Desenvolvimento de um Framework para replicação de dados entre bancos heterogêneos

João Batista Gianisini Júnior

Alexander Roberto Valdameri - Orientador

Roteiro da apresentação

- Introdução
- Objetivos
- Motivação
- Revisão bibliográfica
- Especificação
- Implementação
- Operacionalidade
- Conclusões
- Relevância Pessoal

A1

Introdução

- Organizações com estrutura física descentralizada necessitam de recursos para compartilhamento dos dados:
- Replicação de dados;
- Heterogeneidade entre os SGBDs participante;
- Comunicação assíncrona;
- Integração do Hibernate e Java Message Service(JMS).

Slide 3

A1 Acho melhor vc colocar menos texto... coloque itens... e entao vc fala sobre eles... tem bastante texto em alguns slides...
Alexander; 22/6/2006

Objetivos

- Implementar um framework que servirá como uma camada adicional entre a aplicação e a base de dados;
- Interceptar e replicar as transações de um banco de dados local para um banco de dados remoto, respeitando a seqüência dos comandos aplicados;
- Encapsular no framework todo processo de replicação.

Motivação

- Muitas empresas estão presas a mecanismos de replicação de dados dos SGBDs comerciais. Estas tecnologias não permitem que seja utilizado este recurso entre outros SGBDs distintos, como por exemplo a replicação entre uma base ORACLE e MySQL.

A2

Replicação de dados

- Manter duas ou mais cópias em diferentes servidores;
- Pode ser replicado um conjunto de tabelas, ou até mesmo aplicar filtros sobre as linhas em que se deseja replicar;
- Replicação provê melhor desempenho e maior disponibilidade.

Slide 6

A2 Tens que padronizar os titulos dos slides... a fonte e o tamanho... é bom que sejam sempre os mesmos... dentro do possivel eh claro...
Alexander; 22/6/2006

Análise dos Requisitos

- Latência: tempo permitido para que haja sincronismos entre os participantes;
- Autonomia: dados suficientes para que o participante possa operar, mesmo com indisponibilidade do servidor central;
- Conflitos de atualização: o mesmo registro sendo alterado ao mesmo tempo em réplicas diferentes;
- Disponibilidade física: o link disponível deve ser suficiente ao volume de dados.

Tipos de Arquitetura

- Não replicados: somente uma cópia do dado na rede;
- Totalmente replicados: todos os participante possuem todos os dados;
- Parcialmente replicados: o dado pode estar alocado em um ou mais participantes.

Hibernate

- Solução Java para gerenciamento de dados persistentes;
- Camada entre o aplicativo e um banco de dados relacional;
- Realiza ORM: persistência de um objeto Java para tabelas de um banco de dados relacional.
- Automatização da compatibilidade de SGBDs.

Base de dados suportadas pelo Hibernate

- DB2
- HSQL DB
- HypersonicSQL
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- SAP DB
- Sybase
- Timesten
- Apache DerbyHP
- NonStop SQL/MX 2.0
- Firebird
- FrontBase
- Informix
- Ingres
- Interbase
- Mckoi SQL
- Pointbase Embedded
- Progress

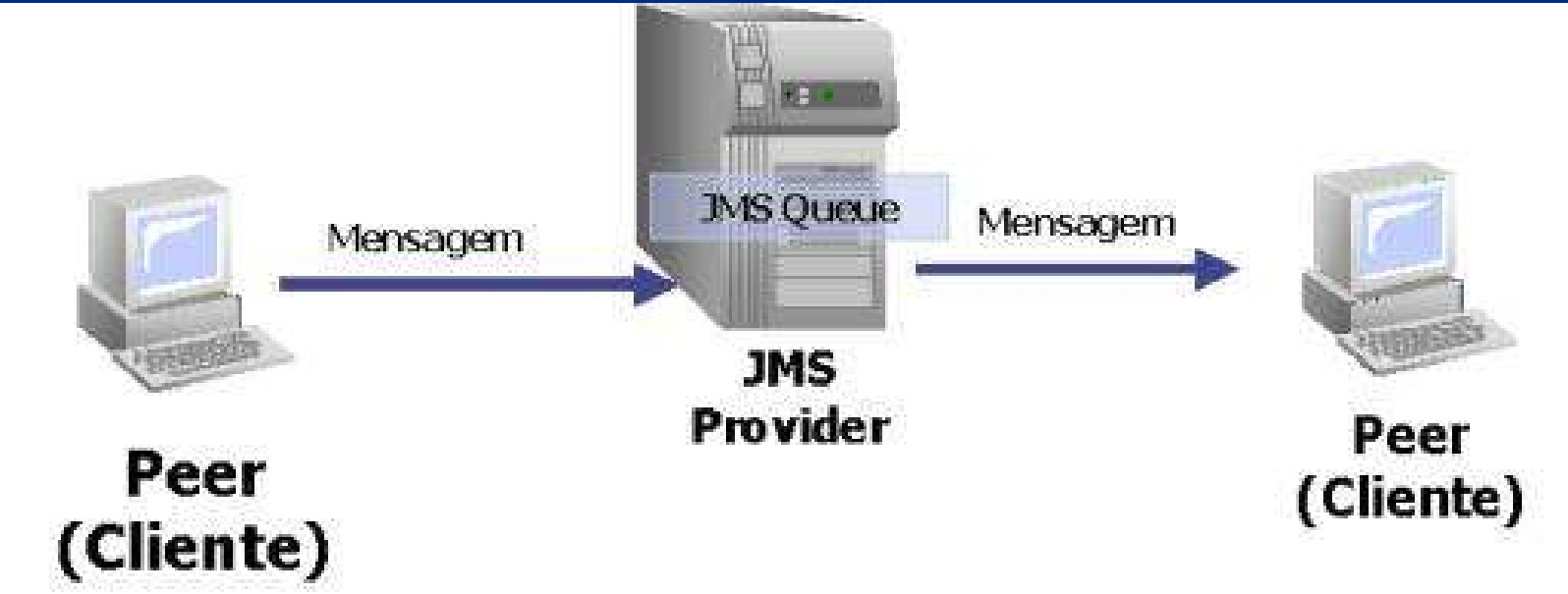
Java Message Service(JMS)

