Framework de replicação de dados com criptografia simétrica utilizando notificações para Android.

Acadêmico - Fernando Klock

Orientador – Dalton Solano dos Reis

Roteiro

- Introdução
- Fundamentação teórica
- Desenvolvimento
- Resultados e discussões
- Conclusão e Extensões
- Demonstração

Introdução

- Objetivos do trabalho
  - Demonstrar o desempenho da plataforma Android na replicação de dados utilizando a biblioteca SQLite
  - Aplicar a segurança da informação com a utilização de criptografia simétrica na transferência de dados
  - Utilizar o conceito de *push* para facilitar o processo de notificação de possíveis atualizações no dispositivo móvel

Fundamentação Teórica

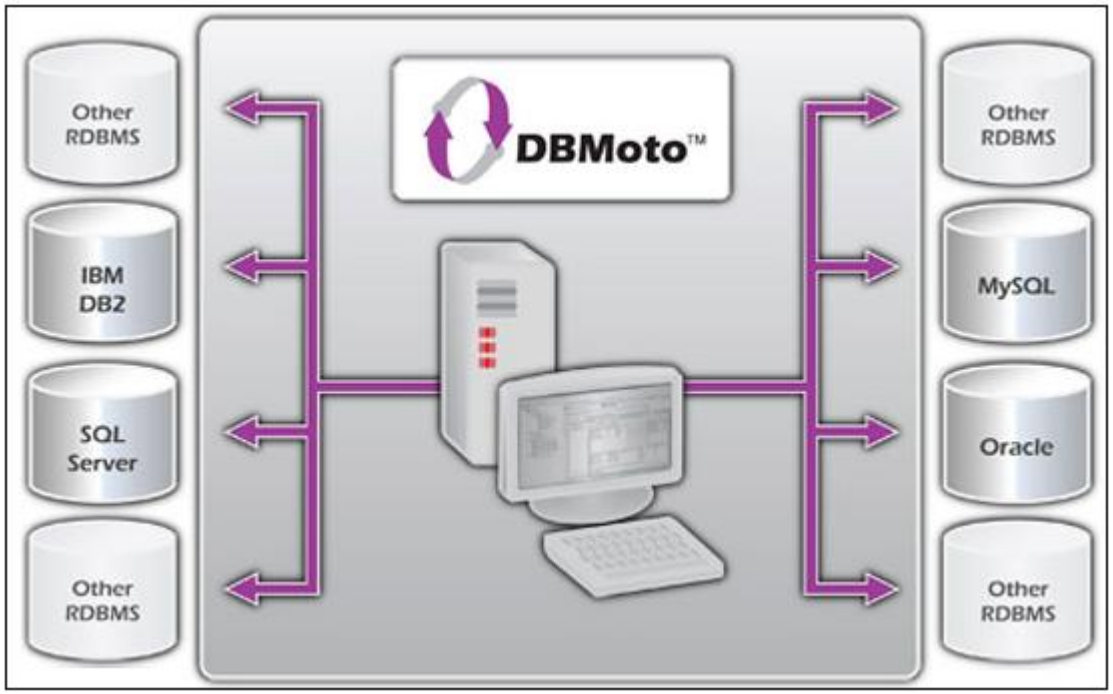
- Plataforma Android
  - Arquitetura: Linux Kernel, Android Runtime, Libraries, Application Framework, Application
  - SQLite
- Criptografia
  - Criptografia Assimétrica
  - Criptografia Simétrica
  - Tamanhos distintos de chaves

Fundamentação Teórica

- *Push* (Notificações)
  - Notificar aplicações móveis através de um servidor
  - Objetivos: - atualizar versões e dados
    - avisar a ocorrência de algum evento
- Socket
  - Canal de comunicações entre dois host
  - Métodos: - UDP (*User Datagram Protocol*)
    - TCP/IP (*Transmission Control Protocol / Internet Protocol)*

Fundamentação Teórica

- Trabalhos correlatos
  - DBMoto



Fundamentação Teórica

- Trabalhos Correlatos
  - Heros
    - Framework desenvolvido na PUC para gerenciar replicação entre banco de dados Heterogêneos
    - Disponibiliza comunicação com protocolos RPC, JSON e socket
    - Composto por seis frameworks: interface, esquema, consulta, transação, comunicação e regra

