GingaFrEvo & GingaRAP

Evolução do Middleware Ginga para Múltiplas Plataformas (Componentização)
&
Ferramentas para Desenvolvimento e Distribuição de Aplicações Declarativas

Relatório Técnico RT-27

Manual Operacional do Módulo Recommender

PROGRAMA CTIC

REDE NACIONAL DE ENSINO E PESQUISA

Apoio MCT

HISTÓRICO

Data	Versão	Descrição	Autor
17/11/10	0.1	Elaboração do manual de instalação e utilização do módulo.	Tiago Pomponet
08/12/10	0.2	Descrição da introdução, objetivo, escopo e diagramas de classes	Tiago Pomponet
09/12/10	1.0	Revisão final do documento	Erick Melo

ÍNDICE

1 Introdução

Sistemas de recomendação têm sido muito empregados em páginas Web, com relativo sucesso, principalmente em sistemas de comércio eletrônico. Com advento da TVDI e, conseqüentemente, com o aumento da quantidade de serviços é imprescindível que apareçam soluções para auxiliar os usuários na escolha de seus programas prediletos. Nesse contexto, sistemas de recomendação emergem como uma solução possível para atender às necessidades dos usuários.

O objetivo principal do módulo **Recommender** é permitir a aquisição de dados visando conhecer o comportamento do usuário, oferecer recursos de mineração de dados e disponibilizar acesso a essas informações, permitindo aos desenvolvedores de sistemas de recomendação concentrar seus esforços em detalhes de interface e navegabilidade com o telespectador.

O módulo Recommender pode ser definido como a composição de diversos componentes que realizam atividades necessárias para a construção de sistemas de recomendação. Essas atividades vão desde a captura da interação do usuário até o processo de mineração dos dados e recomendação de conteúdo.

1.1 Objetivos

Este documento descreve o manual operacional de uso do Módulo **Recommender**. É descrita a API de uso do componente além da estruturação do código-fonte e instruções para compilação, instalação e utilização.

1.2 Escopo

Neste documento são apresentados os diversos componentes relacionados a sistemas de recomendação. Os componentes referem-se majoritariamente ao projeto **GingaRAP**, especificamente ao projeto **GingaAiyê**.

1.3 Estrutura do Documento

O documento está estruturado da seguinte forma: a Seção 2 apresentado um tutorial de instalação do componente; a Seção 3 apresenta detalhes técnicos do componente: diagrama de classes e descrição da API do componente; a Seção 3 ainda apresenta exemplos de uso do componente.

2 Instalação

2.1 Conteúdo do componente

O Módulo Recommender é composto pelos seguintes componentes:

Componente	Diretório	Descrição
AgentListener	Recommender/AgentListener	Interface para obter operações de interação do usuário.
Database	Recommender/Database	Abstração para um banco de dados sqlite3.
DataMining	Recommender/DataMining	Componente que realiza a mineração de dados utilizando diferentes algoritmos.
EIT	Recommender/EIT	Abstração da tabela de EIT com informações de serviços.
ExportXML	Recommender/ExportXML	Componente que exporta os dados de interação do banco de dados para um formato XML.
LocalAgent	Recommender/LocalAgent	Componente responsável por interpretar e armazenar as interações do usuário.
MiningAlgorithm	Recommender/MiningAlgorithm	Disponibiliza algoritmos para mineração de dados do Recommender.
Scheduler	Recommender/Scheduler	Gerencia o agendamento geral para mineração de dados.
SchedulerItem	Recommender/SchedulerItem	Controla a execução de agendamento de uma aplicação específica.
SDT	Recommender/SDT	Abstração da tabela SDT contendo informações sobre o programa e a emissora.
SystemResource	Recommender/SystemResource	Componente que agrupa funções que permitem interagir com o hardware em que o middleware está sendo executado.
Utils	Recommender/Utils	Este componente agrupa funções gerais que são utilizadas pela maioria dos módulos.

Tabela 1: Localização dos arquivos dos componentes do Recommender.

