**“FAMaV – Ferramenta de Administração de Máquinas Virtuais: Uma proposta comparativa entre ferramentas para gerenciamento de MVs na Computação em Nuvem”**

**Autores: Yuri Toledo (**[**yuritoledo93@gmail.com**](mailto:yuritoledo93@gmail.com)**)**

**Luis Nakamura (**[**luixjp@gmail.com**](mailto:luixjp@gmail.com)**)**

**Funcionamento FAMaV**

Neste projeto foi proposto o desenvolvimento de uma CLI (*Command Line Interface*) que implementasse os métodos mais comumente utilizados por administradores de máquinas virtuais. Esses métodos são provenientes da biblioteca *libvirt*, que permite a comunicação com diversos hipervisores (*hypervisors*), tais como KVM/QEMU, Xen, MS Hyper-V, IBM PowerVM, etc). Atualmente, existem algumas ferramentas que implementam esses métodos como, por exemplo a ferramenta *virsh*, porém a nova ferramenta proposta neste documento foi desenvolvida buscando adicionar algumas funcionalidades que possam facilitar aos administradores de máquinas virtuais realizar à execução de determinadas operações de uma forma mais eficiente. A ferramenta desenvolvida foi nomeada como: Ferramenta Administrativa de Máquinas Virtuais (FAMaV), e a seguir é feita uma breve descrição de seu funcionamento.

**Ferramentas Utilizadas**

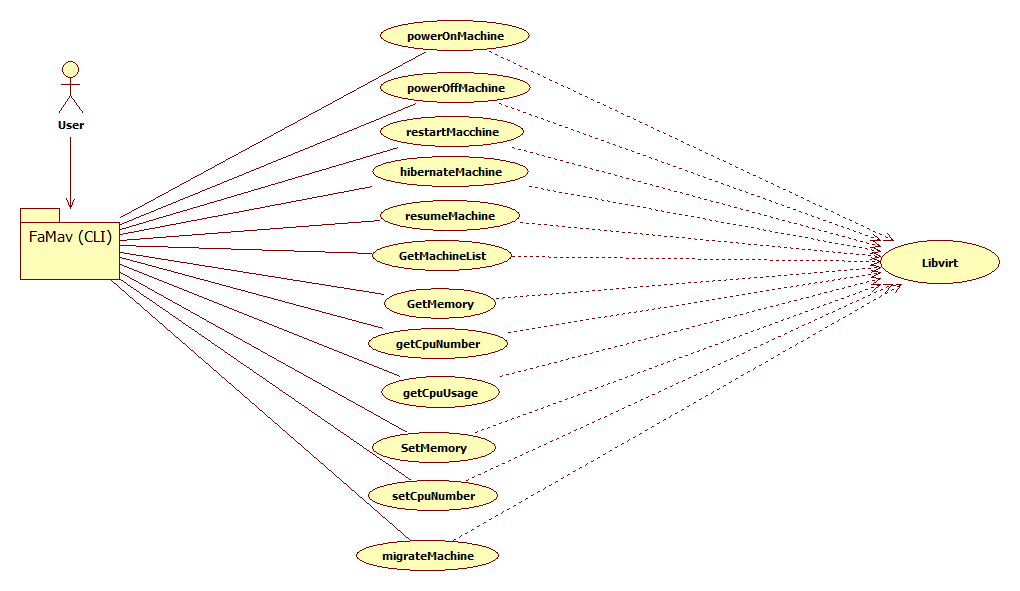
Para o desenvolvimento foi utilizada a biblioteca da API *libvirt*, que é uma API amplamente utilizada para o gerenciamento de máquinas virtuais. A *libvirt* possui bibliotecas para diversas linguagens de programação, como Java, C#, Pearl, Python, entre outras.

Para a FAMaV foi escolhida a utilização da linguagem de programação Java por ser uma linguagem amplamente utilizada e que é conhecida pela sua portabilidade de sistemas já que ela roda sobra a *Java Virtual Machine* e por ter uma ampla documentação sobre a biblioteca *libvirt* para Java.

**Modo de Operação**

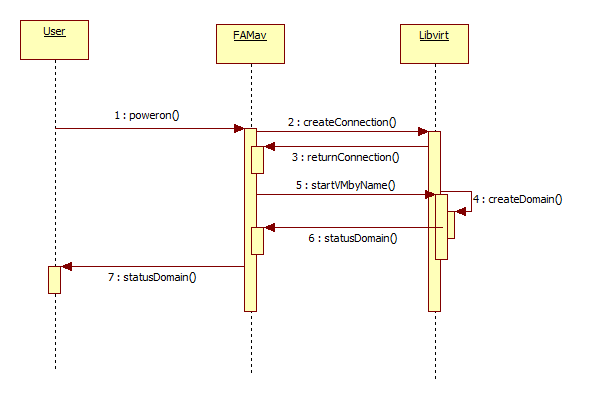
O usuário final, ira interagir com o programa por meio de uma *Command Line* *Interface* (CLI), a FAMaV, que através dos métodos desenvolvidos interagem com a biblioteca *libvirt* e retornam os resultados das operações para a FAMaV. A seguir são apresentados alguns diagramas que exemplificam o funcionamento da ferramenta.

