

Grande Organizador de Dados (GOD)

Great Data Organizer

Programação Orientada a Objetos (MAC0441/5714)
Departamento Ciência da Computação – DCC
Instituto de Matemática e Estatística – IME
Universidade de São Paulo – USP

Dezembro, 2014

Sumário

1	Introdução	3
2	Funcionalidades principais	3
2.1	Módulos	3
2.1.1	Kernel	3
2.1.2	Banco de dados	3
2.1.3	E-mails	3
2.1.4	Filtros	3
2.1.5	Gráficos	4
2.1.6	Páginas web	4
2.1.7	Planilhas	4
2.1.8	Processadores	4
2.1.9	Redes sociais	4
2.1.10	Textos	5
2.2	Aplicações	5
2.2.1	Agregação de conferências	5
2.2.2	Agregação de informações acadêmicas	5
2.2.3	Análise de sentimento de consumo	6
2.2.4	Análise de sentimento político	6
3	Núcleo (GODKernel)	7
3.1	GODData	7
3.1.1	Atributos	7
3.1.2	Métodos	7
3.1.3	Exemplos	8
3.2	ConfigurationOfGOD	8
3.2.1	Métodos	8
3.2.2	Exemplos	8
4	Banco de dados (GODBases)	9
4.1	Magma	9
4.2	Classes do GODBases	9
4.2.1	GDBServerController	9
4.2.2	GDBClientController	10
4.2.3	GDBDatabase	10
4.2.4	GDBData	11
4.2.5	GDBConference	11
5	E-mails (GODEmail)	12
5.1	MAILService	12
5.2	MAILSender	13
5.3	MAILReceiver	14
5.4	MAILSecurePOP3Client	15

5.5	Exemplos	15
6	Filtros (GODFilter)	16
6.1	FILTMainFilter	16
6.2	FILTMainFilterTest	16
6.3	FILTTTextFilter	17
6.4	FILTTTextFilterTest	17
6.5	FILTConferenceFilter	17
6.6	FILTConferenceFilterTest	18
6.7	Exemplos	18
7	Gráficos (GODGraphGenerator)	20
7.1	Classes	20
7.2	Entrada	20
7.3	Saída	20
7.4	Exemplo de uso	21
8	Web (GODWeb)	23
8.1	WEBPage	23
8.1.1	Atributos	23
8.1.2	Métodos	23
8.1.3	Exemplos	23
8.2	WEBElememt	24
8.2.1	Atributos	24
8.2.2	Métodos	24
8.2.3	Classes de uma WEBPage	24
8.3	WEBForm	25
8.3.1	Atributos	25
8.3.2	Métodos	25
8.3.3	Exemplos	25
8.4	WEBFormElement	25
8.4.1	Classes de um formulário	26
8.4.2	Exemplos	26
8.5	WEBFileLibrary	27
8.5.1	Métodos	27
8.5.2	Exemplos	27
8.6	WEBGodHome	27
8.6.1	Atributos	27
8.6.2	Métodos	27
8.7	WEBHomeView	28
8.8	WEBPageFetcher	28
8.8.1	Métodos	28
8.8.2	Exemplos	28
8.9	Usando o GODWeb	28
8.9.1	Exemplos	29

9 Planilhas (GODSpreadsheet)	30
9.1 Estrutura	30
9.1.1 Indexação em todos os níveis	30
9.2 How to do	30
9.2.1 Abrir arquivos	31
9.2.2 Salvar arquivos	31
9.2.3 Criar uma SSSpreadsheetData e populá-la	31
9.2.4 Executar alguma ação em todas as células da planilha	32
9.2.5 Diferentes modos de acessar uma célula	32
10 Processadores (GODProcessors)	33
10.1 PCSStatisticsCalculator	33
10.1.1 Métodos	33
10.1.2 Exemplos	33
10.2 PCSTagger	34
10.2.1 Variáveis de Instância	34
10.2.2 Métodos	34
10.2.3 Exemplos	34
10.3 PCSGODTagger	34
10.3.1 Variáveis de Instância	35
10.3.2 Métodos	35
10.3.3 Exemplos	35
10.4 PCSPreprocessor	35
10.4.1 Variáveis de Instância	35
10.4.2 Métodos	36
10.4.3 Exemplos	36
11 Redes sociais (GODSocialNetIO)	37
11.1 Team	37
11.2 Description	37
11.3 Classes of Fetchers	37
11.3.1 SNETFetcher	37
11.3.2 SNETFacebookFetcher	37
11.3.3 SNETTwitterFetcher	38
11.4 Classes of GODDatas	39
11.4.1 SNETTwitterTweetsData	39
11.4.2 SNETTwitterTweet	40
11.4.3 SNETFacebookPostsData	40
11.4.4 SNETFacebookPost	40
11.4.5 SNETFacebookComment	41
11.4.6 SNETFacebookPagesData	41
11.4.7 SNETFacebookPage	41
11.4.8 SNETFacebookPageDescriptionData	41
11.4.9 SNETFacebookGroupsData	42
11.4.10 SNETFacebookGroup	42