Desenvolvimento

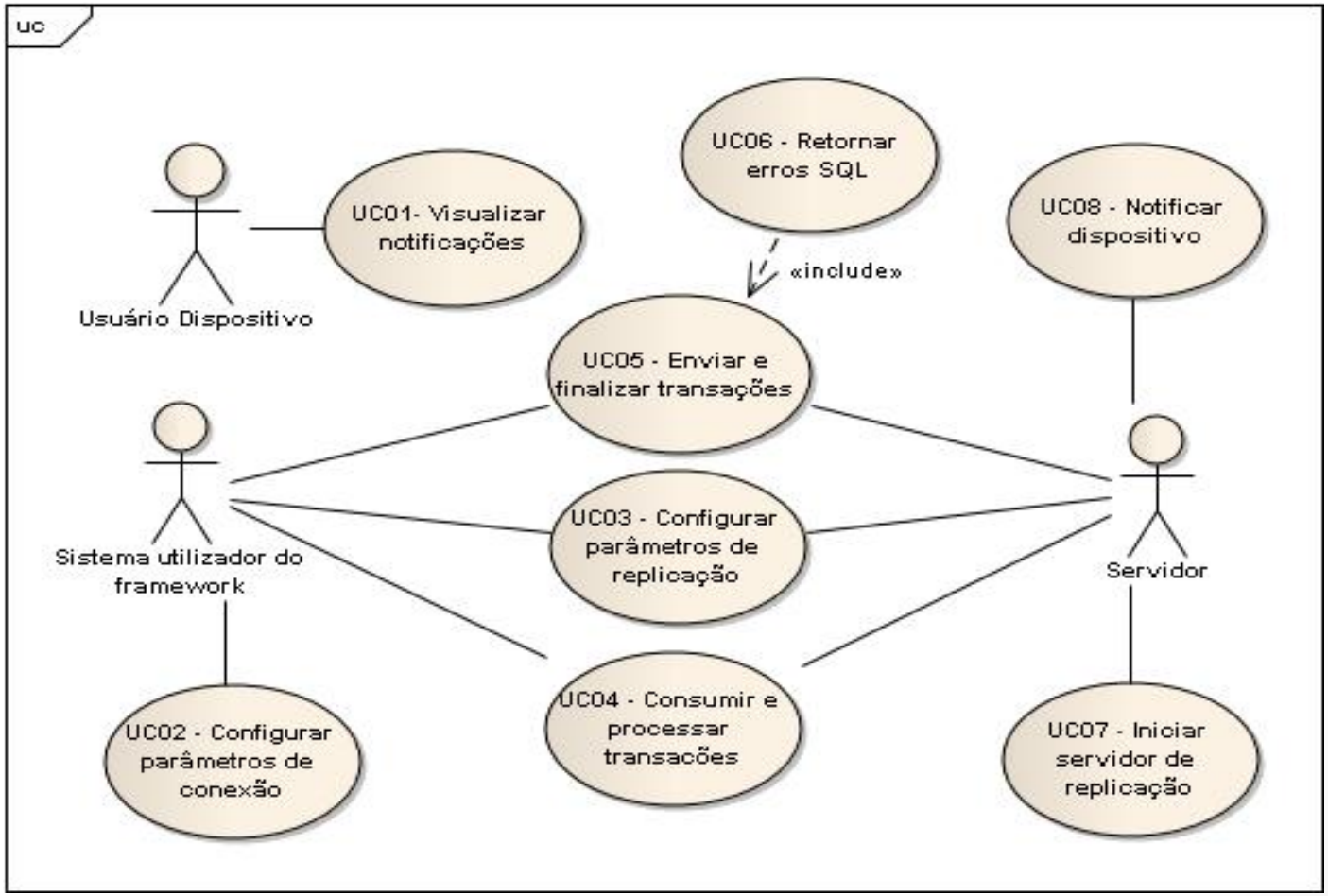
- Requisitos
  - Permitir replicar a base de dados SQLite do dispositivo para o servidor e do servidor para o dispositivo (RF01)
  - Permitir receber notificações no dispositivo com alerta de dados a serem replicados do servidor (RF02)
  - Permitir o envio de notificação do servidor para o dispositivo (RF03)
  - Permitir represar dados no dispositivo e no servidor quando os mesmos não tiverem acesso a web (RF04)
  - Permitir criptografar dados para serem replicados (RF05)
  - Permitir descriptografar dados recebidos (RF06)
  - Permitir alterar o tamanho de chave para criptografia (RF07)

