4
Um Exemplo de Implementação

Neste capítulo será discutida uma implementação baseada na arquitetura proposta. Para tanto, será explicado como a arquitetura proposta se casa com as necessidades da aplicação InfoPAE descrito na seção 4.1. Em seguida, serão abordadas as tecnologias utilizadas. Logo após, serão apresentados os componentes participantes do ambiente de produção e detalhes de configuração deste ambiente. Finalmente, será feita uma breve passagem dos aspectos a serem levados em consideração na realização dos testes.

4.1. InfoPAE

Baseada na arquitetura proposta por este trabalho, foi desenvolvido um sistema de replicação para o *software* InfoPAE [DIAS, 2003]. O InfoPAE é um sistema desenvolvido pelo laboratório de pesquisa Tecgraf da PUC-Rio em parceria com a Petrobras. Este software visa informatizar os planos de ações de emergências através da caracterização de *cenários* e as respectivas medidas a serem tomadas. Em uma emergência, diversas informações de apoio são associadas de forma a embasar as decisões e enriquecer os relatórios gerados. Estas informações de apoio incluem fotos, documentos, dados georeferenciados e outros.

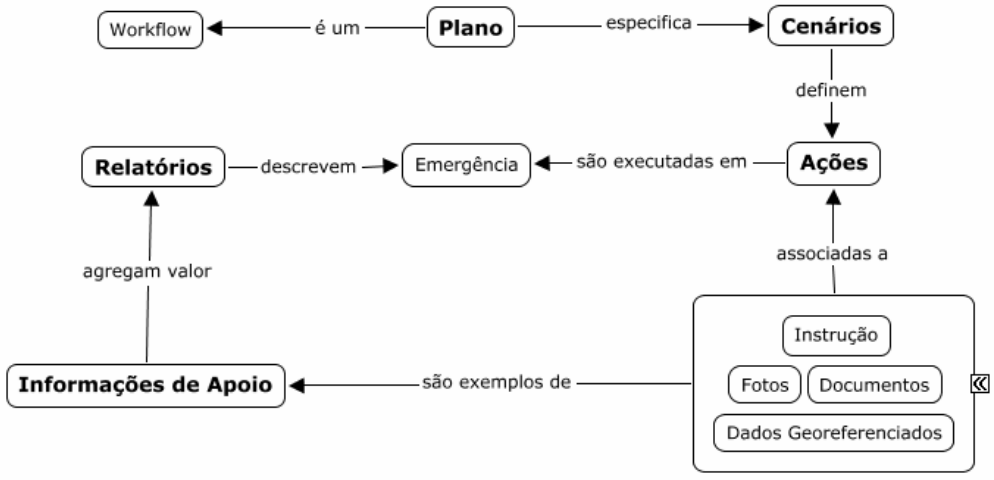

Figura 22. Domínio do InfoPAE

Como este software é executado em situações de emergência, exige-se um alto grau de tolerância à falhas. Nestas condições, faz-se necessário que o sistema esteja disponível durante eventuais falhas de rede. Para promover isto, a estratégia adotada foi a replicação de dados entre pontos críticos com maior probabilidade de queda do serviço de rede.

Entretanto, a necessidade de sincronizar os dados com as informações de apoio sugere a implementação de um serviço de replicação capaz de suportar as regras de negócio envolvidas. Portanto, os dados do InfoPAE caracterizam uma base heterogênea.