2.2 Requisitos do Sistema

O Módulo Recommender requer:

Software	URL
gingacc-tuner	http://svn.softwarepublico.gov.br/trac/ginga/browser/gingacc-cpp*
telemidia-system	http://svn.softwarepublico.gov.br/trac/ginga/browser/telemidia-util-cpp*

gingacc-player	http://svn.softwarepublico.gov.br/trac/ginga/browser/gingacc-cpp*
gingacc-wac	http://lince.dc.ufscar.br/svn/ctic2009/wac-cpp/
XML-C++ Parser	http://xerces.apache.org/xerces-c/download.cgi*

Tabela 2: Requisitos do Sistema

* Já disponibilizados na máquina virtual 0.11.2.

2.3 Compilação e Instalação

O módulo Recommender pode ser obtido no seguinte endereço:

URL
http://lince.dc.ufscar.br/svn/ctic2009/recommender-cpp/

O módulo pode ser compilado de dois modos diferentes:

a) Compilação e instalação de todos os componentes

```
cd Recommender
./install.sh
make
sudo make install
```

b) Compilação e instalação independente de cada componente:

Para utilizar este modo é preciso respeitar a interdependência que os componentes possuem. A seguir está explicado o processo utilizando uma hierarquia onde os primeiros componentes apresentados são os que possuem pouca ou nenhuma dependência com os demais, e os últimos possuem alto nível de dependência em relação aos componentes anteriores:

```
cd Recommender/Utils
./autogen.sh
make
sudo make install
```

```
cd Recommender/SystemResource
./autogen.sh
make
sudo make install
```

```
cd Recommender/Database
./autogen.sh
make
sudo make install
```

```
cd Recommender/EIT
./autogen.sh
make
sudo make install
```

```
cd Recommender/SDT
./autogen.sh
make
sudo make install
```

```
cd Recommender/AgentListener
./autogen.sh
make
sudo make install
```

```
cd Recommender/LocalAgent
./autogen.sh
make
sudo make install
```

```
cd Recommender/ExportXML
./autogen.sh
make
sudo make install
```

```
cd Recommender/MiningAlgorithm
./autogen.sh
make
sudo make install
```

```
cd Recommender/DataMining
./autogen.sh
make
sudo make install
```

```
cd Recommender/SchedulerItem
./autogen.sh
make
sudo make install
```

```
cd Recommender/Scheduler
./autogen.sh
make
sudo make install
```

3 Módulo Recommender

Nesta seção será apresentado quais são as funcionalidades de cada componente presente no Recommender.

3.1 AgentListener

3.1.1 Descrição geral

Este módulo é responsável por disponibilizar ao **Recommender** algumas funções que são fornecidas pelo Ginga. Para obter estas ações é necessário utilizar as interfaces do módulo e implantá-las junto a algumas classes do Ginga, como por exemplo, para obter a ação de troca de canal, é necessário utilizar uma interface do AgentListener no módulo gingacc/tuner. As principais funções disponibilizadas por este módulo são: Mudanças de canal, mudanças de volume e mudanças no arquivo NCL de reprodução atual.

3.1.2 Observações

Este módulo não foi totalmente inserido no **Ginga** pois na versão atual de desenvolvimento não foi possível encontrar um serviço de alto nível para tratamento de alterações no áudio, como por exemplo, mudança de volume.

3.1.3 Estrutura dos arquivos

Pasta	Descrição
Recommender/AgentListener	Arquivos de configuração e compilação do módulo
Recommender/AgentListener/src	Arquivos de código fonte
Recommender/AgentListener/include	Arquivos de cabeçalho

3.1.4 Principais operações

Operação	*Descrição*
void volUp()	*Acionado quando o volume é incrementado.*
void volDown()	*Acionado quando o volume é decrementado.*
void channelUp()	*Acionado quando o canal é incrementado.*
void channelDown()	*Acionado quando o canal é decrementado.*
void channelChange(const char* info)	*Acionado quando ocorre uma mudança de canal.*
void nclChange(const char* info)	*Acionado quando ocorre uma interação com uma aplicação NCL.*

3.2 Database

3.2.1 Descrição geral

O módulo **Recommender** utiliza um arquivo de banco de dados do SGBD sqlite3 para armazenamento das informações relevantes que precisam ser guardadas por outros componentes. Desta maneira este módulo é responsável por criar mecanismos que possibilitem a manipulação correta de informações neste arquivo. As operações comuns disponibilizadas por este módulo são: criação do banco de dados, inserção, remoção e busca de informações.