Desenvolvimento

- Requisitos
  - O sistema será desenvolvido na linguagem Java (RNF01)
  - O sistema utilizará a biblioteca SQLite (RNF02)
  - O sistema deve ser compatível com o sistema operacional Android 2.2 ou posterior (RNF03)

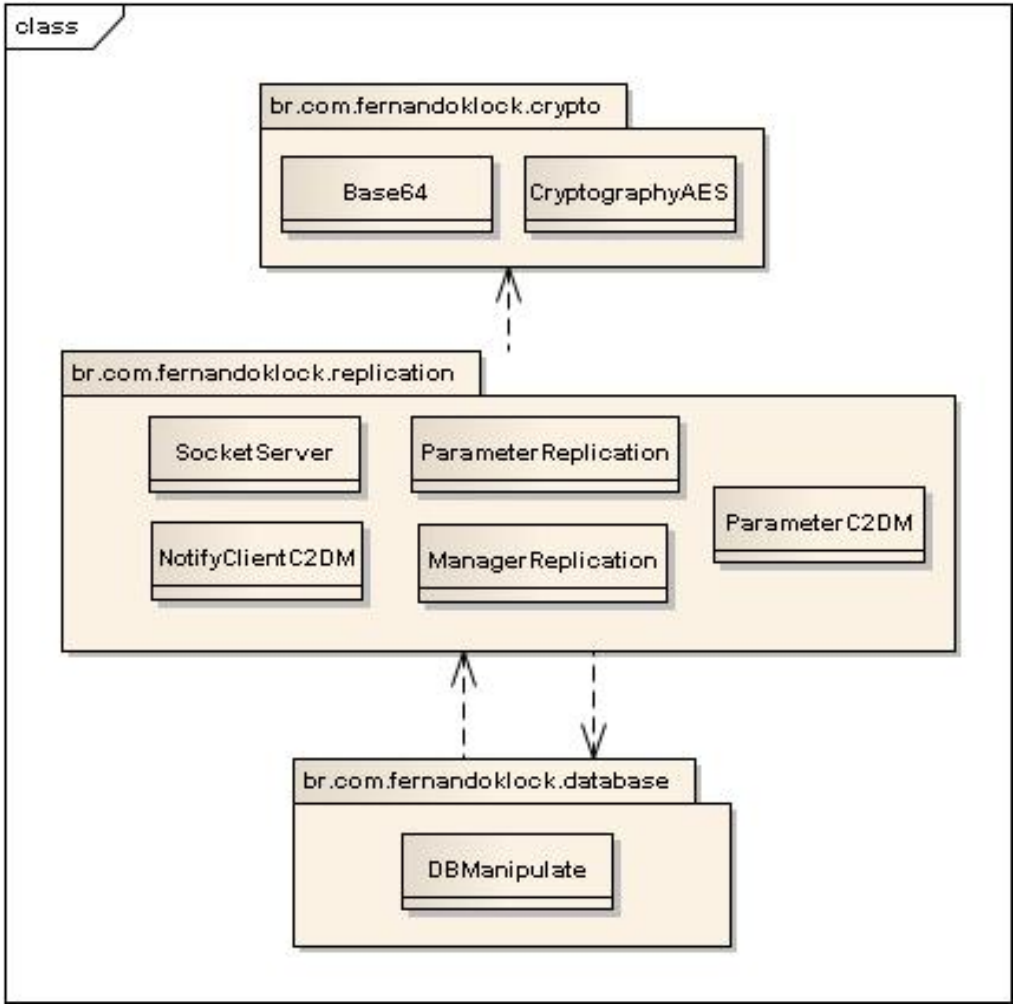
Desenvolvimento

- Especificação
  - Enterprise Architect