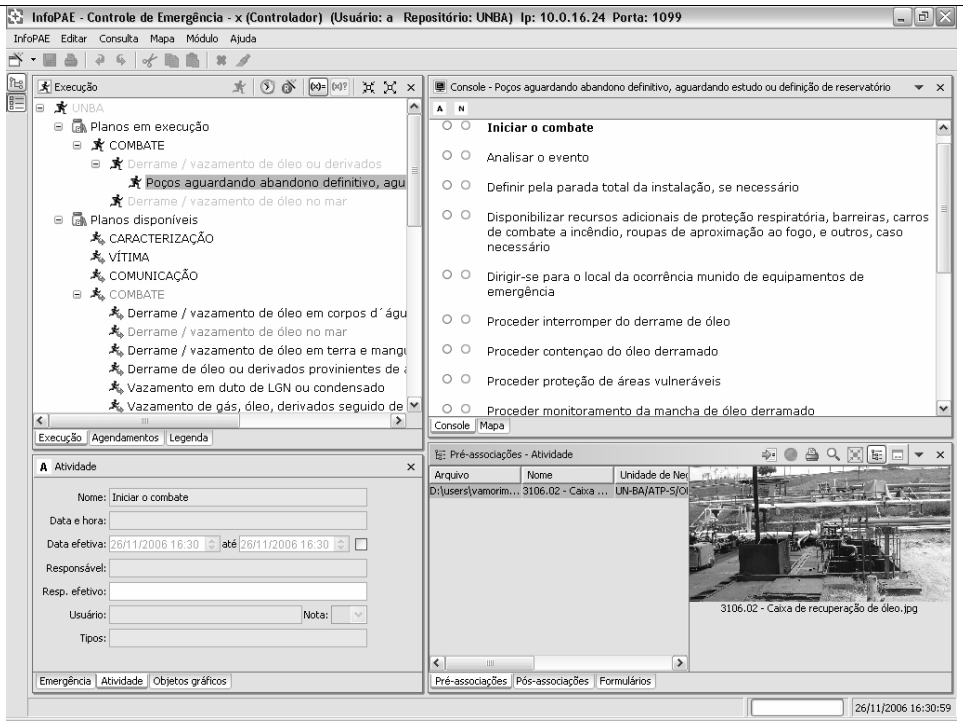

Figura 23. Executando uma emergência no InfoPAE

4.2. RePAE: Uma Implementação de Referência

Para se demonstrar a viabilidade da arquitetura de replicação baseada em agentes, foi desenvolvido o RePAE, uma implementação de referência da arquitetura proposta por este trabalho. O RePAE é responsável por replicar as bases de dados do *software* InfoPAE. O InfoPAE foi escolhido por se tratar de um sistema real que utiliza bases heterogêneas. Os módulos desta base são os repositórios, arquivos em disco e dados geográficos.

Os *repositórios* são como bancos de dados lógicos implementados em um único banco de dados físico. Eles particionam os dados de maneira a facilitar o gerenciamento entre os dados de cada unidade de negócio. Repositórios podem se relacionar através de uma *relação de importação*. Se um respositório **B** importa um repositório **A**, então todos os dados de **A** são visíveis a **B**. O número e o tipo de repositório variam entre os diversos bancos do ambiente InfoPAE.

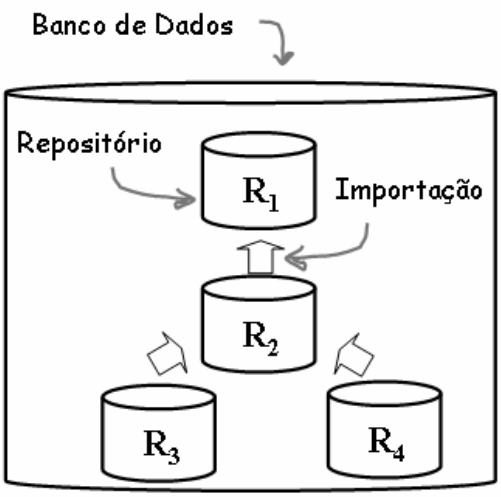

Figura 24. Repositórios

Cada repositório pode referenciar um conjunto de arquivos mantidos externamente ao banco de dados. Estes arquivos incluem documentos, planilhas, fotos, planos e imagens de satélite.

Finalmente, uma base de dados do InfoPAE também possui um conjunto de dados geográficos fundamentais para apoio às regras de negócio de planos de emergências. Estes dados são estruturados com estratégias de otimização de maneira que a aplicação necessita criar diversas tabelas e visões em tempo de execução. Isso torna a tarefa de replicação extremamente complexa com as ferramentas existentes, que sempre consideram um conjunto de tabelas fixas.