3.2.2 Observações

- Qualquer manipulação no banco de dados criada por outros módulos deve utilizar este módulo como intermediário, garantindo assim a consistência dos dados.
- Este componente utiliza interfaces da classe sqlite3 para garantir o funcionamento entre o banco de dados sqlite e código C/C++. Para obter os arquivos necessários acesse:

"http://www.sqlite.org/capi3ref.html"

3.2.3 Estrutura dos arquivos

Pasta	Descrição
Recommender/Database	Arquivos de configuração e compilação do módulo
Recommender/Database/src	Arquivos de código fonte
Recommender/Database/include	Arquivos de cabeçalho

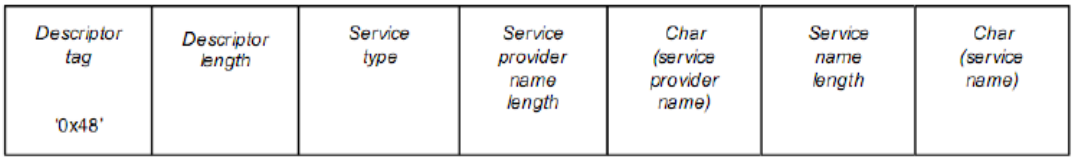
3.2.4 Principais operações

Operação	*Descrição*
bool createDatabase()	*Cria o banco de dados.*
bool closeDatabase()	*Fecha a conexão como o banco de dados.*
bool deleteDatabase()	*Apaga o banco de dados.*
bool query(const char* sql)	*Realiza um comando SQL no banco de dados.*
bool query(const char* sql, vector<string>* head, vector<string>* data)	*Realiza um comando SQL no banco de dados devolvendo nos parâmetros os dados consultados.*

3.3 EIT

3.3.1 Descrição geral

Este módulo fornece uma abstração para a tabela EIT. A tabela EIT é responsável em prover informações sobre os serviços disponibilizados pelos provedores de serviços. Abaixo temos um exemplo dos campos da tabela EIT:



3.3.2 *Observações*

Tendo como referência a versão 0.11.2 do Ginga, não foi possível obter corretamente os dados fornecidos pela tabela EIT.

3.3.3 *Estrutura dos arquivos*

Pasta	Descrição
Recommender/EIT	Arquivos de configuração e compilação do módulo
Recommender/EIT/src	Arquivos de código fonte
Recommender/EIT/include	Arquivos de cabeçalho

3.3.4 *Principais operações*

Operação	*Descrição*
char* getDescriptor_conteudo_genero()	*Recupera a informação com o gênero do serviço.*
char* getDescriptor_conteudo_subgenero()	*Recupera a informação com o subgênero do serviço.*
char* getDescriptor_event_shor_sinopse()	*Recupera informações de sinopse do serviço.*

3.4 SDT

3.4.1 *Descrição geral*

Este módulo cria uma abstração para a tabela SDT responsável em informar o nome do provedor de serviços e o serviço associado. As principais operações deste módulo são: obtenção do nome do provedor de serviço e do serviço associado ao programa sendo executado.

3.4.2 *Observações*

Tendo como referência a versão 0.11.2 do Ginga, não foi possível obter corretamente os dados fornecidos pela tabela SDT.

3.4.3 *Estrutura dos arquivos*

Pasta	Descrição
Recommender/SDT	Arquivos de configuração e compilação do módulo
Recommender/SDT/src	Arquivos de código fonte
Recommender/SDT/include	Arquivos de cabeçalho

3.4.4 *Principais operações*

Operação	*Descrição*
char* getServiceName()	*Recupera o nome do serviço: o programa*
char* getServideProvider()	*Recupera o nome do provedor de serviços: a emissora*