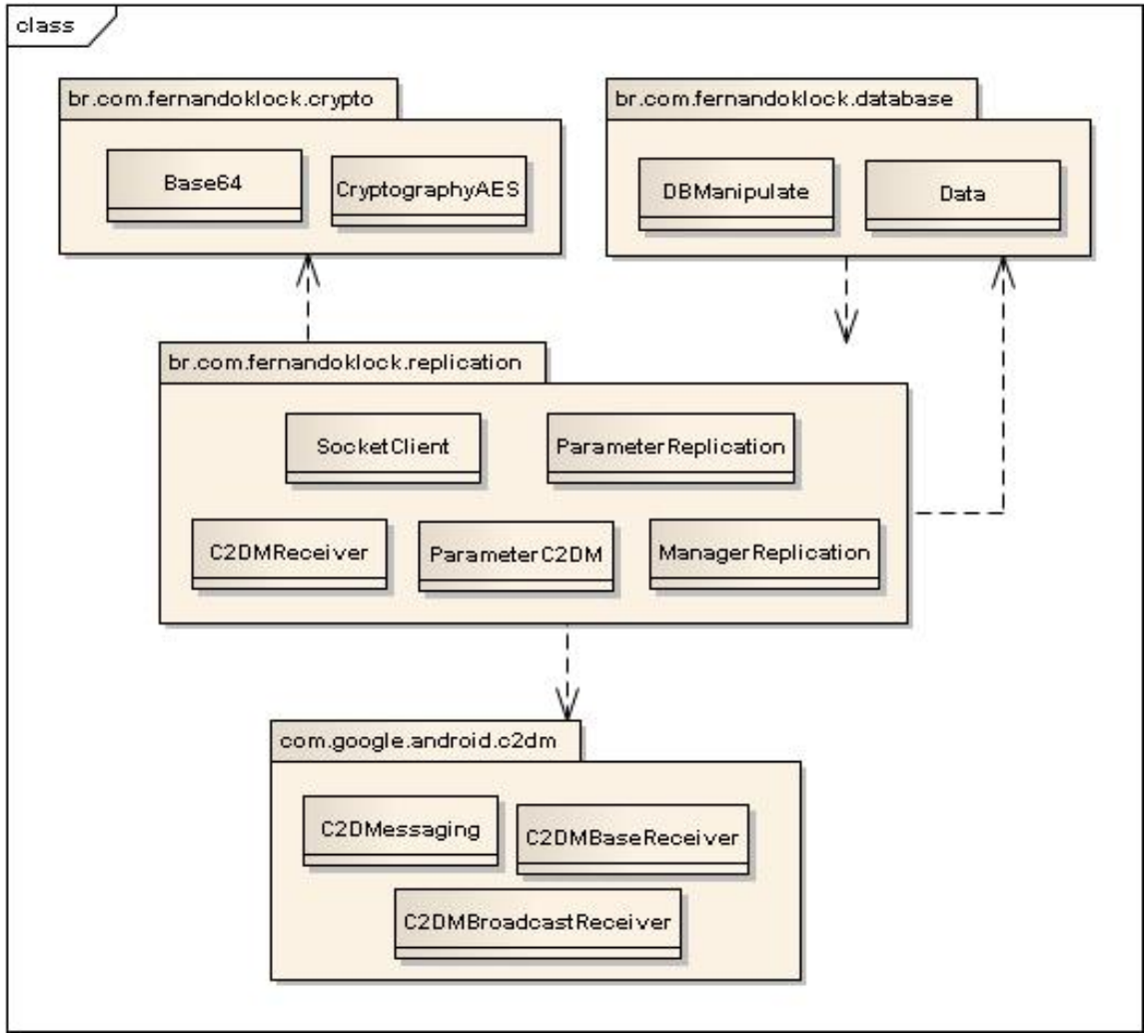
Desenvolvimento

- Pacotes do Servidor



Desenvolvimento

- Pacotes do Dispositivo



Desenvolvimento

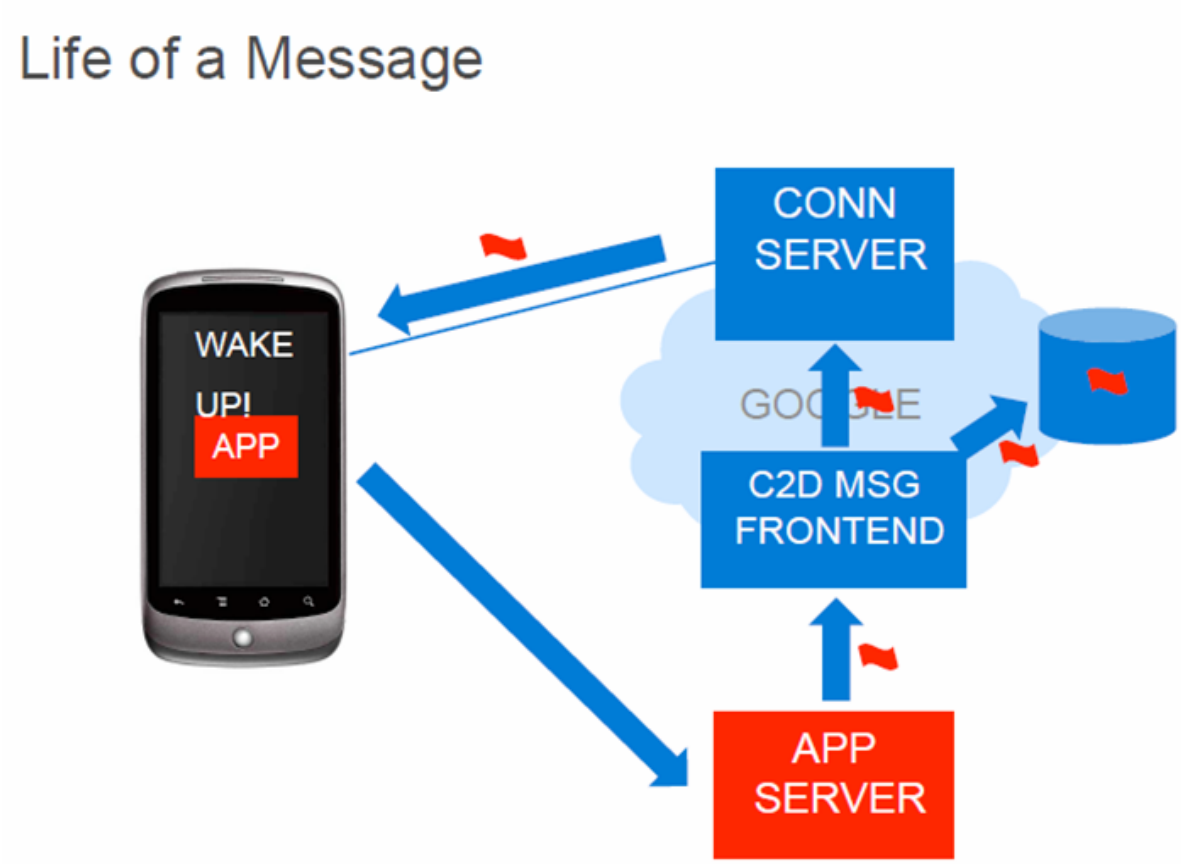
- Técnicas e ferramentas utilizadas
  - Eclipse
  - Android Development Tools (ADT)
  - Emulador
  - Motorola Milestone (Android 2.2)

Desenvolvimento