11.4.11 SNETFacebookGroupDescriptionData	42
11.4.12 SNETFacebookEventsData	42
11.4.13 SNETFacebookEvent	43
11.4.14 SNETFacebookUserData	43
11.4.15 Exemplos	43
12 Textos (GODTextIO)	44
12.1 Equipe	44
12.2 Pré-requisitos	44
12.3 Classes	44
12.4 Exemplos	46
13 Agregador de informações acadêmicas (GODAcademics)	47
13.1 Classe ACADPaper	47
13.1.1 Atributos de instância	47
13.1.2 Métodos de instância	47
13.2 Classe ACADQualis	47
13.2.1 Atributos de classe	48
13.2.2 Atributos de instância	48
13.2.3 Métodos de classe	48
13.2.4 Métodos de instância	48
13.3 Classe ACADResearcher	48
13.3.1 Atributos de instância	48
13.3.2 Métodos de instância	49
13.4 Classe ACADTag	49
13.4.1 Atributos de instância	49
13.5 Classe ACADTagPool	49
13.5.1 Atributos de classe	50
13.5.2 Atributos de instância	50
13.5.3 Métodos de classe	50
13.5.4 Métodos de instância	50
13.6 Classe ACADAcademicsHome	50
13.7 Classe ACADAcademicsProfile	50
13.8 Classe ACADAcademics	50
13.8.1 Atributos de instância	50
13.8.2 Métodos de instância	51
13.9 Exemplos	51
14 Agregador de conferências (GOD's Call)	52
14.1 Equipe	52
14.2 Descrição geral	52
14.3 Tutorial de inicialização	52
14.3.1 Dependências	52
14.3.2 Inicialização da aplicação	52
14.4 Implementação	52

14.4.1 Modelos do domínio	52
14.4.2 Crawling	53
14.4.3 Web Application	53
14.4.4 Testes	54
14.5 Exemplos	54
15 Análise de sentimentos de consumo e político (GODSentimentAnalysis)	55
15.1 Introdução	55
15.2 SAInterface	55
15.2.1 Léxicos de Sentimento	55
15.3 SAFetcher	56
15.4 CSAApp	56
15.5 PSAAApp	56
15.6 PSAPoliticalDataLoader	56
15.7 SAApp	56
15.8 SAOutputGenerator	56
15.9 SASentimentAnalyser	56
15.10SASentimentLabel	57
15.11SASentimentTable	57
15.12Rodando as Aplicações	57
15.12.1 Análise de Sentimento de Consumo	57
15.12.2 Análise de Sentimento de Político	57
15.13Exemplos de código	58
16 Diagramas de classe do GOD	60
16.1 GOD	60
16.2 Banco de dados (GODBases)	60
16.3 E-mails (GODEmail)	61
16.4 Filtros (GODFilter)	61
16.5 Gráficos (GODGraphGenerator)	62
16.6 Planilhas (GODSpreadsheet)	63
16.7 Processadores (GODProcessors)	64
16.8 Redes sociais (GODSocialNetIO)	64
16.9 Textos (GODTextIO)	65
16.10Web (GODWeb)	66
16.11Agregação de conferências (GODAcademics)	66
16.12Agregação de informações acadêmicas (GODsCall)	67
16.13Análise de sentimentos de consumo e político (GODSentimentAnalysis)	68
17 Requisitos não implementados e sugestões de requisitos futuros	69
17.1 Banco de dados (GODBases)	69
17.2 E-mails (GODEmail)	69
17.3 Filtros (GODFilter)	69
17.4 Gráficos (GODGraphGenerator)	69
17.5 Planilhas (GODSpreadsheet)	70

17.6 Processadores (GODProcessors)	70
17.7 Redes sociais (GODSocialNetIO)	70
17.8 Textos (GODTextIO)	71
17.9 Web (GODWeb)	71
17.10 Agregação de conferências (GODAcademics)	71
17.11 Agregação de informações acadêmicas (GODsCall)	71
17.12 Análise de sentimentos de consumo e político (GODSentimentAnalysis)	72
17.13 Sugestões de novos módulos de fontes de dados	72
17.14 Sugestões de novas aplicações	72
18 Instalação	74
18.1 Pré-requisitos	74
18.1.1 Instalação do Metacello	74
18.1.2 Instalação do Seaside	74
18.1.3 Instalação do Magma	74
18.1.4 SqueakSSL-bin	75
18.1.5 pdflatex e abiword	76
18.1.6 Repositórios do GOD	77
18.1.7 Instalação do GOD	77
18.1.8 Inicialização do GOD	77
18.1.9 Problemas identificados	77

1 Introdução

Grande Organizador de Dados (GOD) é um sistema composto por módulos que realizam a entrada e saída de diversas fontes de informação, como arquivos texto, planilhas, html, emails e redes sociais, entre outros. Essas informações podem ser utilizadas para a criação de aplicações que fazem a análise de dados de diferentes formas, como análise de sentimentos e agregadores de informações. O GOD foi desenvolvido em Smalltalk pelos alunos da disciplina de Programação Orientada a Objetos. A seguir são descritas as principais funcionalidades dos módulos e das aplicações do GOD.