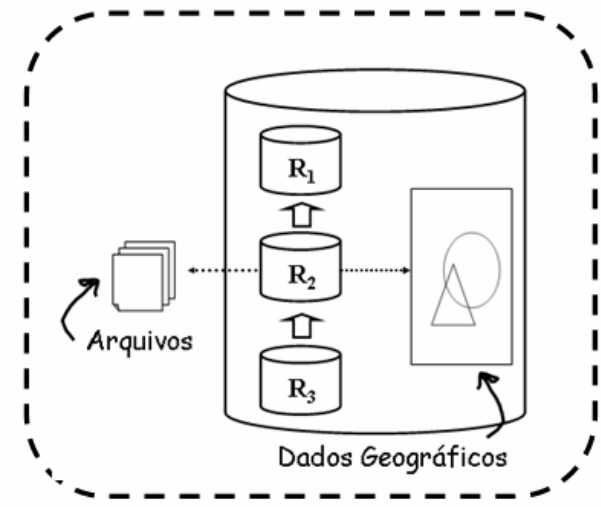

Figura 25. Base de dados do InfoPAE

Assim, o RePAE surge como uma solução para replicar esta base heterogênea, realizando uma replicação que mantêm tanto a consistência transacional como a consistência referencial.

4.3. Tecnologias Adotadas pelo RePAE

Nesta seção será apresentada uma breve descrição das tecnologias e ferramentas envolvidas neste trabalho.

Java [JAVA, 2007] foi a principal linguagem de programação utilizada neste trabalho. O principal critério par a adoção de Java foi a grande quantidade de ferramentas, tecnologias e bibliotecas disponíveis.

A plataforma Java para o ambiente *enterprise* (JEE) possui uma gama de especificações de apoio ao desenvolvimento de software que exigem avançado grau de robustez. Neste trabalho, as tecnologias JDBC, JTA, JMS e JNDI foram amplamente utilizadas.

A API *Java Database Connectivity* (JDBC) padroniza o acesso à fonte de dados como banco de dados e planilhas eletrônicas. Já a especificação *Java Transaction API* (JTA) padroniza um conjunto de interfaces que facilitam a comunicação entre os envolvidos em transações distribuídas: gerente de transações, recursos, servidor de aplicações e as aplicações transacionais. A API *Java Message Service* (JMS), por sua vez, define serviços de mensageria capazes de desacoplar componentes de uma aplicação via comunicação assíncrona e distribuída. Finalmente, a API *Java Naming and Directory Interface* (JNDI) promove um ponto de acesso unificado aos serviços apresentados.

Como implementação para as especificações *JEE* foi adotado o ambiente *WebLogic* [WEBLOGIC, 2007] que está em conformidade com a especificação JEE, o que garante que a migração para outra solução seja factível e de baixo custo. WebLogic foi escolhido por se tratar da solução mais robusta e com maior quantidade de recursos.

Em parceria com o WebLogic, foi utilizado o *WebLogic Scripting Tool* (WLST). O WSLT é uma interface para execução de scripts em linha de comando capazes de automatizar a configuração e facilitar o controle do sistema. O WLST foi escolhido devido a sua facilidade de programação. Sua interface em *Jython* [JYTHON, 2007] permite execução de scripts em lote ou de forma iterativa,

aliando o dinamismo da linguagem *Python* [PILGRIM, 2004] e mantendo o acesso ao código Java existente.

A Figura 26 resume o relacionamento entre as tecnologias apresentadas.