- Criptografia
  - Algoritmo AES (*Advanced Encryption Standard*)
  - API *javax.cripto*
  - Chaves de 128, 192 e 256 bits
- Gerenciamento de replicação
  - Bancos de dados *Source*
  - Banco de dados *Target*
  - *After-image:* banco DBReplication

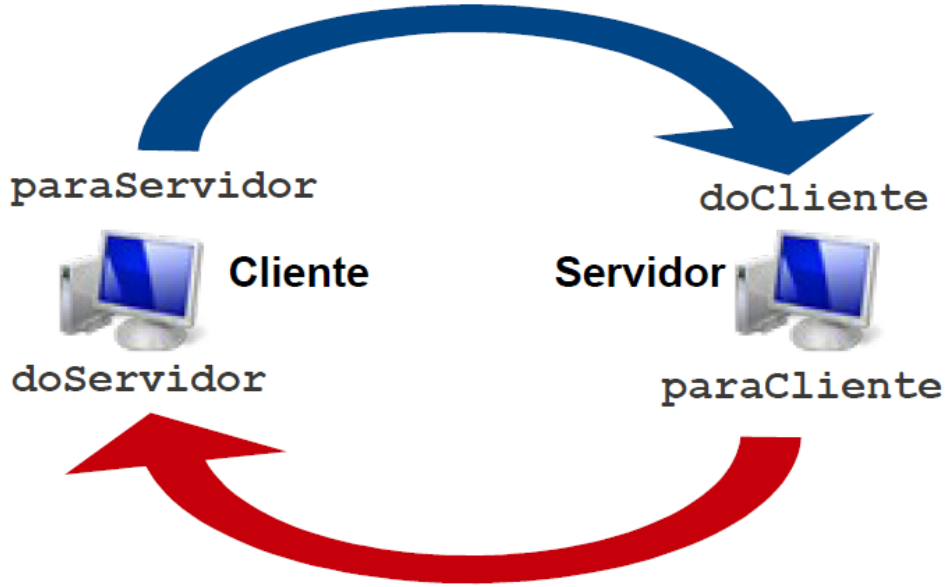
Desenvolvimento

- *Push* (Notificações)