**Diagrama de Casos de Uso**



Esse diagrama mostra o relacionamento do usuário com a FAMaV, que faz as chamadas dos métodos desenvolvidos e utiliza a *libvirt* para as implementações dos mesmos.

**Diagrama de Sequência do método poweron()**



**Sintaxe:**

-poweron <arg[1]> - Onde arg[1] é o nome da máquina virtual a ser ligada.

**Funcionamento:**

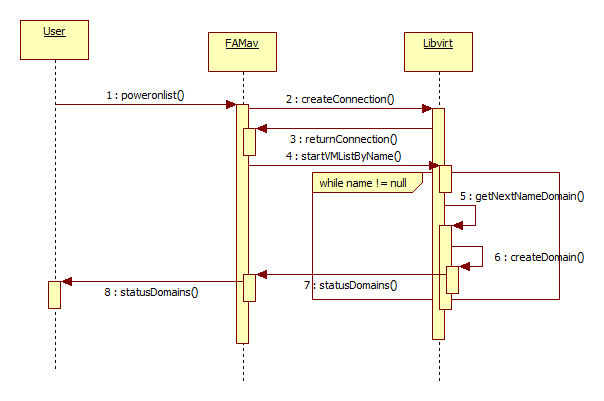
1 – O usuário entra com comando e o nome da máquina virtual a ser ligada através da linha de comando.

2- A FAMaV chama o método *createConnection*() da classe Connection.

3 – Um objeto da classe *PowerOn* é instanciado passando-se como argumento a conexão criada.

4- Na classe *PowerOn* é chamado o método *startVMbyName*(*name*), que utiliza o método *createDomain*() da *libvirt* para ligar a máquina desejada.

5 – É retornado 1 caso a máquina tenha sido ligada com sucesso.

**Diagrama de Sequência do método poweronlist()**

**Sintaxe:**

-poweronlist <arg[1]> <arg[2]> ... <arg[n]> - Onde arg[n] são os nomes das maquinas virtuais a serem ligadas.

**Funcionamento:**

1 – O usuário entra com o comando e a lista das máquinas virtuais a serem ligadas.

2 - A FAMaV chama o método *createConnection*() da classe Connection.

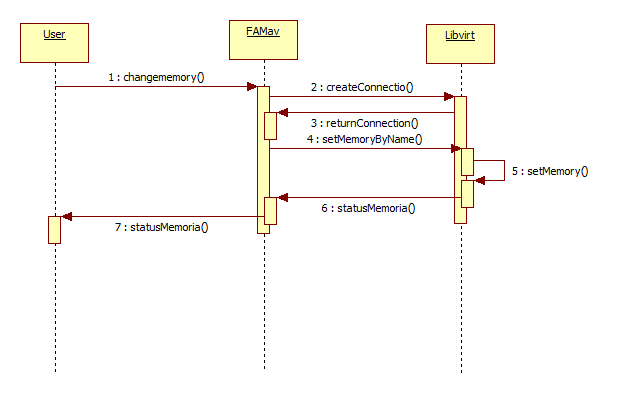
3 – A FAMav cria uma *ArrayList*<*String*> e adiciona o nome das máquinas virtuais à ela.

4 – Um objeto da classe *PowerOn* é instanciado passando-se como argumento a conexão criada.

5- Na classe *PowerOn* é chamado o método *startVMbyList*(*ArrayList*<*String*>), que utiliza o método *createDomain*() da *libvirt* dentro de um *looping* para ligar as máquinas desejadas.

6 – É retornado 1 caso as máquinas tenham sido ligadas com sucesso.

**Diagrama de Sequência do método changememory()**



**Sintaxe:**

-changememory <arg[1]> <arg[2]> - Onde arg[1] é o nome da máquina que terá sua memória modificada e arg[2] é a nova memória da máquina em Kb.

**Funcionamento:**

1 – O usuário entra com o comando e a nova quantidade de memória em kbs.

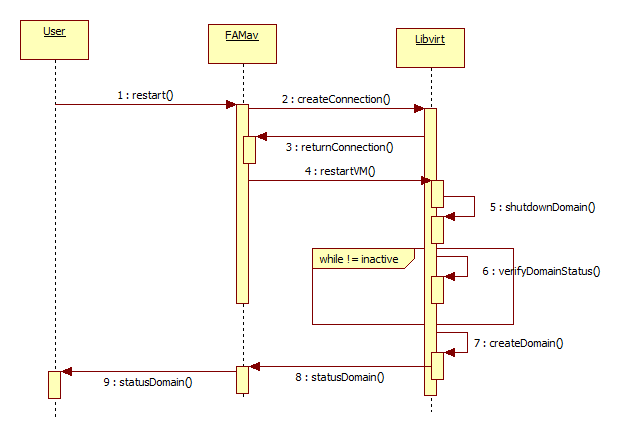
2 - A FAMaV chama o método *createConnection*() da classe Connection.

