja efetuar uma sincronização *multimaster* de bancos de dados distribuídos em diversos dispositivos móveis. Sua interface gráfica foi montada de forma a permitir uma prática usabilidade e uma fácil identificação das funcionalidades existentes.

Por se tratar de uma aplicação implementada sob o conceito *open-source*, futuramente pode ser aperfeiçoada para possibilitar mais funcionalidades de sincronização ainda não conceitualizadas. Pode-se considerar também a construção de um conjunto de métricas baseados em modelos matemáticos, os quais já foram parcialmente pensados baseados nas variáveis e relacionamento entre elas, para estabelecer medidas de desempenho do ambiente apresentado.

## 5 REFERÊNCIAS

AGARWAL, S.; STAROBINSKI, D.; TRACHTENBERG, A. On the scalability of data synchronization protocols for PDAs and mobile devices. **IEEE Network**, v. 16, n. 4, p. 22–28, 2002. Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1020232>> Acessado em: 28/06/2013.

APACHE. Apache HTTP Server Project. Disponível em: <<http://httpd.apache.org/>> Acessado em: 03/07/2013.

B'FAR, R. **Mobile Computing Principles**. New York, USA: Cambridge, 2005.

COMSCORE. U.S. Market Share by OS. ,2012. comScore. Disponível em: <<http://www.comscore.com/>> Acessado em: 21/06/2013.

DB-ENGINES. DB-Engines Ranking - Historical Trend. ,2013. Disponível em: <[http://db-engines.com/en/ranking\\_trend](http://db-engines.com/en/ranking_trend)> Acessado em: 28/06/2013.

DENNY, M.; FRANKLIN, M. J. Edison: Database-Supported Synchronization for PDAs. **Distributed and Parallel Databases**, v. 15, n. 2, p. 95–116, 2004. Disponível em: <<http://link.springer.com/10.1023/B:DAPD.0000013067.55166.a2>> Acessado em: 15/06/2013.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 4ª ed. São Paulo, Brasil, 2005.

GOOGLE. Banco Mundial. ,2013. Google. Disponível em: <<http://www.google.com/publicdata/>> Acessado em: 20/06/2013.

KALKOV, I.; FRANKE, D.; SCHOMMER, J. F.; KOWALEWSKI, S. A Real-time Extension to the Android Platform. **Java Technologies for Real-time and Embedded Systems**, p. 105–114, 2012.

KUMAR, V. **Mobile Database Systems**. New Jersey, USA: Wiley, 2006.

LAURSEN, P. **Introduction to the SQLyog Job Agent (SJA)**. 2009.

LAURSEN, P.; TON, Q. Using SQLyog Enterprise to Effectively Synchronize MySQL Databases. ,2011. Disponível em: <[https://static.webyog.com/pdfs/Using\\_SQLyog\\_Enterprise\\_to\\_Effectively\\_Synchronize\\_MySQL\\_Databases.pdf](https://static.webyog.com/pdfs/Using_SQLyog_Enterprise_to_Effectively_Synchronize_MySQL_Databases.pdf)> Acessado em: 05/07/2013.

MEIER, R. Communication paradigms for mobile computing. **ACM SIGMOBILE Mobile Computing and Communications Review**, v. 6, n. 4, p. 56–58, 2002. Disponível em: <<http://portal.acm.org/citation.cfm?doid=643550.643555>> Acessado em: 15/06/2013.

MYSQL. MySQL Reference Manuals. Disponível em: <<http://dev.mysql.com/doc/>>. Acessado em: 3/7/2013.

NARAMORE, E.; GERNER, J.; SCOUARNEC, Y. LE. **Beginning PHP5 , Apache , and MySQL Web Development**. Wrox, 2005.

NIELSEN. The Mobile Media Report. ,2011. Disponível em: <<http://www.nielsen.com/>> Acessado em: 19/06/2013.

OZSU, M. T.; VALDURIEZ, P. **Principles of Distributed Database Systems**. 3rd ed. New York, USA: Springer, 2011.

PALAZZO, S.; PULIAFITO, A.; SCARPA, M. Design and evaluation of a replicated database for mobile systems. **Wireless Networks** 6, v. 6, p. 131–144, 2000.

PALME, E.; TAN, C.; SUTANTO, J.; PHANG, C. W. Choosing the Smart Phone Operating System For Developing Mobile Applications. **International Center For Electronic Commerce**, p. 146–152, 2010.

PHP. Manual do PHP. Disponível em: <<http://www.php.net/manual/>> Acessado em: 4/7/2013.

RAHIMI, S. K.; HAUG, F. S. **Distributed Database Management Systems A Practical Approach**. Wiley, 2010.

RAMYA, B. S.; KODURI, S. B.; SEETHA, M. A Stateful Database Synchronization Approach for Mobile Devices. **International Journal of Soft Computing and Engineering (IJSCE)**, v. 2, n. 3, p. 316–320, 2012.

SCHWARTZ, B.; ZAITSEV, P.; TKACHENKO, V.; et al. **High Performance MySQL**. 2nd ed. California, USA: O'Reilly, 2008.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 3ª Edição ed. São Paulo, Brasil: Makron Books, 1999.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Database System Concepts**. 6th ed. McGraw-Hill, 2011.

TONIN, G. S. **Tendências em Computação Móvel**. São Paulo, Brasil, 2012.

W3TECHS. Usage of web servers for websites. ,2013a. Disponível em: <[http://w3techs.com/technologies/overview/web\\_server/all](http://w3techs.com/technologies/overview/web_server/all)> Acessado em: 01/07/2013.

W3TECHS. Usage of server-side programming languages for websites. ,2013b. Disponível em: <[http://w3techs.com/technologies/overview/programming\\_language/all](http://w3techs.com/technologies/overview/programming_language/all)> Acessado em: 02/07/2013.