3.5 LocalAgent

3.5.1 *Descrição geral*

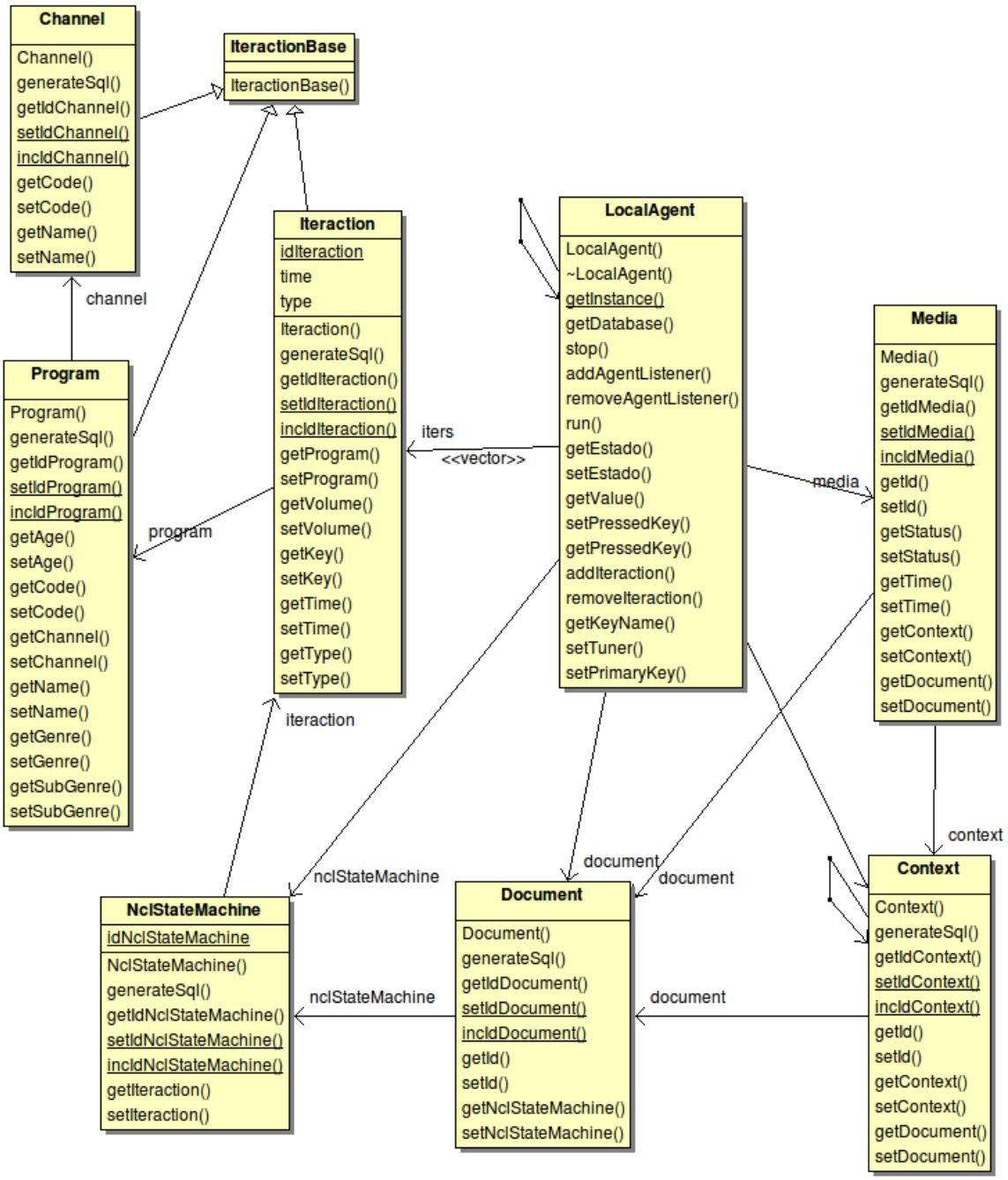
Este módulo é responsável por interpretar os comandos dados pelo usuário (mudança de canal e volume e teclas de interação), obtidos por meio da interface AgentListener, e armazenar esses dados de tal maneira que o módulo de mineração (DataMining) possa extrair informações para uso na identificação de padrões de interação por parte do usuário. Este módulo é acoplado ao Ginga, isto é, sempre que o Ginga é executado o LocalAgent também entrará em ação na coleta de comandos do usuário, por causa disso a classe LocalAgent é um singleton, podendo existir somente uma única instância desta classe. Este módulo utiliza a Classe Database para persistir os dados relevantes no banco de dados "interaction.db" presente na pasta de arquivos de configuração do Ginga:

"/usr/local/etc/ginga/files/recommender/interaction.db"

3.5.2 *Observações*

- Tendo como referência a versão 0.11.2 do Ginga, não foi possível obter corretamente os dados fornecidos pela tabela SDT e EIT.
- Como diversas funções que este módulo utiliza para a coleta de dados ainda não foi corretamente implementada na versão de desenvolvimento 0.11.2 do Ginga, a obtenção de alguns dados não passa pelo AgentListener, isto é, alguns dados de interação são obtidos diretamente dos players do Ginga (em um nível mais baixo), enquanto outros não puderam ser obtidos de forma alguma.