- É uma API da linguagem Java que permite aplicações como criar enviar e ler mensagens;
- Auxilia em situações onde a conexão é fornecida de forma intermitente;
- Suporta a comunicação de forma síncrona ou assíncrona.

Provider e Cliente JMS

- **Provider:** são aplicações servidoras, desenvolvidas por empresas especializadas que implementam as interfaces do padrão JMS, por exemplo SonicMQ, MQSeries, OpenJMS;
- **Cliente:** são programas altamente especializados que ao se comunicar com um provider pode produzir e consumir mensagens.

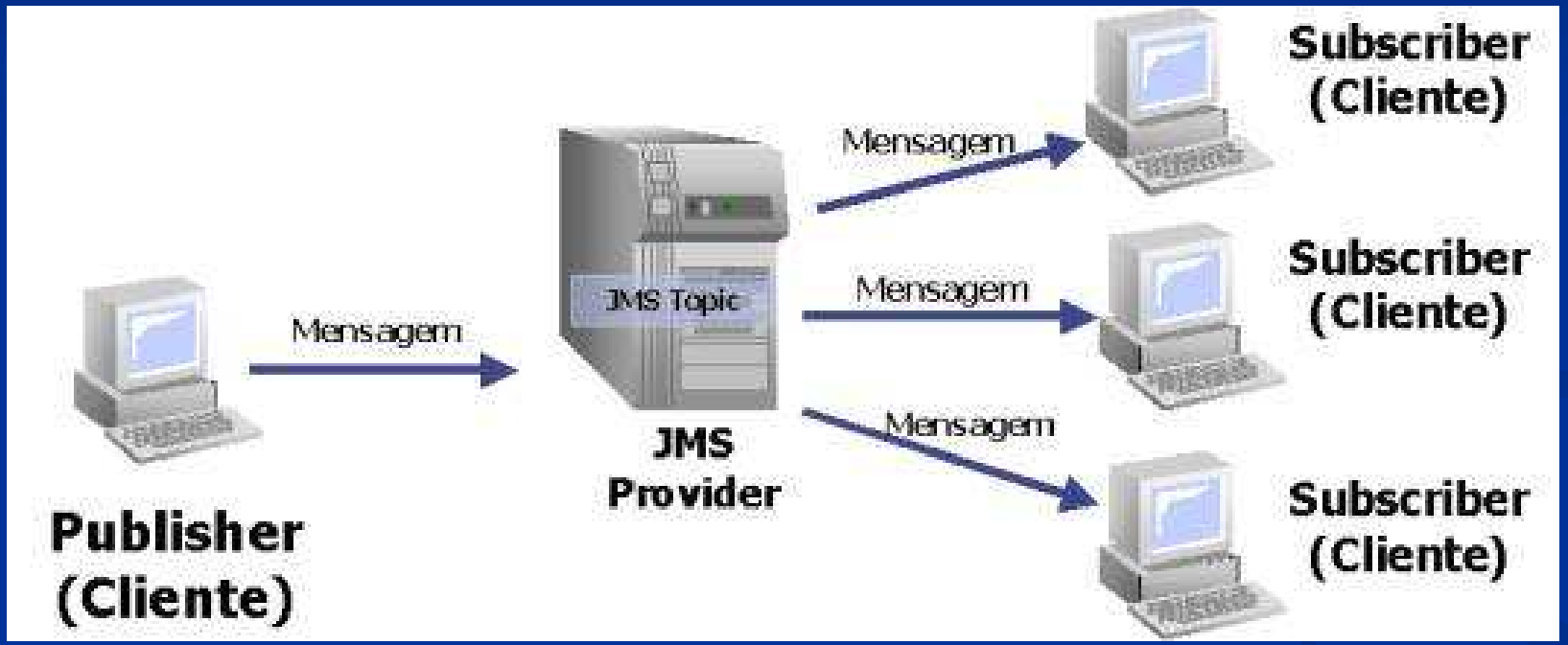
Domínio Ponto a Ponto(P2P)



Fonte: Wayne e Lima (2004, p. 1).

A3

Domínio publish/subscribe



Fonte: Wayne e Lima (2004, p. 1).

Slide 14

A3 nos trabalhos correlatos... recomendo colocar apenas em um slides e somente uma referencia aos trabalhos... e entao vc falaria brevemente sobre cada um deles...