3 – Um objeto da classe *Memory* é instanciado passando-se como argumento a conexão criada e a nova memória.

4- Na classe *Memory* é chamado o método *setMemorybyName*(*name*), que utiliza o método *setMemory*() da *libvirt* para mudar a memória da máquina.

5 – É retornado 1 caso a máquina tenha trocado a memória com sucesso.

**Diagrama de Sequência do método restart()**



**Sintaxe:**

-restart <arg[1]> - Onde arg[1] é o nome da máquina que será reiniciada.

**Funcionamento:**

1 – O usuário entra com o comando e o nome da máquina que será reiniciada.

2 - A FAMaV chama o método *createConnection*() da classe Connection.

3 – Um objeto da classe *Restart* é instanciado passando-se como argumento a conexão criada.

4- Na classe *Restart* é chamado o método *restartVM*(*name*), que verifica se a máquina estar ligada, caso não é enviada uma mensagem de erro informando que a máquina estar desligada. Caso contrário o método *shutdown*() da *libvirt* manda desligar a máquina. Em um *looping* é verificado enquanto a máquina estiver ligada, quando ela for desligada é utilizado o método *create*() que liga a máquina.

5 – É retornado 1 caso a máquina tenha reiniciada com sucesso.

**Demais métodos**

Os demais métodos possuem funcionamento parecido com os listados a cima. A diferença está apenas nos parâmetros e nome dos métodos, portanto a lista de suas sintaxes são as seguintes:

**Método poweroff()**

-poweroff <arg[1]> - Onde arg[1] é o nome da máquina virtual a ser ligada.

**Método poweroffid()**

-poweroffid <arg[1]> - Onde arg[1] é o id da máquina virtual a ser desligada.

**Método powerofflist()**

-powerofflist <arg[1]> <arg[2]> ... <arg[n]> - Onde arg[n] são os nomes das maquinas virtuais a serem desligadas.

**Método hibernate()**

-hibernate <arg[1]> - Onde arg[1] é o nome da máquina virtual a ser hibernada.

**Método hibernateid()**

-hibernateid <arg[1]> - Onde arg[1] é o id da máquina virtual a ser hibernada.

**Método hibernatelist()**

-hibernatelist <arg[1]> <arg[2]> ... <arg[n]> - Onde arg[n] são os nomes das maquinas virtuais a serem hibernadas.

**Método resume()**

-resume <arg[1]> - Onde arg[1] é o nome da máquina virtual a ser retomada.

**Método restarlist()**

-restart <arg[1]> <arg[2]> ... <arg[n]> - Onde arg[n] são os nomes das maquinas virtuais a serem reiniciadas.

**Método changememoryid()**

-changememoryid <arg[1]> <arg[2]> - Onde arg[1] é o id da máquina virtual a ter sua memória mudada e arg[2] é a nova memória em Kb.

**Método memoryalloc()**

-memoryalloc <arg[1]> <arg[2]> - Onde arg[1] é o nome da máquina virtual a ter sua memória informada em Kb e arg[2] é o devicePath.

**Método infostatus()**

-infostatus <arg[1]> <arg[2]> - Onde arg[1] é o nome da máquina virtual a ter suas informações mostradas e arg[2] é o devicePath.

**Método maxmemory()**

-maxmemory <arg[1]> <arg[2]> - Onde arg[1] é o nome da máquina virtual a ter sua memória máxima informada em Kb e arg[2] é o devicePath.

**Método usecpu()**

-usecpu <arg[1]> <arg[2]> - Onde arg[1] é o nome da máquina virtual a ter seu uso de cpu informado e arg[2] é o devicePath.

**Método nodeinfo()**

-nodeinfo – Mostra informações sobre o nó da máquina real.

**Método diskallocation()**

-diskallocation <arg[1]> <arg[2]> - Onde arg[1] é o nome da máquina virtual a alocação do disco informada e arg[2] é o devicePath.

**Método maxdisk()**

-maxdisk <arg[1]> <arg[2]> - Onde arg[1] é o nome da máquina virtual a ter o máximo possível de disco a ser alocado informado e arg[2] é o devicePath.

**Método listinactives()**

-listinactives – Lista os domínios inativos na máquina.

**Método listactives()**

-listactives – Lista os domínios ativos na máquina.

**Método listall()**

-listall – Lista todos os domínios presentes na máquina, os ligados e os desligados.

**Método listnetworks()**

-listnetworks – Lista as redes utilizadas.

**Método migrate()**

-migrate <arg[1]> <arg[2]> <arg[3]> - Onde arg[1] é o nome da máquina de origem, arg[2] é o uri de destino da máquina a ser migrada e arg[3] é o limite de migração em Mbps.

**Conexões não Locais**

Para se realizar qualquer um dos métodos listado em uma máquina difere é só seguir o seguinte padrão:

-nl <arg[1]> *método a ser executado e seus respectivos parâmetros –* Onde arg[1] é a uri da máquina em que será realizado a operação desejada.