- Este componente possui diversas classes que permitem garantir a consistência dos dados salvos no banco de dados. A estrutura destas classes é descrita pelo diagrama de classes abaixo:



3.5.3 Estrutura dos arquivos

Pasta	Descrição
Recommender/LocalAgent	Arquivos de configuração e compilação do módulo
Recommender/LocalAgent/src	Arquivos de código fonte
Recommender/LocalAgent/files	Arquivos de configuração: banco de dados para exportação após instalação do módulo.
Recommender/LocalAgent/include	Arquivos de cabeçalho

3.5.4 Principais operações

Operação	*Descrição*
static LocalAgent* getInstance()	*Recupera a instância da classe LocalAgent.*
void run()	*Ativa a execução em thread do LocalAgent. A partir desta chamada o componente irá monitorar todas as interações realizadas pelo usuário.*
bool stop()	*Desativa o monitoramento de interações do usuário.*
bool addAgentListener(AgentListener* listener)	*Adiciona no vetor de "listeners" um monitorador do tipo AgentListener para registrar as interações do usuário.*
bool removeAgentListener(AgentListener* listener)	*Remove do vetor de "listeners" um monitorador específico.*

3.6 ExportXML

3.6.1 Descrição geral

Componente que exporta os dados de interação do banco de dados para um formato XML. Os arquivos XML salvos por este módulo são salvos na pasta de configurações do Ginga: "/usr/local/etc/ginga/files/recommender".

3.6.2 Observações

- Este componente utiliza a biblioteca Xerces-C++ XML Parser para fornecer suporte a armazenamento de informações utilizando arquivos XML.
- Este módulo é executado independentemente do módulo Recommender e do Ginga.

3.6.3 Estrutura dos arquivos

Pasta	Descrição
Recommender/ExportXML	Arquivos de configuração e compilação do módulo

Recommender/ExportXML/src	Arquivos de código fonte
Recommender/ExportXML/include	Arquivos de cabeçalho

3.6.4 Principais operações

Operação	*Descrição*
int exportXML(string file)	*Exporta os dados para o arquivo XML indicado como parâmetro.*

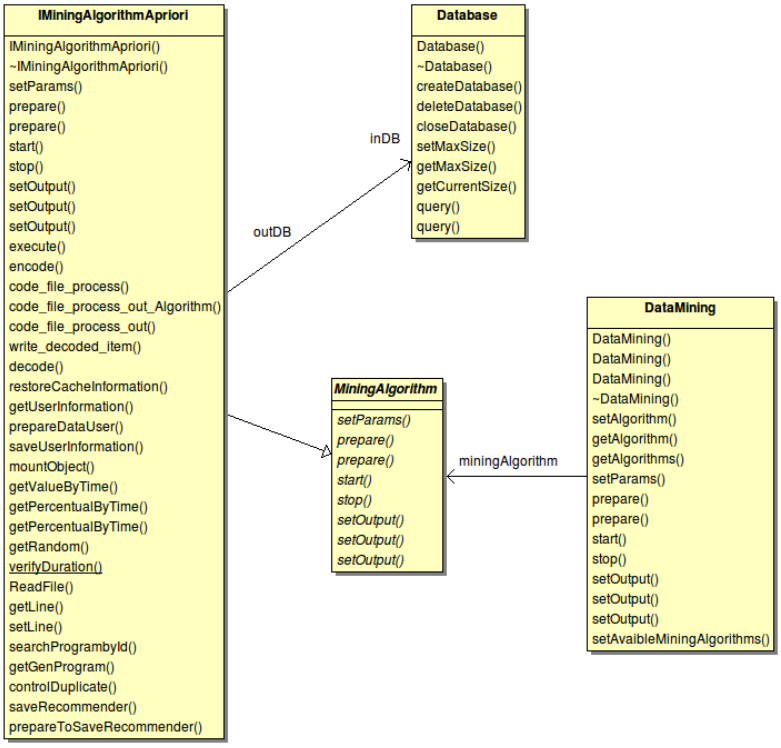
3.7 MiningAlgorithm

3.7.1 Descrição geral

Este componente é responsável por gerenciar o funcionamento dos algoritmos disponíveis para a mineração de dados. Nesta versão do módulo **Recommender** foi implementado somente um algoritmo para a mineração de dados: o algoritmo Apriori, que tem como base encontrar regras de associação sobre um conjunto de dados para determinar os itens que aparecem com mais freqüência nas interações executadas pelo usuário.