Alexander; 22/6/2006

TRABALHOS CORRELATOS

- Postgres-R: é um projeto *open source* que implementa um mecanismo de replicação de dados entre bancos Postgres. Preserva os eventos transacionais e não replica consultas;
- Heros: é um framework que possibilita replicação entre bancos e aplicações heterogêneas. Possibilita também que seja utilizado vários protocolos de comunicação diferentes.

ESPECIFICAÇÃO

Diagrama de Casos de Uso

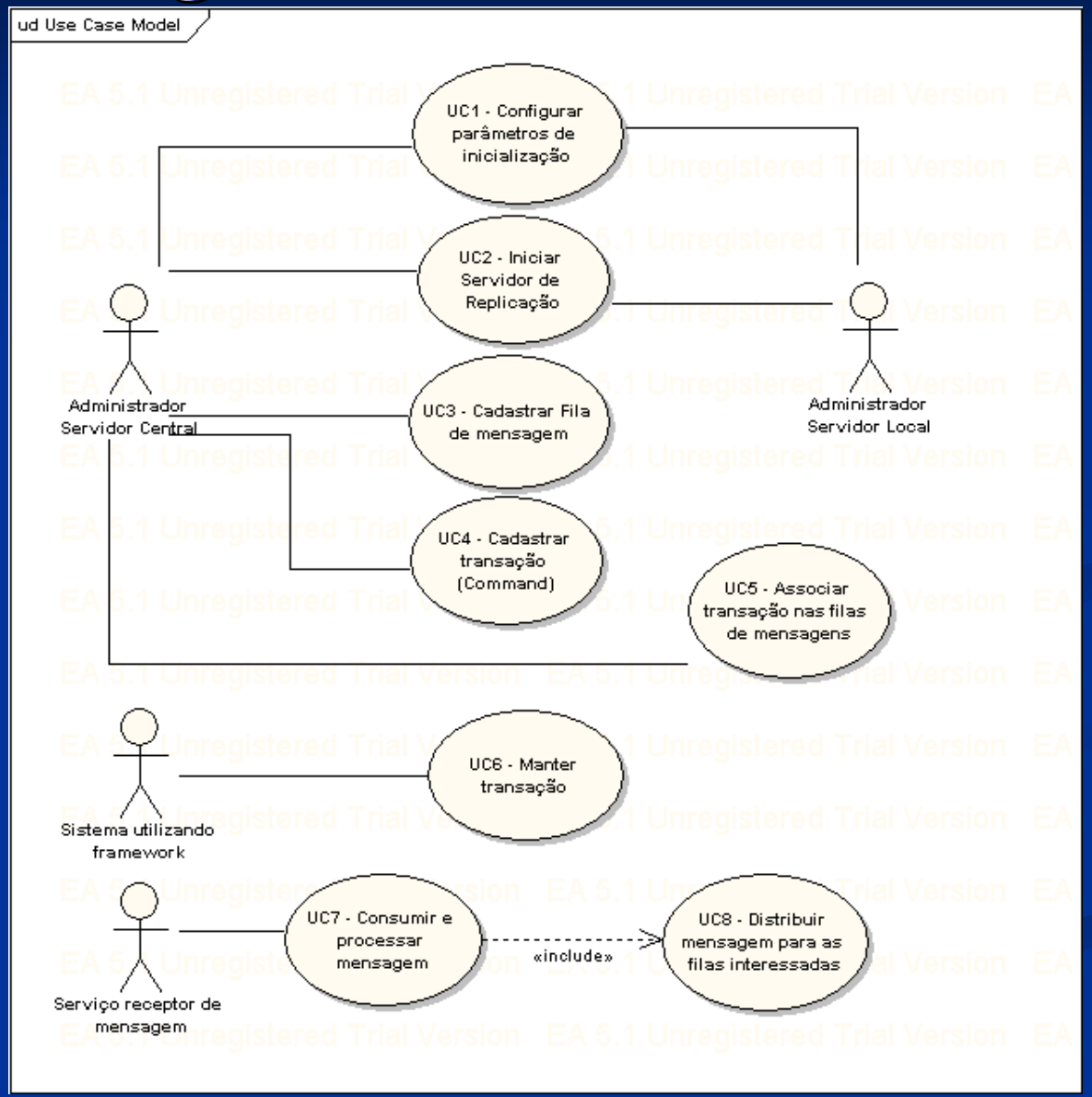


Diagrama de Classes da camada de persistência

- Interfaces para persistência dos dados e gerenciamento de sessão e transação;
- Fraco acoplamento com a tecnologia de persistência;
- Classe que encapsula a comunicação com a camada de replicação;