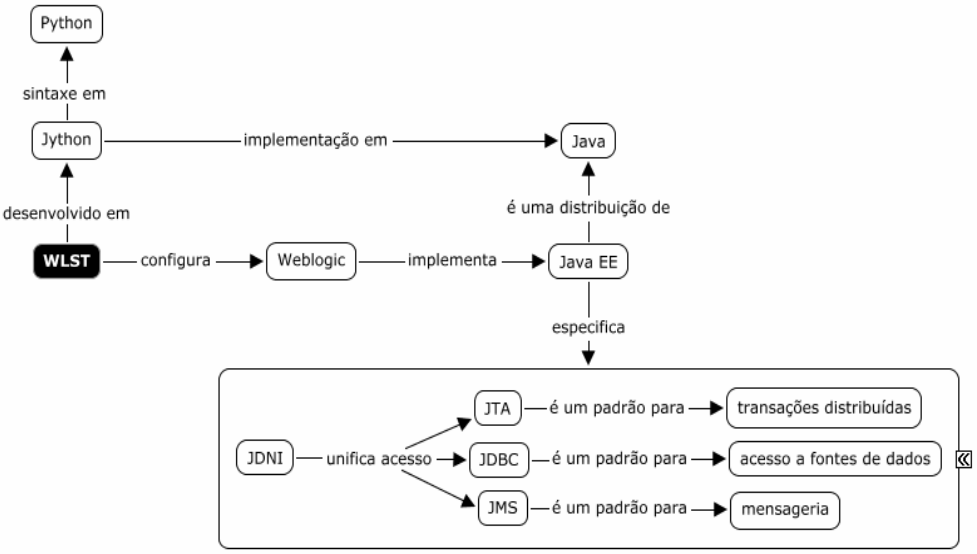

Figura 26. Tecnologias

Java foi utilizada para a implementação do observador, exportador e importador. Também foi utilizada para a implementação do copiador e sincronizador. Já o analisador e o configurador foram implementados utilizando a linguagem Jython, pois utilizam a ferramenta WLST.

As tecnologias JDBC e FTP foram utilizadas para acessar as bases de dados, realizando operações de leitura e escrita. JDBC também constitui a base para a implementação do agente observador. A tecnologia JTA é utilizada pelo exportador, importador e sincronizador para implementar as operações transacionais. Através de JMS são realizadas as trocas de mensagens entre exportador e importador.

Através de JNDI, o exportador, importador e o sincronizador acessam os recursos do servidor JEE que implementam as tecnologias JDBC, JTA e JMS. Estes serviços são criados pelo utilitário configurador, através da ferramenta WLST.

4.4. Componentes Físicos do Ambiente

Um ambiente de replicação do RePAE possui os seguintes componentes físicos:

- **Bancos das bases de dados**: disponibilizadm os módulos de dados repositórios.
- **Servidores de FTP das bases de dados**: disponibilizam módulos de dados que estão no sistema de arquivos.
- **Banco de configuração**: armazenam a topologia do ambiente de replicação
- **Banco de mensagens**: armazenam as mensagens de atualizações a serem aplicadas nas bases escravas.
- **Servidor JEE**: responsável por abrigar os serviços de replicação.

Assim, suponha a seguinte topologia lógica:

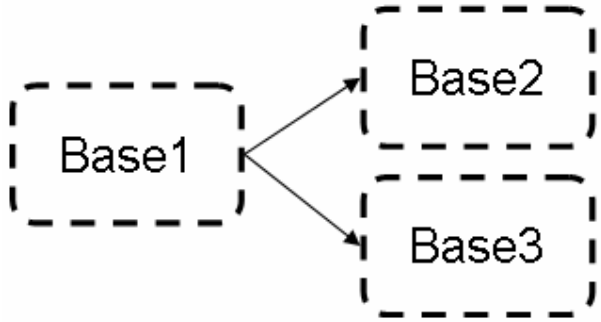

Figura 27. Exemplo de topologia lógica

Nesta topologia, a *base1* é a base mestre e *base2* e *base3* são bases escravas. Seguindo este exemplo, suponha que cada base possua um banco de dados e um conjunto de arquivos associados. Para cada base, será gerado um serviço de replicação que rodará no servidor JEE.