3.7.2 Observações

- Os arquivos referentes a implementação do algoritmo Apriori foram obtidas no site: **http://www.cs.bme.hu/~bodon/en/index.html.**
- Este componente é integrado com outros componentes do módulo Recommender seguindo a especificação do diagrama de classes abaixo:



3.7.3 Estrutura dos arquivos

Pasta	Descrição
Recommender/MiningAlgorithm	Arquivos de configuração e compilação do módulo
Recommender/MiningAlgorithm/src	Arquivos de código fonte
Recommender/MiningAlgorithm/apriori23/src	Arquivos de código fonte da biblioteca apriori
Recommender/MiningAlgorithm/include	Arquivos de cabeçalho
Recommender/MiningAlgorithm/apriori23/include	Arquivos de cabeçalho da biblioteca apriori

3.7.4 Principais Operações

Operação	*Descrição*
bool setParams(vector<char*> params)	*Ajusta os parâmetros necessários para a execução do algoritmo.*
bool prepare()	*Prepara os dados que serão processados.*
void execute()	*Executa o algoritmo Apriori.*
vector<string> searchProgramById()	*Busca na tabela EIT os programas recomendados.*
void prepareToSaveRecommender()	*Prepara o banco para salvar as recomendações.*
void saveRecommender()	*Salva a recomendação no arquivo Lua.*

3.8 DataMining

3.8.1 Descrição Geral

Componente que realiza a mineração de dados utilizando diferentes algoritmos. Este componente também realiza manipulação sobre os dados para adequar ao formato específico de mineração de cada algoritmo.

3.8.2 Estrutura dos arquivos

Pasta	Descrição
Recommender/MiningAlgorithm	Arquivos de configuração e compilação do módulo
Recommender/MiningAlgorithm/src	Arquivos de código fonte
Recommender/MiningAlgorithm/include	Arquivos de cabeçalho

3.8.3 Principais operações

Operação	*Descrição*
bool setParams(vector<char*> params)	*Ajusta os parâmetros necessários para a execução do algoritmo.*
bool setAlgorithm(const char* algorithm)	*Define o algoritmo que será utilizado no processamento dos dados.*
const char* getAlgorithm()	*Retorna o algoritmo que será utilizado no processamento dos dados.*
vector<char*>* getAlgorithms()	*Retorna todos os algoritmos disponíveis para mineração de dados.*

3.9 Scheduler

3.9.1 Descrição geral

Como o processo de mineração de dados é dependente da quantidade de dados assim como do algoritmo utilizado, foi necessário criar um módulo para gerenciar o agendamento da mineração de dados. Este componente utiliza o FCron (http://fcron.free.fr/) para agendamento dos itens de mineração.

3.9.2 Estrutura dos arquivos

Pasta	Descrição
Recommender/Scheduler	Arquivos de configuração e compilação do módulo
Recommender/Scheduler/src	Arquivos de código fonte
Recommender/Scheduler/include	Arquivos de cabeçalho

3.9.3 Principais operações

Operação	*Descrição*
bool schedule()	*Agenda a execução de uma aplicação.*
bool load(const char* file)	*Carrega para a memória informações de agendamento armazenadas em arquivo.*
bool store(const char* file)	*Salva em arquivo informações de agendamento da memória.*

3.10 SchedulerItem

3.10.1 Descrição geral

Controla a execução de uma aplicação específica que foi previamente agendada. A principal função deste componente é gerenciar a execução de uma aplicação: inicialização, início da execução, fim da execução e parada na execução.

