Universidade Federal de Ouro Preto

Instituto de Ciências Exatas e Aplicadas

Departamento de Computação e Sistemas

SISTEMAS DISTRIBUÍDOS
Resumo - Andamento do Trabalho Prático

Guilherme Marx Ferreira Tavares 14.1.8006
Ícaro Bicalho Quintão 14.1.8083
Leonardo Sartori de Andrade 15.1.8061

Professora - Carla Rodrigues Figueiredo Lara

João Monlevade
21 de outubro de 2018

Sumário

1 Adequação ao cronograma

Na apresentação da proposta, propusemos um cronograma de trabalho que não pôde ser cumprido. Isso se deu ao fato de que o *Middleware* que iríamos utilizar não serviu ao nosso propósito e após muitas pesquisas, vimos ser inviável o uso de arquitetura *peer-to-peer* nesse trabalho de replicação de dados.

2 Decisões de Projeto

A primeira e mais importante decisão de projeto tomada foi o abandono da arquitetura *peer-to-pier* e uso de arquitetura Cliente-Servidor.

Como o fato de não se ter um servidor central é pré-requisito do projeto, nosso trabalho fica concentrado em duas entidades:

- Banco de IDs: Um mini-servidor, serve para armazenar IDs das operações feitas no Banco de Dados e o nome do respectivo cliente que fez essa operação.
- Cliente: Apesar do nome, esse cliente é ao mesmo tempo Cliente (na relação com o Banco de IDs) e pode ser Cliente ou Servidor na relação com outros Clientes.

Essa comunicação funciona da seguinte forma:

1. Cliente que deseja fazer uma alteração no banco de dados local se comunica com o Banco de IDs para que este atualize sua tabela de IDs colocando o nome deste Cliente no respectivo local de sua alteração (O Banco de IDs não armazenará nenhum script de alteração no banco de dados, somente o ID da operação e o nome do cliente que a fez).
2. Todos os Clientes possuem uma *thread* que monitora essa tabela no Banco de IDs via RMI, assim, no momento em que essa thread detecta uma alteração na tabela, os Clientes pegam o nome de quem fez a alteração através de um método remoto.
3. Uma vez em posse do nome de quem fez a alteração, os Clientes desatualizados se comunicam via outra interface RMI com o Cliente que está atualizado e pegam o script da alteração que foi feita no banco.
4. Ao aplicar o mesmo script em seus respectivos Bancos de Dados Locais, os bancos se mantém atualizados, a replicação foi feita com sucesso.

Outra decisão importante tomada foi com relação a arquitetura *Model-View-Controller*, foi decidido que seria mais organizado manter toda interação de rede dentro do pacote *Model* de modo que este teria, então, as regras de negócio, toda a estrutura de controle do banco de dados e ainda toda a interface de comunicação em Rede.

Dessa forma, fica claro que o pacote *Model* concentra a maior parte do trabalho, deixando o cronograma inicial (com 1 mês de desenvolvimento de cada pacote) obsoleto, uma vez que o *Model* é a parte mais trabalhosa desse projeto.

Dessa forma, por não ter como trazer o pacote *Model* pronto no momento dessa apresentação, decidimos por criar um programa simples de teste dessas interfaces RMI para demonstrar como será seu funcionamento no futuro.

3 O que já foi feito?

1. Repositório do projeto no GitHub: `https://github.com/guilhermemarx14/Concessionaria`
2. Diagrama Entidade-Relacionamento do Banco de Dados;
3. Esquema Relacional (projeto lógico) do Banco de Dados com definição das tabelas;
4. Script de criação do Banco;
5. Interface RMI da rede, que será utilizada tanto na comunicação entre Cliente-Banco de IDs quanto entre Cliente-Cliente;
6. Objetos Model e Objetos DAO.

4 Próximos passos

- Prototipação das telas do projeto e definição das consultas possíveis ao Banco.
- Complementação dos Objetos Model e DAO para suportar todas essas possíveis consultas.
- Desenvolvimento dos pacotes View e Controller.