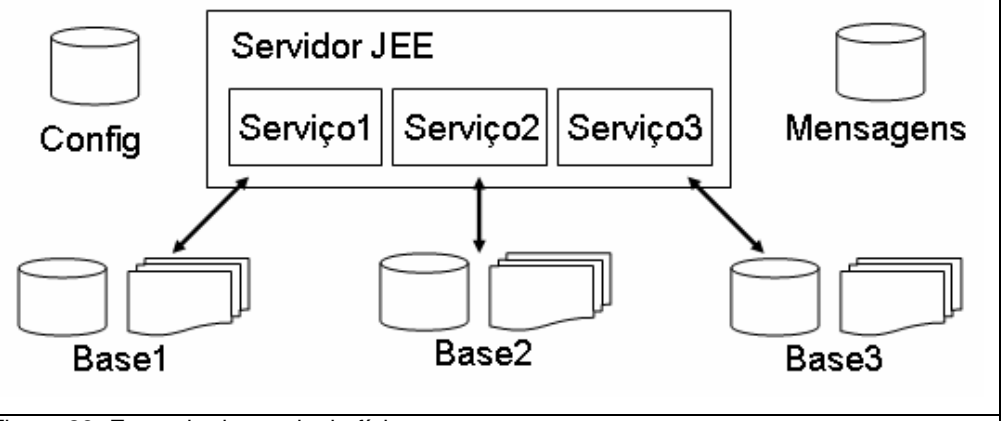

Figura 28. Exemplo de topologia física

Na próxima seção será vista como este ambiente é configurado.

4.5. Instalação e Configuração

Uma distribuição do RePAE é mostrada na Figura 29.

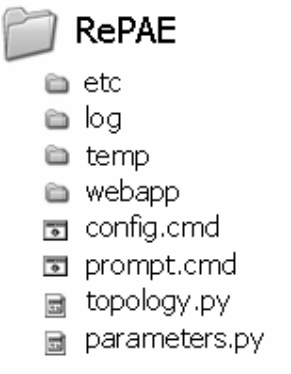

Figura 29. Estrutura de arquivos e diretórios

O diretório *etc* guarda a implementação do configurador e um *template* para um serviço de replicação. Com base neste *template* são gerados os serviços de replicação para cada base de dados. O diretório *log* é responsável por armazenar os arquivos de *logs* dos serviços de replicação. Já no diretório *temp* são armazenados os arquivos temporários enquanto ainda não foram completamente enviados para seus respectivos destinos. O script *config.cmd* chama o configurador e realiza todas as configurações do ambiente baseada nos arquivos de configuração. O script *prompt.cmd* possui a capacidade de realizar ajustes internos, úteis na manutenção e resolução de conflitos. Finalmente, nos arquivos

topology.py e parameters.py são configuradas a topologia e parâmetros de ambiente necessários para o configurador.

Para se montar o ambiente de configuração é necessário configurar os bancos, os servidores de FTP, os parâmetros de ambiente, a topologia e os serviços de replicação. Estas etapas são detalhadas a seguir.

1. **Configurar Bancos**

   Antes de iniciar a replicação, é necessária que as bases mestres e escravas estejam equivalentes. Para garantir isto, o processo mais simples consiste em gerar uma base escrava a partir de uma cópia de uma base de mestre. Esta cópia pode ser realizada através do utilitário copiador, utilizando uma operação de *backup* seguida de uma operação de *restore*.

2. **Configurar os Servidores de FTP**

   Cada base deve possuir um servidor de FTP, com usuário e senha devidamente configurados. Também deve possuir um caminho único para impedir conflitos entre diferentes bases no mesmo ambiente de replicação. Para a base *Base1*, um exemplo de possível caminho único é ftp://maquina1:21/repae_base1.