3.10.2 *Estrutura dos arquivos*

Pasta	Descrição
Recommender/SchedulerItem	Arquivos de configuração e compilação do módulo
Recommender/SchedulerItem/src	Arquivos de código fonte
Recommender/SchedulerItem/include	Arquivos de cabeçalho

3.10.3 *Principais operações*

Operação	*Descrição*
bool init(vector<char*>* params)	*Inicializa a aplicação com os parâmetros especificados.*
bool start()	*Coloca a aplicação em execução.*
bool stop()	*Para a execução da aplicação.*
bool pause()	*Coloca a aplicação em modo pause.*

3.11 SystemResources

3.11.1 *Descrição geral*

Componente que agrupa funções que permitem interagir com o hardware em que o middleware está sendo executado. Disponibiliza ao módulos as principais informações sobre o hardware: memória, disco e giro de CPU. Também permite reservar recursos de CPU para serem utilizados durante a mineração de dados.

3.11.2 *Estrutura dos arquivos*

Pasta	Descrição
Recommender/SystemResources	Arquivos de configuração e compilação do módulo
Recommender/SystemResources/src	Arquivos de código fonte
Recommender/SystemResources/include	Arquivos de cabeçalho

3.11.3 *Principais operações*

Operação	*Descrição*
long getCOUClock()	*Retorna a frequência da CPU.*
int getCPUCount()	*Retorna a quantidade de CPUs da máquina.*
long getMemorySize()	*Retorna a quantidade total de memória.*
long getMemoryFree()	*Retorna a quantidade de memória livre.*

off_t getDiskSize()	*Retorna a quantidade de espaço de armazenamento.*
off_t getDiskFree()	*Retorna a quantidade de espaço de armazenamento livre.*
int reserve()	*Reserva um recurso do sistema para uma aplicação.*
bool freeReserve()	*Desfaz uma reserva de recurso do sistema.*
bool allocate()	*Faz uso de um recurso do sistema reservado anteriormente.*
bool freeAllocation()	*Libera o uso de um recurso do sistema.*

3.12 Utils

3.12.1 Descrição geral

Este componente agrupa funções gerais que são utilizadas pela maioria dos módulos. Entre as funções existem: tratamento de informações de tempo do sistema, conversão entre tipos e separação de cadeias de caracteres em tokens.

3.12.2 Observações

- *Todas as operações deste componente são do tipo "static" pois precisam ser acessadas diretamente pelo escopo da classe, isto é, sem necessitar da instanciação de um objeto da classe.*

3.12.3 Estrutura dos arquivos

Pasta	Descrição
Recommender/Utils	Arquivos de configuração e compilação do módulo
Recommender/Utils/src	Arquivos de código fonte
Recommender/Utils/include	Arquivos de cabeçalho

3.12.4 Principais operações

Operação	*Descrição*
void Tokenize()	*Separa uma cadeia de caracteres em diversos strings (tokens). Utiliza um delimitador para controle de tokens.*
string convertIntToString()	*Converte um tipo inteiro para tipo std::string.*
int convertStringToInt()	*Converte um tipo std::string para o tipo inteiro.*
int getDayNow()	*Retorna o dia atual.*
int getMonthNow()	*Retorna o mês atual.*
int getYearNow()	*Retorna o ano atual.*

4 Utilização do Recommender

4.1 Utilização para desenvolvimento

Para este tipo de utilização não é preciso instalar o módulo completamente, ou seja, somente os componentes necessários ao desenvolvimento precisam de instalação. Para todos os componentes, após sua compilação, são gerados arquivos de biblioteca dinâmica ".so", portanto para utilização dos componentes basta "linkar" a biblioteca na compilação do novo projeto e também realizar os "includes" para os arquivos de cabeçalho necessários para o uso da biblioteca dos componentes.

Arquivos das bibliotecas dinâmicas: "/usr/local/lib/ginga/recommender"

Arquivos de cabeçalhos : "/usr/local/include/ginga/recommender"

4.2 Utilização para testes de aplicação

Este tipo de utilização exige que o módulo seja instalado por completo.

4.2.1 Aplicação 1: Monitoramento das interações do usuário

Quem realiza o modo monitoramento no Recommender é o componente LocalAgent que é integrado junto à execução do ginga. Portanto, para que o componente comece a catalogar as informações de interação do usuário basta executar normalmente o Ginga, tanto para arquivos locais quanto para arquivos fornecidos por um stream.

```
ginga
```

ou

```
ginga --ncl arquivo_teste.ncl
```

Os dados serão armazenados no banco de dados interaction.db na pasta de configurações do Ginga: "/usr/loca/etc/ginga/files/recommender".

