Exame de MAC5853 – Desenvolvimento de Sistemas de Computação

aluno: Marcelo da Silva Reis

banca: Paulo J.S. Silva, Flávio S.C. Silva e Alfredo Goldman

Instituto de Matemática e Estatística
Universidade de São Paulo

Primeiro semestre de 2009

Rotas (L) – Um serviço simples de controle de rotas

Fase 1 – Projeto

Descrição da inicialização dos componentes do sistema

As descrições apresentadas neste documento seguem as arquiteturas apresentadas nos diagramas *ModeloObjetosRotas*, *ModeloObjetosCliente*, *ModeloObjetosCET* (modelos de objetos em notação UML), assim como os modelos ER e físico do banco de dados dos componentes *Rota* e *CET* (diagramas *MER-Rotas*, *MER-CET*, *ModeloFisicoRotas* e *ModeloFisicoCET*).

1 Inicialização do componente *Cliente*

Ao executar o componente *Cliente*, antes de mais nada o sistema inicializa um objeto da classe *LeitorGPS*. Se não consegue inicializar *LeitorGPS*, o sistema fica em *loop* até se conectar com o GPS. Em seguida, o mapa viário é carregado, em um objeto de *MapaViárioCliente*, utilizando para isso um arquivo XML no formato como o do exemplo abaixo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Inicializacao do Modulo "Cliente" -->
<inicializacao_cliente>
        <dados_cliente>
        <cpf>11100011100</cpf>
        </dados_cliente>
        <grafo nome="mapa viario" data_versao="16/02/2009">
                <titulo>Mapa do Sistema Viario de Sao Paulo</titulo>
                <numero_nos>12345</numero_nos>
                <!-- no_inicial e no_final sao o id da via no grafo. -->
                <via>
                        <nome>Rua Fradique Coutinho</nome>
                        <bairro>Pinheiros</bairro>
                        <numero_inicial>300</numero_inicial>
                        <numero_final>450</numero_final>
                        <no_inicial>9786</no_inicial>
                        <no_final>10022</no_final>
                </via>
        </grafo>
</inicializacao_cliente>
```

Onde o elemento *cpf* é a "chave" que *Cliente* utiliza para se comunicar com *Rotas*. As informações dos elementos *via* são armazenadas em objetos da classe *Via*, enquanto que para os entroncamentos são utilizados objetos

da classe *Entroncamento*. Note que os entroncamentos são numeros de 1 até *numero_nos*, bastando dessa forma percorrer todas as arestas para criar em $\Theta(|A|)$ unidades de tempo todos os entroncamentos (A é o conjunto de vias do mapa viário).

Se existir algum erro no formato do arquivo XML de inicialização, *Cliente* escreve uma mensagem de erro na tela e no log do sistema e encerra a execução.

Por fim, o sistema inicializa um objeto da classe *InterfaceCliente*; o sistema exibe a localização atual (obtida de leitorGPS) e aguarda a digitação de um destino válido.

2 Inicialização do componente *Rotas*

A inicialização de *Rotas* segue a sequência de eventos abaixo:

1. *Rotas* inicializa o banco de dados; caso ocorra um erro de inicialização, é gravada no log do sistema uma mensagem de erro e o programa é encerrado.

2. *Rotas* cria um objeto da classe *MapaViário* e chama um método para criar as vias e entroncamentos, a partir do banco de dados. Para isso utiliza-se do seguinte comando SQL na consulta ao banco:

   ```
   SELECT noInicial, noFinal, nomeVia, nomeBairro,
          numeroInicial, numeroFinal
     FROM Via
   ```

   Os entroncamentos são criados a partir das vias.

3. o objeto de *MapaViárioRotas* então inicializa *AtualizadorMapaViário*, chamando o método de solicitação de inicialização.

4. atualizadorMapaViário tenta se conectar ao componente *CET*;

   (a) caso obtenha sucesso, a seguinte mensagem XML é enviada à *CET*:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- Mensagem de inicializacao do Modulo "Rotas" a "CET". -->
   <mensagem_inicializacao />
   ```

(b) caso não obtenha sucesso (*timeout*), uma mensagem de erro é registrada no log e uma nova tentativa é feita (ítem 4).