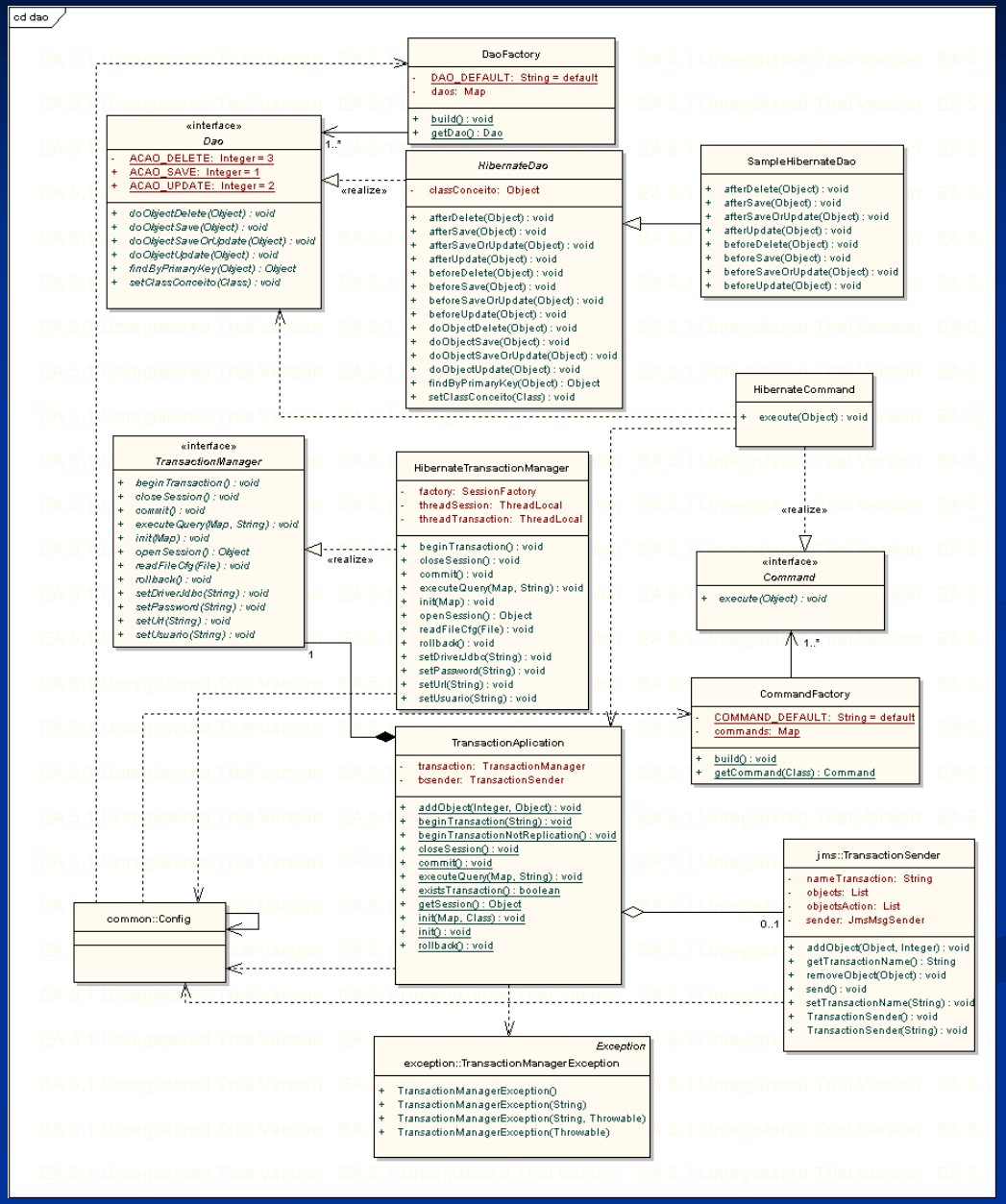


Diagrama de Classes da camada de replicação

- Garantia que somente uma conexão será aberta com cada provider;
- Compartilhamento de conexão;
- Recurso para filtrar qual transação deve ser replicada;
- Serviço receptor de mensagem autônomo;