3. **Configurar Parâmetros de Replicação**

   Um ambiente de replicação necessita da definição de parâmetros globais utilizados por todos os serviços de replicação. Estes parâmetros envolvem configurações do servidor JEE, banco de configurações, banco de mensagens e limites de intervalos utilizados em diversos pontos do serviço de replicação. A figura abaixo ilustra uma possível configuração destes parâmetros.

```
# Nome do domain criado para o RePAE
serverDomain = 'repae'

# Usuario do weblogic para este domain
serverUser = 'user'

# Senha do usuario do servidor para este domain
weblogicPassword = 'd2VibG9naWM+JGk1VDNtQV8xbkYwUF5l'

# Nome do servidor do domain
serverName = 'AdminServer'

# URL de acesso ao servidor
serverUrl = 'https://farol:9002'

# Banco de configuracao da topologia
configHost = 'forte'
configPort = '1695'
configName = 'VAMORIM_REPAE_CONFIG'
configUser = 'infopae'
configPassword = 'aW5mb3BhZV8+JGk1VDNtQV8xbkYwUF5l'

# Banco de armazenamento das mensagens
messagesHost = 'forte'
messagesPort = '1695'
messagesName = 'VAMORIM_REPAE_JMS'
messagesUser = 'infopae'
messagesPassword = 'aW5mb3BhZV8+JGk1VDNtQV8xbkYwUF5l'

# Diretorios para o RePAE
repaeHome  = '/tmp/vamorim/RePAE'
repaeTemp = '/tmp/vamorim/RePAE/temp'
repaeLog = '/tmp/vamorim/RePAE/log'

# Parâmetros de ajustes de performance
waitForRepositoryUpdatesInterval = 60000
waitForRepeatConnectionInterval = 15000
waitForRepeatTransactionInterval = 15000
transactionInitTimeout = 60
transactionMaxTimeout = 90000
```

Figura 30. Exemplo de configuração de parâmetros

Estes parâmetros incluem configurações do ambiente JEE, definição do banco de configurações da topologia, declaração do banco de armazenamento de mensagens e parâmetros para configurações finas de desempenho dos serviços.

4. Configurar Topologia

Uma topologia concentra as informações de quais módulos são replicados, e com qual direção de fluxo. Estas configurações envolvem parâmetros de

acesso a banco de dados e FTP. A figura abaixo ilustra a configuração da topologia replicando um módulo da base1 para a base2 e para a base3.

```
base(id='base1',
     ftp(url='ftp://host1', user='user1', passoword='+kYwUF5lJGDNtQV8xb'),
     database(host='host1', port='1111', name='banco1', user='user1', passoword='mb3BhZV8+JwUF5l'))

base(id='base2',
     ftp(url='ftp://host2', user='user2', passoword='+kYwUF5lJGDNtQV8xb'),
     database(host='host2', port='2222', name='banco2', user='user2', passoword='mb3BhZV8+JwUF5l'))

base(id='base3',
     ftp(url='ftp://host3', user='user3', passoword='+kYwUF5lJGDNtQV8xb'),
     database(host='host3', port='3333', name='banco3', user='user3', passoword='mb3BhZV8+JwUF5l'))

module(id ='module1', master='base1')

flow(module='module1', fromBase='base1', toBase='base2')
flow(module='module1', fromBase='base1', toBase='base3')
```

Figura 31. Configuração de uma topologia

Inicialmente, são definas as bases de dados, indicando o servidor FTP e o Banco de Dados em questão. Em seguida, são declarados os módulos de dados. E finalmente são definidos os fluxos de replicação, indicando qual módulo será replicado de qual base mestre para qual base escrava.

5. Configurar Serviços de Replicação

A configuração dos serviços envolve a configuração do servidor JEE, criação de filas de mensagens, JDBC, configuração do JNDI, configurações para disponibilizar transações distribuídas, geração de aplicações e outros. Entretanto, o utilitário Configurador realiza todas estas etapas através dos parâmetros e topologia definidos. Sendo assim, basta executar o comando *config.cmd*

4.6. Testes

A realização dos testes é dividida em duas etapas: testes horizontais e testes verticais. Os testes horizontais garantem que todas as funcionalidades do software estão mapeadas e são devidamente replicadas. Já os testes verticais garantem que os requisitos da seção 3.2 estão devidamente implementados.

Para a realização dos testes horizontais, diversos casos de teses são automatizados através de um *robô* que simula a interação humana com a aplicação. Um testador grava uma interação com um software e em seguida o robô

pode repetir o mesmo procedimento diversas vezes. As alterações são então replicadas para outra base de dados. Uma nova interação é gravada na base escrava verificando se os dados estão equivalentes com os dados da base mestre.

Já a realização dos testes verticais exige maior participação humana na simulação da geração de falhas de ambiente e na verificação do restabelecimento do software.