Andamento do Trabalho Prático

Guilherme Marx, Ícaro Bicalho e Leonardo Sartori

22 de outubro de 2018

UFOP
Universidade Federal
de Ouro Preto

Sumário

Adequação ao cronograma

- Na apresentação anterior, foi proposto o seguinte cronograma:
  1. De 14/09 à 13/10: Desenvolvimento do módulo Model.
  2. De 14/10 à 13/11: Desenvolvimento do módulo View.
  3. De 14/11 à 09/12: Desenvolvimento do módulo Controller.
- O mesmo não pôde ser seguido, isso devido ao fato de que o *Middleware* que iríamos utilizar não serviu ao nosso propósito e, após muitas pesquisas, vimos a inviabilidade de continuar usando arquitetura *Peer-To-Peer* nesse projeto.

UFOP
Universidade Federal de Ouro Preto

Arquitetura de Rede

- Abandono da arquitetura *Peer-To-Peer* e adoção da arquitetura Cliente-Servidor.
- Como a ideia do trabalho é não ter um servidor centralizado, o projeto foi dividido em duas entidades:
  1. *Banco de IDs*: Um mini-servidor, serve para armazenar IDs das operações feitas no Banco de Dados e o nome do respectivo cliente que possui os scripts dessa alteração.
  2. *Cliente*: Apesar do nome, ele é cliente em sua relação com o Banco de IDs e também pode ser cliente ou servidor em sua relação com outros Clientes.
- Comunicação em sua maioria é feita por meio de duas interfaces RMI, uma entre Cliente e Banco de IDs e outra entre Cliente e Cliente.

UFOP
Universidade Federal de Ouro Preto

A comunicação RMI

1. Cliente que deseja fazer uma alteração no banco de dados local se comunica com o Banco de IDs para que este atualize as entradas em sua tabela de IDs colocando o nome deste Cliente no respectivo local da tabela.
2. Todos os clientes possuem uma *Thread* que monitora essa tabela via RMI, assim, no momento em que essa *Thread* detecta uma alteração na tabela, os Clientes pegam o nome de quem realizou essa alteração através de um método remoto.
3. Uma vez em posse do nome de quem fez a alteração, os Clientes desatualizados chamam um método remoto no Cliente atualizado e pegam o *script* da alteração.
4. Ao aplicar o mesmo *script* em seus respectivos Bancos de Dados locais, os bancos se mantém atualizados, a replicação foi feita com sucesso.

UFOP
Universidade Federal de Ouro Preto

A arquitetura MVC

- Para facilitar a organização das responsabilidades, decidimos por atribuir ao pacote *Model* as funções de Coordenar alterações no banco de dados e comunicar via RMI com o Banco de IDs e com os outros Clientes.
- Assim, fica claro que o pacote *Model* concentra a maior parte do trabalho, o que deixa o cronograma inicial (com 1 mês para o desenvolvimento de cada módulo) obsoleto.
- Por esse motivo, não pudemos trazer o pacote *Model* completo para apresentação, como havia sido previsto.

UFOP
Universidade Federal de Ouro Preto

O que já foi feito

1. Diagrama Entidade-Relacionamento do Banco de Dados;
2. Esquema Relacional (projeto lógico) do Banco de Dados;
3. *Script* de criação do Banco de Dados;
4. Interfaces RMI de todas as formas de comunicação do projeto;
5. Objetos Model.

UFOP
Universidade Federal de Ouro Preto

Dúvidas?

Dúvidas, críticas, sugestões?

Repositório GitHub do projeto

https://github.com/guilhermemarx14/Concessionaria