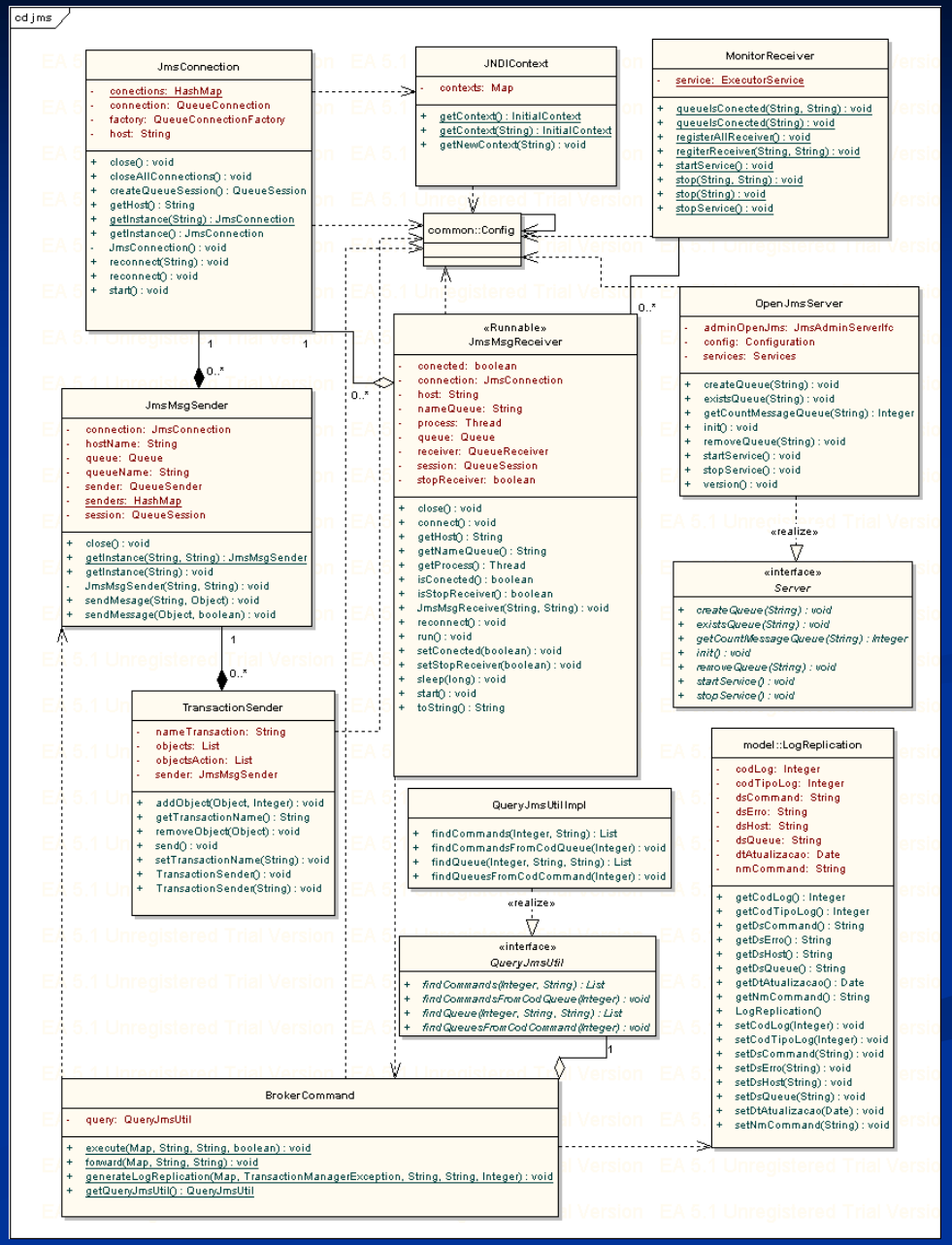
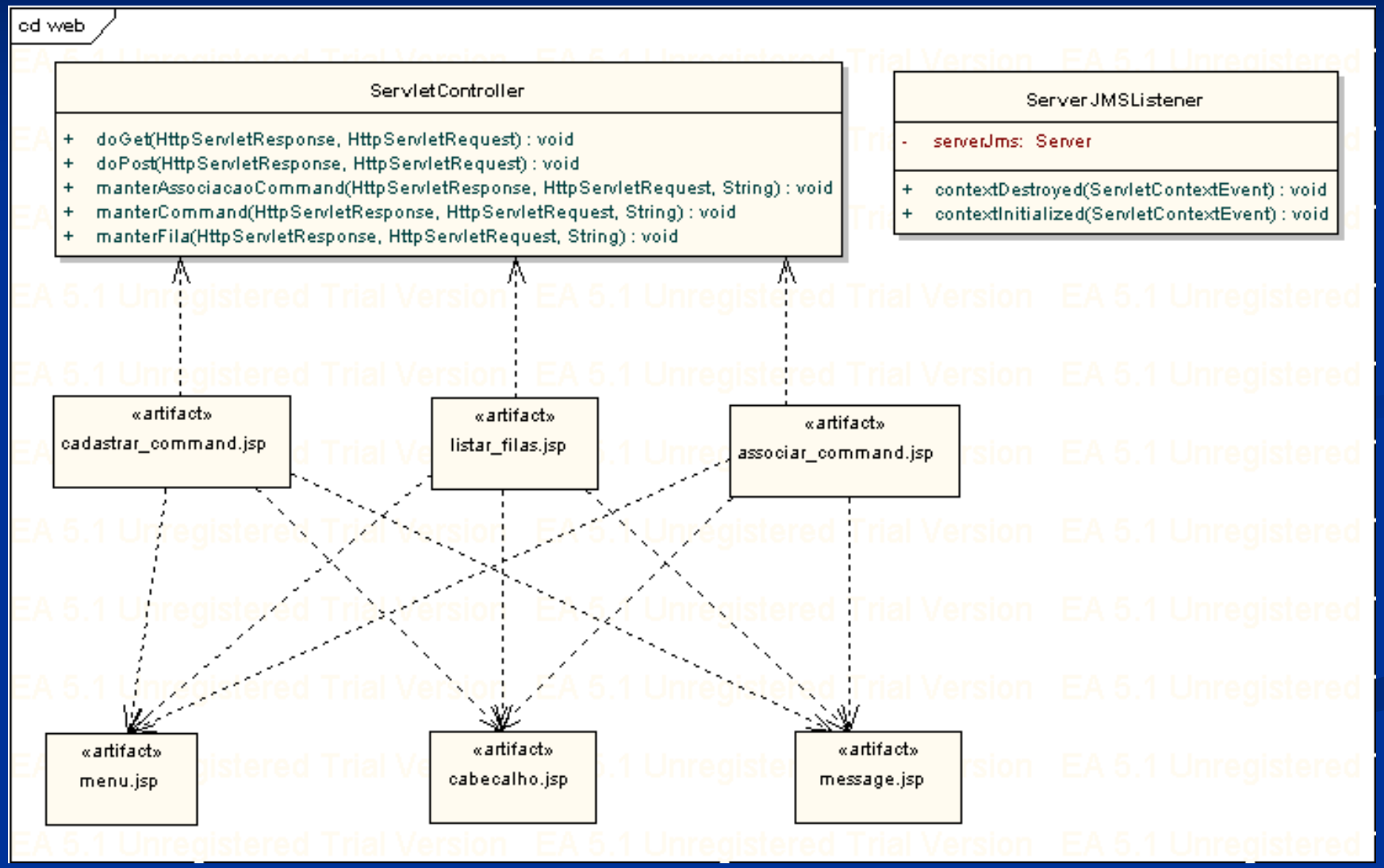