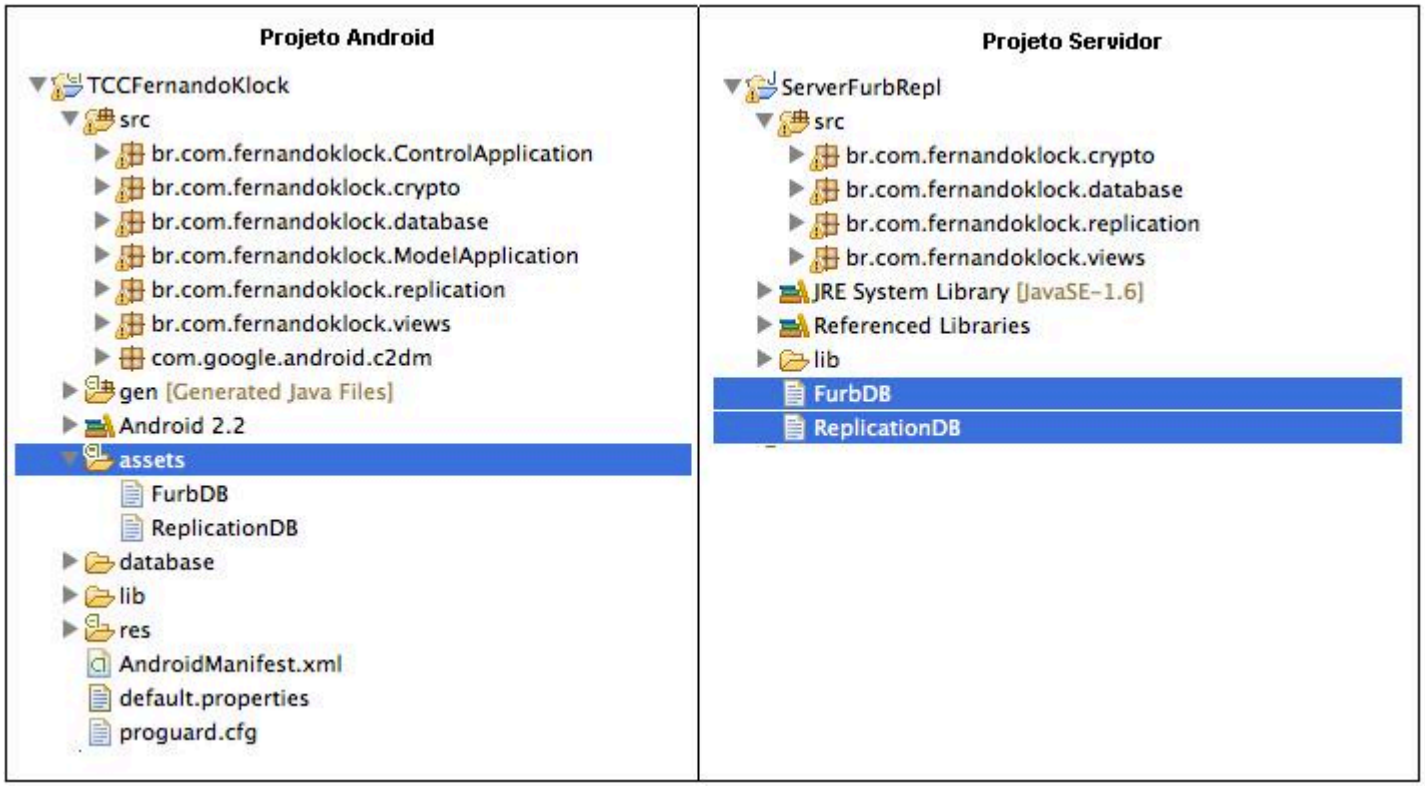
Desenvolvimento

- Socket
  - Dispositivo móvel conecta no servidor
  - Troca de *tags* entre os equipamentos
  - Validação para chave de criptografia e tipo de banco de dados