4.2.2 Aplicação 2: Armazenar dos de interações em um arquivo XML completo

Para realizar este tipo de operação é necessário executar o Ginga utilizando como parâmetro a tag "--xml " que informa ao programa que a aplicação deverá somente carregar os módulos para acesso ao banco de dados e também de conversão dos dados para arquivos XML. Após a exportação dos dados o Ginga será finalizado automaticamente.

```
ginga --xml arquivo_teste.xml
```

O arquivo XML final será armazenado na pasta de configurações do Ginga: “/usr/local/etc/ginga/files/recommender”.

4.2.3 Aplicação 3: Gerar arquivos de recomendação em formato Lua (teste)

Pelo fato da versão atual de desenvolvimento do Ginga não possuir diversas funções para o funcionamento adequado do componente de obtenção de informações sobre serviços e programas: tabelas SDT e EIT, não foi possível gerar uma aplicação genérica que possibilite testar quaisquer informações catalogadas pelo módulo LocalAgent. Desta maneira, a aplicação vista a seguir é baseada em um teste que utiliza uma entrada padrão de dados sobre determinadas interações de um usuário. Também é utilizada uma tabela de programas (abstração da EIT) específica para este caso.

Dentro da pasta Recommender/DataMining/Test é possível encontrar um caso de teste que permite visualizar exemplos de recomendação. Para utilizar este teste faça:

1) Compile o programa de teste main.cpp

   Basta estar na pasta Recommender/DataMining/Test e digitar “make” e em seguida “sudo make install” no Shell de comandos.

2) Execute o programa teste gerado:

   Basta estar na pasta Recommender/DataMining/Test e digitar ”./test”.

3) Abra os arquivos de recomendação “.lua” presentes na pasta.

Como este teste é baseado em uma entrada única, todas as execuções desta aplicação irão gerar o mesmo resultado (itens de recomendação iguais).

5 Responsáveis pelo Documento

Nome: Cesar Augusto Camillo Teixeira

E-mail: cesar@dc.ufscar.br

Telefone: (0xx16) 3351-8614

Instituição: Laboratório para Inovação em Computação e Engenharia da Universidade Federal de São Carlos (Lince-UFSCar)

Nome: Erick Lazaro Melo

E-mail: erick_melo@dc.ufscar.br

Telefone: (0xx16) 3351-8614

Instituição: Laboratório para Inovação em Computação e Engenharia da Universidade Federal de São Carlos (Lince-UFSCar)

6 Apêndice 1 – Alterações nos componentes do Ginga

Este apêndice descreve como alterar alguns componentes do Ginga para que ofereçam as funcionalidades necessárias para o módulo Recommender.

6.1 Ginga-cpp

- No arquivo main.cpp é necessário incluir o cabeçalho do componente LocalAgent, para isso inclua as linhas no cabeçalho:

```
#include "recommender/LocalAgent.h"
using namespace ::br::ufscar::lince::ginga::recommender;
```

- No arquivo main.cpp é necessário criar a instância do módulo LocalAgent, para isso inclua a linha abaixo após a chamada do método startPresentation:

```
LocalAgent::getInstance();
```

- No arquivo main.cpp é necessário habilitar a tag para chamar o ginga no modo de exportação de arquivo XML, para isso declare a variável abaixo dentro da função main:

```
bool isXML = false;
```

e inclua as linhas abaixo dentro do comando for que verifica:

```
else if ((strcmp(argv[i], "--xml") == 0) && ((i + 1) < argc)) {
        nclFile.assign(argv[i+1], strlen(argv[i+1]));
        cout << "argv = '" << argv[i+1] << "' nclFile = '";
        cout << nclFile << "'" << endl;
        isXML = true;

}
```

- No arquivo main.cpp é necessário incluir o cabeçalho do componente ExportXML, para isso inclua a linha abaixo no cabeçalho do arquivo:

```
#include “recommender/exportXML”
```

- No arquivo main.cpp é necessário incluir a chamada para a exportação do arquivo XML, para isso inclua as linhas abaixo antes da chamada do método startPresentation():

```
if(isXML) {
        exportXML(nclFile);
}
```