Diagrama de Classes do gerenciador de replicação



Implementação

- Utilizado o padrão DAO de projetos na camada de persistência para garantir o fraco acoplamento com a tecnologia de persistência utilizada;
- Como padrão foram realizadas as interfaces da camada de persistência especificamente para a tecnologia Hibernate;

Implementação

- Utilizado o padrão Singleton de projetos para permitir o compartilhamento de conexões com os providers na camada de replicação, e garantir que exista apenas uma instância de conexão.

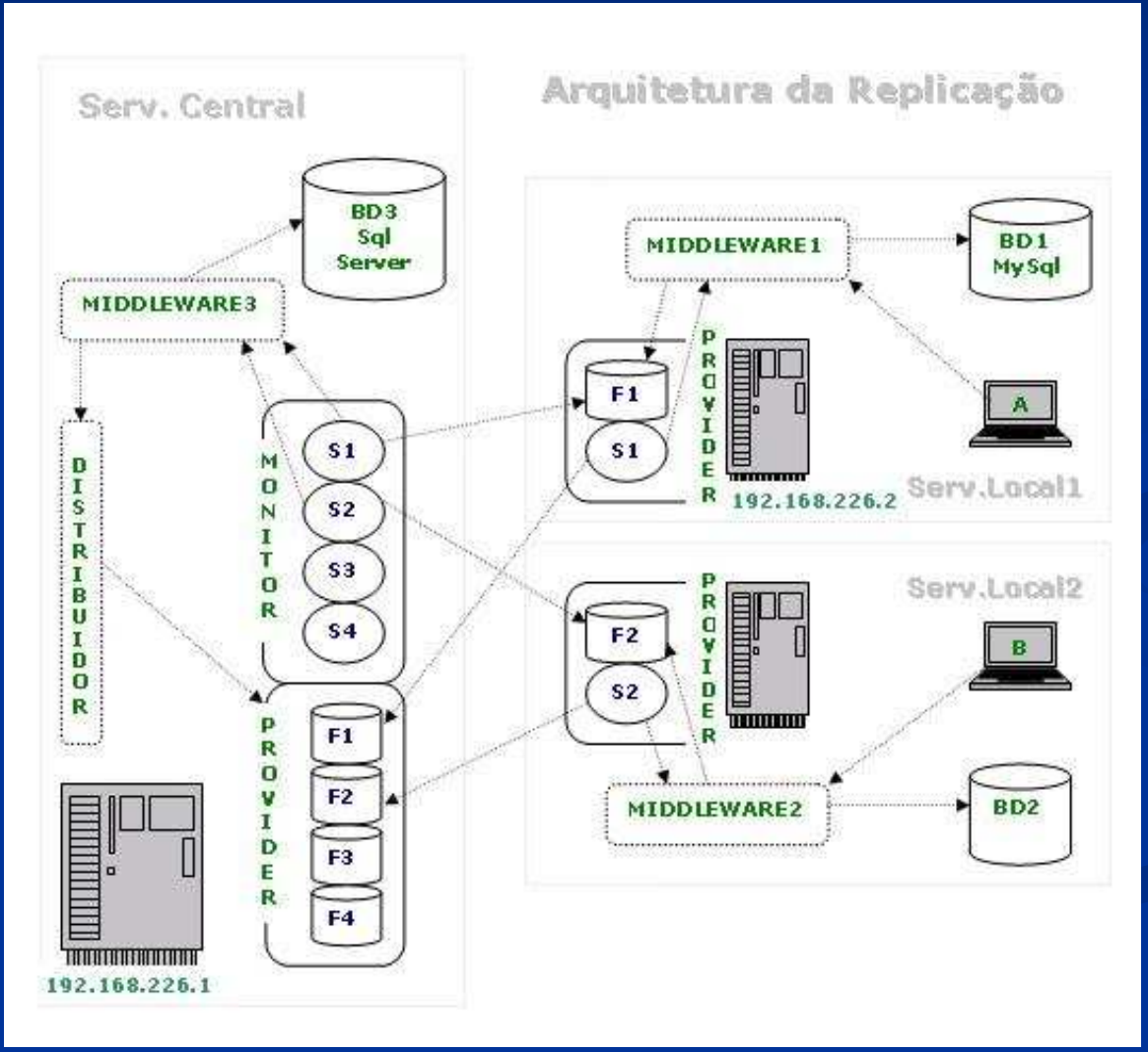
Ferramentas utilizadas

- Eclipse para desenvolvimento do framework;
- Tomcat como container web para o gerenciador de replicação;
- OpenJMS como provider de mensagens;
- API JDOM para leitura dos arquivos de configuração do framework;
- API log4j para customização e geração de logs;

Operacionalidade

- Para demonstrar a operacionalidade, foi criado um simples modelos de dados contendo as tabelas USUARIO e GRUPO_USUARIO.
- A partir do modelo de dados foi desenvolvido uma simples aplicação, que se beneficiou dos recursos do framework para manter este modelo de dados e realizar a replicação para um destino informado.