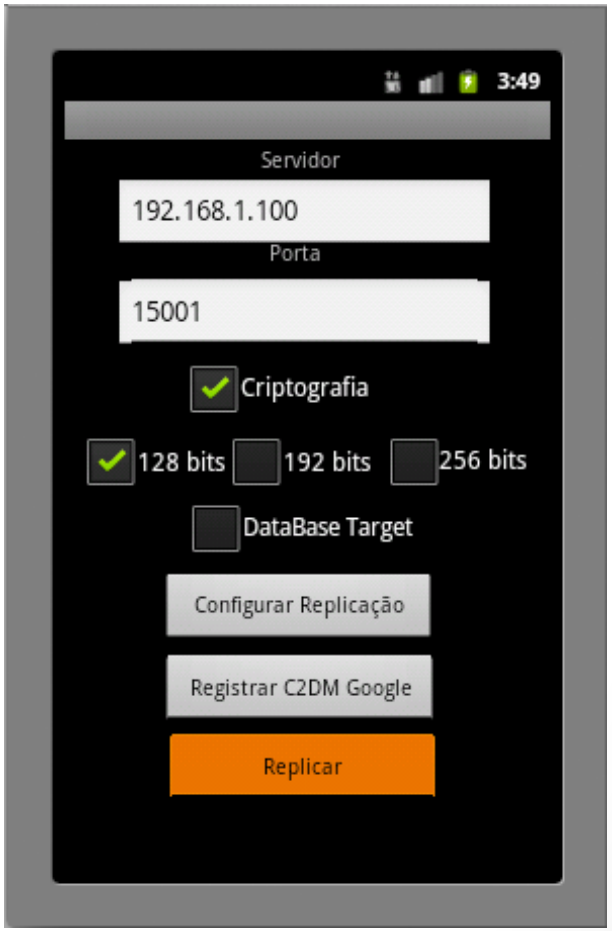
Desenvolvimento

- Operacionalidade da aplicação



Desenvolvimento

- Operacionalidade da aplicação



Resultados e Discussões

- Socket
  - *Listener* aberto no servidor para recepção dos dados
  - Agil ao replicar dados entre os integrantes da replicação
  - Baixa segurança no servidor (porta aberta)

- Utilização do SQLite no Android
  - Chamada de métodos distintos para cada comando SQL executado (*insert*, *update*, *delete* e *select*)
  - Lento ao executar grandes transações

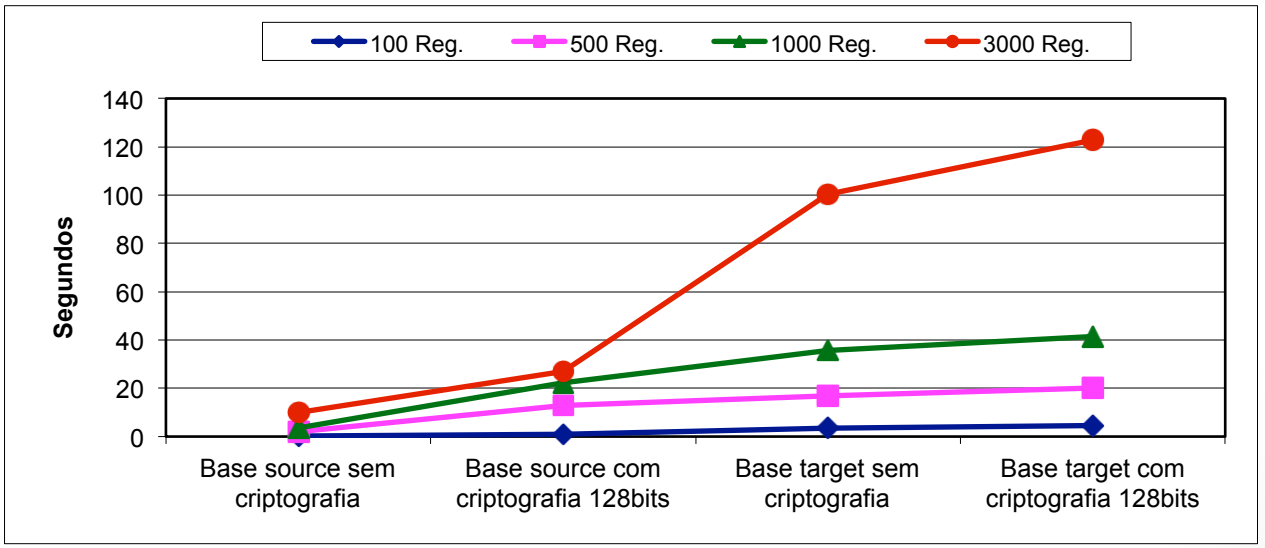
Resultados e Discussões

- *Push* (Notificações)
  - Seguro na entrega de notificações
  - Dependência dos servidores da Google
  - Economia de bateria e CPU do dispositivo na ocorrência de atualizações