5. *CET* devolve uma mensagem XML, que pode ser:

   - Resposta esperada, como a do exemplo abaixo:

     ```
     <?xml version="1.0" encoding="UTF-8"?>
     <!-- Mensagem de resposta de "CET" ao pedido de inicializacao. -->
     <mensagem_resposta_inicializacao>
             <via>
                     <no_origem>5678</no_origem>
                     <no_destino>11456</no_destino>
                     <taxa_ocupacao>0.27</taxa_ocupacao>
             </via>
     </mensagem_resposta_inicializacao>
     ```

     Onde a quantidade de elementos do tipo "via" corresponde à todas as rotas cuja taxa de ocupação sejam diferentes de zero.

   - Resposta de erro, caso não tenha recebido corretamente uma de *Rotas*:

     ```
     <?xml version="1.0" encoding="UTF-8"?>
     <!-- Mensagem de resposta de "CET" ao pedido de inicializacao. -->
     <erro_leitura_mensagem />
     ```

6. atualizadorMapaViário processa o arquivo XML recebido:

   (a) se o arquivo foi recebido corretamente e o parsing mostra tratar-se de uma mensagem de erro, retorna-se ao ítem (4.a).

   (b) se o arquivo foi recebido corretamente, é feito parsing do mesmo, atualizando os valores de *taxaOcupação* nos objetos do tipo *Via* e atualizando o horário do atributo *horárioÚltimaAtualização* do objeto mapaViárioRotas.

   (c) caso tenha ocorrido um erro na transmissão do XML, retorna-se ao ítem (4.a).

7. atualizadorMapaViário encerra a conexão com *CET* e é destruído.

8. *Rotas* inicializa um objeto da classe *ServidoraRotas*.

9. *Rotas* inicializa os escalonadores (objetos das classes *EscalonadorMapaViário*, *EscalonadorContas* e *EscalonadorLimpaRotas*).

10. *Rotas* inicializa a interface do funcionário (objeto de *InterfaceFuncionário*.

3 Inicialização do Componente *CET*

1. *CET* inicializa o banco de dados; caso ocorra um erro de inicialização, é gravada no log do sistema uma mensagem de erro e o programa é encerrado.

2. Um objeto da classe *MapaViárioCET* é criado e chama um método para criar as vias e entroncamentos, a partir do banco de dados. Para isso é chamado o seguinte comando SQL:

```
SELECT Vias.noInicial, Vias.noFinal, taxaOcupacao
  FROM Vias, FluxoVia
 WHERE Vias.noInicial = FluxoVias.noInicial
   AND Vias.noFinal = FluxoVias.noFinal
   AND horario >= SELECT DATE_SUB('hora_atual', INTERVAL 15 MINUTE)
   AND horario <  SELECT DATE_ADD('hora_atual', INTERVAL 15 MINUTE)
```

   Os entroncamentos são criados a partir das vias.

3. *CET* inicializa um objeto de *MonitorTráfego*; caso não tenha sucesso na inicialização, o sistema fica em loop até que o monitor de tráfego seja inicializado.

4. O sistema então cria um objeto da classe *EscalonadorMapaViário*, que será responsável por "disparar" as atualizações periódicas dos perfis de trânsito no banco de dados.

5. Por fim, é inicializado um objeto da classe *ServidoraCET*, aguardando assim um contato de *Rotas*.

Referências

[1] G.R. Andrews. Foundations of Multithreaded, Parallel, and Distributed Programming. Addison-Wesley, 2000.

[2] MySQL Homepage. http://www.mysql.com/. *Acesso em 10 de fevereiro de 2009.*

[3] C.M.F. Rubira. Análise Orientada a Objetos. IC-Unicamp, 2000.

[4] A. Silberschatz e H. F. Korth. Sistemas de Bancos de Dados. McGraw-Hill, 1989.

[5] W3C XML Homepage. http://www.w3.org/XML/. *Acesso em 10 de fevereiro de 2009.*