Exemplo de arquitetura de replicação



Configuração do Framework

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
   <Persistence>
       <Hibernate file="hibernate.cfg.mssql.xml">
       </Hibernate>
       <TransactionManager  class="br.com.replication.dao.
           hibernate.HibernateTransactionManager">
           </TransactionManager>
   </Persistence>
   <Provider local="192.168.226.1" master="192.168.226.1">
        <QueueLocal name=""></QueueLocal>
       <ContextJNDI file="OpenJmsJndi.properties">
       </ContextJNDI>
      <QueryJMSUtil  class="br.com.replication.jms.server.
       QueryJmsUtilImpl"></QueryJMSUtil>
```

Configuração do Framework

```
      <OpenJms derbyHome="C:\openjms-0.7.7-
       alpha-3\db" user="admin" password="openjms">

      </OpenJms>
      <ConnectionFactory name="ConnectionFactory">
      </ConnectionFactory>
  </Provider>
  <ServiceReceiver timeSleep="5000"
     toleranteTimeOut="3000"
  toleranceReconnect="20000"
     qtdMsgsFromIteration="20">

  </ServiceReceiver>
</Configuration>
```

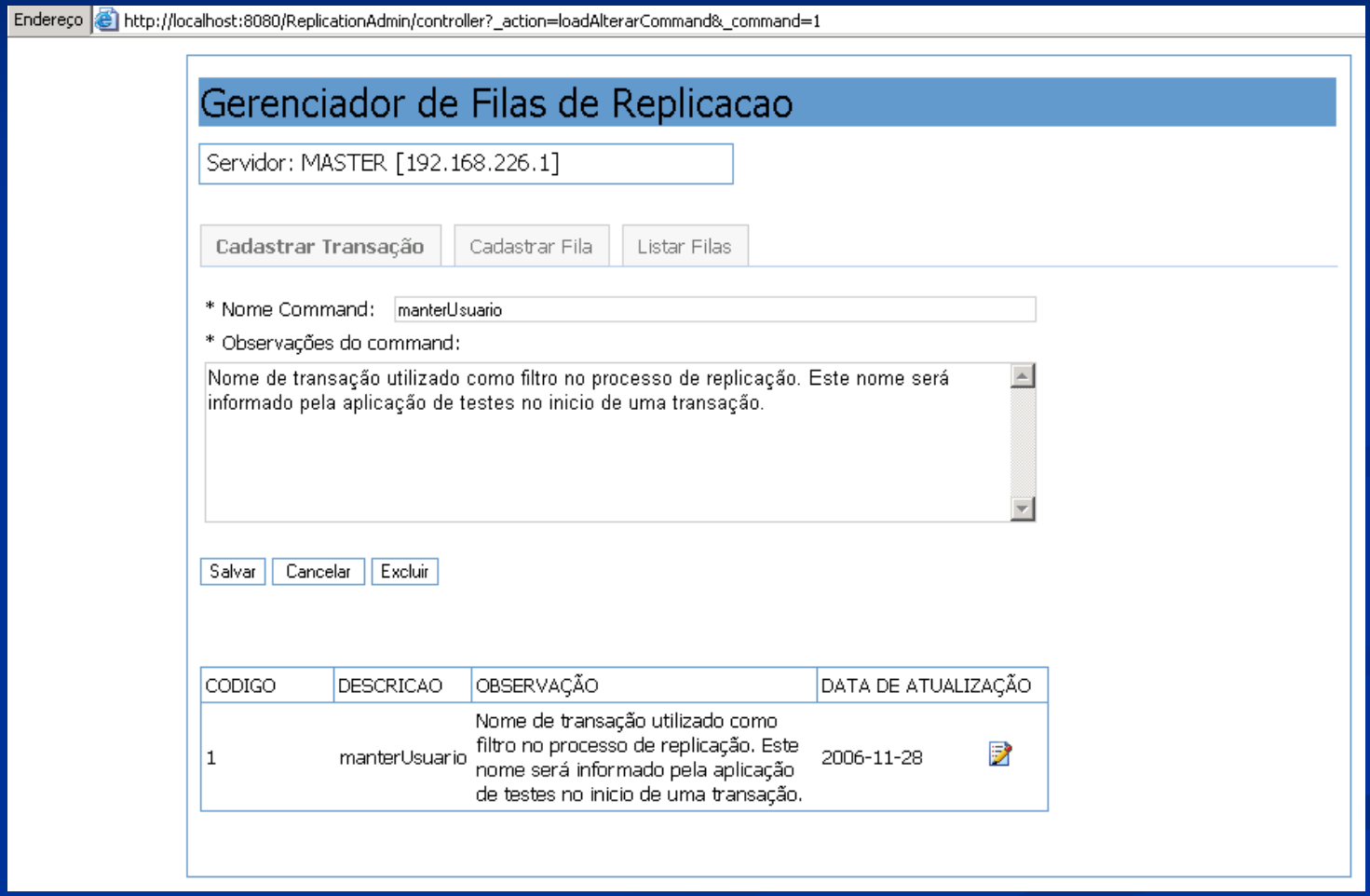
Utilizando o framework

- Carregar configurações contidas em todos os arquivos XML da aplicação:
  `Config.configure();`
- Criar e iniciar configurações da instancia de TransactionManager:
  `TransactionAplication.init();`
- Iniciar uma transação com escopo de replicação:
  `TransactionAplication.beginTransaction("manterUsuario");`