- Criptografia
  - Chaves de 128, 192 e 256 bits (Objetivo inicial era somente 128 bits)
  - Utilização da classe Base64 para transferência de strings

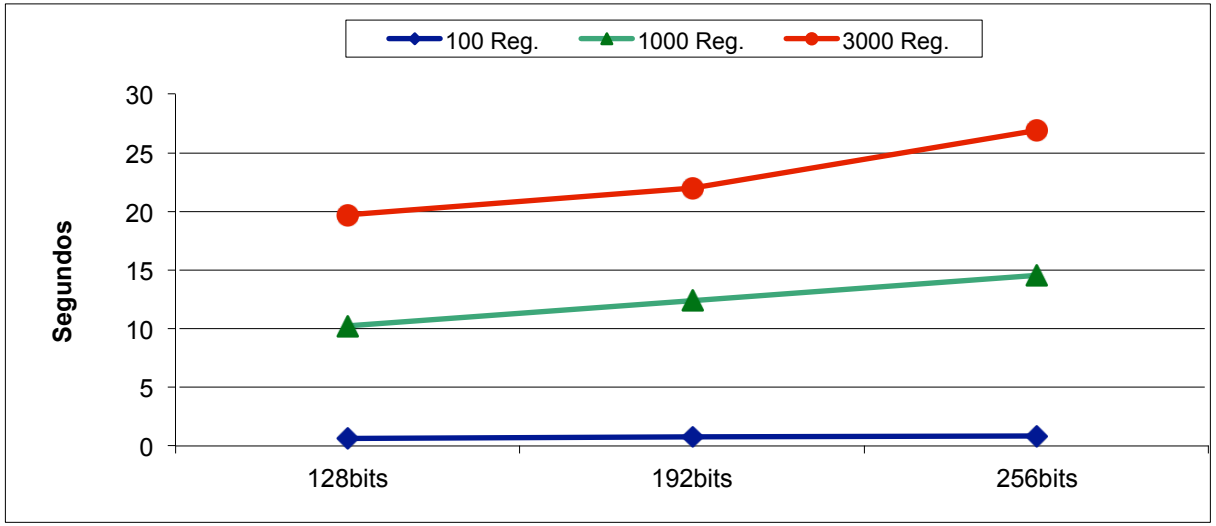
Resultados e Discussões

DESEMPENHO DA TRAMISSÃO DE DADOS (EM SEGUNDOS)				
Quantidade de registros replicados	**Base *source* sem criptografia**	**Base *source* com criptografia 128 bits**	**Base *target* sem criptografia**	**Base *target* com criptografia 128 bits**
100	0,18	1,03	3,31	4,71
500	1,96	12,7	16,82	20,22
1000	3,61	22,34	35,63	41,51
3000	9,96	27,12	100,32	122,67



Resultados e Discussões

DESEMPENHO DA CRIPTOGRAFIA SIMÉTRICA (EM SEGUNDOS)				
Quantidade de registros criptografados	**Chave de 128 bits**	**Chave de 192 bits**	**Chave de 256 bits**	**Diferença 128 bits vs 256 bits**
100	0,64	0,76	0,82	21,90%
500	4,91	5,57	6,73	27,04%
1000	10,24	12,36	14,54	29,57%
3000	19,71	22,01	26,89	26,70%



Conclusão

- Sincronismo das bases de dados: Servidor <-> Dispositivo
- Não apresenta sincronismo em tempo real das bases de dados
- Push se demonstrou muito útil para as solicitações de atualizações
- Semelhança com trabalhos correlatos *refresh* (DBMoto), *socket* (Heros)

Extensões

- Disponibilizar a segurança com a utilização de criptografia assimétrica
- Disponibilizar a replicação de dados entre bancos heterogêneos
- Disponibilizar replicação de dados entre dispositivos móveis, sem o uso do servidor
- Disponibilizar um algoritmo de sincronismo entre os bancos de dados de modo a garantir que não haja erros na replicação

Demonstração

DESEMONSTRAÇÃO NO EMULADOR