- Finalizar transação e replicar:
  `TransactionAplication.commit();`

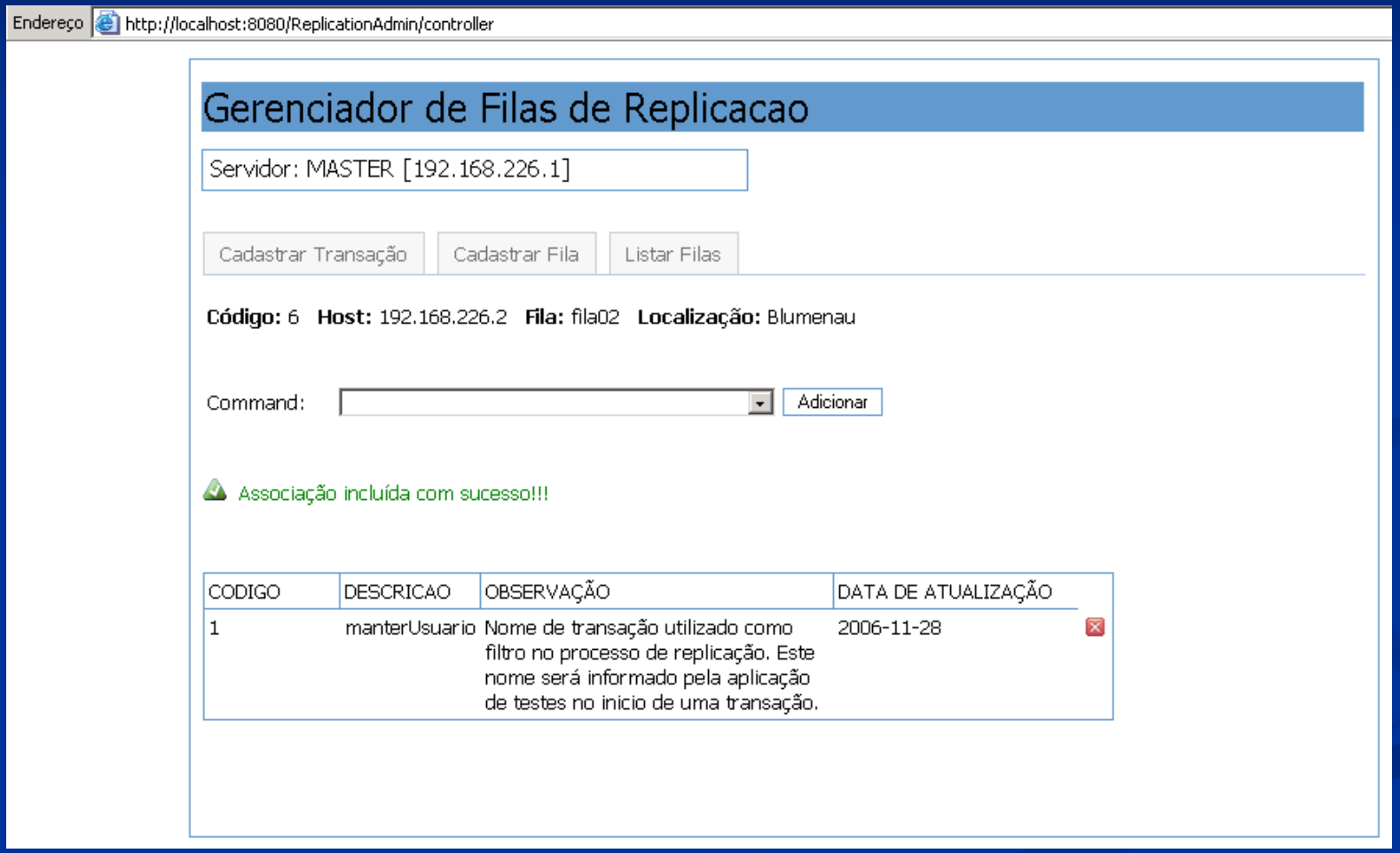
Monitorando filas de mensagens no servidor central

Monitorando fila de mensagem no servidor local

Cadastrando transação



Vinculando transação



Vínculo de transação replicado para provider local

Resultados e Discussões

- Comparativo entre os requisitos de negócio alcançados e não alcançados:
- Latência: tempo entre execução configurável no serviço receptor de mensagem;
- Autonomia: os serviços trabalham de forma autônoma, realizando também a reconexão;
- Disponibilidade física: baixa pois apenas ocorre a replicação quando um registro é atualizado;
- Conflitos de atualização: Não alcançado.

Conclusões

- A heterogeneidade entre as base de dados envolvidas na replicação foi alcançada aderindo a utilização da tecnologia Hibernate.
- Com a utilização de padrões de projetos como DAO e Singleton foi possível fornecer uma estrutura que facilitou a construção de sistemas que necessitam replicar dados.
- O processo de replicação tornou-se configurável e transparente.

Relevância Pessoal

- Aplicação de conhecimento das matérias de Sistemas distribuídos, Engenharia de software e Banco de dados;
- Ter finalizado o desenvolvimento de um projeto, que respeitou as etapas de análise, especificação e implementação;
- Aprofundado conhecimento em programação, utilizando a linguagem Java principalmente nas tecnologias Hibernate e JMS.