2 Funcionalidades principais

2.1 Módulos

2.1.1 Kernel

1. Fornecer uma estrutura de dados básica para a manipulação das diferentes fontes de informação. A classe GODData representa essa estrutura, que é estendida pelos demais módulos para atender suas especificidades.
2. Gerenciar o versionamento do projeto.

2.1.2 Banco de dados

1. Armazenamento de objetos
2. Realização de consultas sobre objetos armazenados
3. Gerenciamento do servidor
4. Gerenciamento de conexões de clientes

2.1.3 E-mails

1. Recebimento de e-mails utilizando uma conexão segura
2. Envio de e-mails utilizando uma conexão segura

2.1.4 Filtros

1. Filtragem de dados por título
2. Filtragem de dados por conteúdo
3. Filtragem de dados por origem
4. Filtragem de dados por tags
5. Filtragem de string por marcadores de texto (por exemplo tags html)

2.1.5 Gráficos

1. Geração de gráfico de barras no formato PNG
2. Geração de gráfico de barras no Squeak (Morph)
3. Geração de gráfico de linhas no formato PNG
4. Geração de gráfico de linhas no Squeak (Morph)

2.1.6 Páginas web

1. Criação de aplicações web
2. Leitura de páginas HTML

2.1.7 Planilhas

1. Leitura e escrita em CSV
2. Leitura e escrita em ODS
3. Leitura em XLSX
4. Junção de duas planilhas
5. Geração de planilhas a partir de dicionários e listas

2.1.8 Processadores

1. Geração de tags relevantes. As tags são obtidas a partir de conteúdo de texto de entrada de acordo com a sua relevância usando o algoritmo de recuperação de informação tf-idf.
2. Cálculo de média, mediana, variância e desvio padrão para dados numéricos

2.1.9 Redes sociais

Facebook

1. Busca de usuários
2. Busca por posts de usuário
3. Filtragem dos posts
4. Busca de páginas
5. Busca por grupos

Twitter

1. Busca na timeline

2. Busca por hashtags
3. Busca de tweets
4. Filtragem por data e por língua
5. Realizar buscas avançadas:
 - (a) Busca contendo todas as palavras
 - (b) Busca por qualquer uma das palavras
 - (c) Busca pela frase exata
 - (d) Busca por qualquer hashtag
 - (e) Busca por todas as hashtags
 - (f) Busca contendo nenhuma das palavras

2.1.10 Textos

1. Leitura e escrita em ODT
2. Leitura e escrita em TXT
3. Leitura e escrita em RTF
4. Leitura e escrita em PDF

2.2 Aplicações

2.2.1 Agregação de conferências

1. Busca por conferências
2. Agregação de informações sobre conferências
3. Leitura de dados do ConfSearch
4. Leitura de dados do WIKICFP
5. Leitura de dados do Qualis
6. Geração de planilha com as conferências resultantes da busca
7. Geração de palavras-chave para conferências

2.2.2 Agregação de informações acadêmicas

1. Busca por pesquisadores a partir de perfis do Google Scholar
2. Cálculo de estatísticas sobre um pesquisador
3. Geração de gráfico de fator de impacto e quantidade de artigos publicados

2.2.3 Análise de sentimento de consumo

1. Contagem de palavras positivas, neutras ou negativas em textos
2. Saída da análise em gráfico
3. Saída da análise em planilha

2.2.4 Análise de sentimento político

1. Contagem de palavras positivas, neutras ou negativas sobre candidatos ou partidos
2. Validação de dados do TSE sobre candidatos e partidos
3. Saída da análise em gráfico
4. Saída da análise em planilha

3 Núcleo (GODKernel)

GODKernel é o módulo que contém a classe que representa a estrutura básica para manipulação de dados do projeto. Essa classe é chamada **GODData** contém os atributos para armazenamento comuns aos diferentes tipos de fontes de dados usados no projeto. **GODData** pode ser estendido para atender às especificidades dessas diferentes fontes de informações.

Esse módulo também é responsável por gerenciar o repositório do projeto, realizando o controle de versões e suas dependências.

Em caso de dúvidas, entre em contato com *Higor: hamario [at] ime.usp.br.*

3.1 GODData

Classe que representa a estrutura básica a serem armazenadas das diferentes fontes de informação, como arquivos texto, planilhas, páginas html, redes sociais, e-mails, entre outras.

3.1.1 Atributos

- **author** - autor de um conteúdo.
- **content** - armazena o conteúdo principal de uma fonte de informação.
- **height** - altura do objeto a ser renderizado.
- **id** - valor exclusivo do objeto.
- **layoutGrid** - matriz que contém objetos **GODData** internos. Atributo que representa a posição na qual os objetos internos devem ser renderizados em uma página web.
- **origin** - origem da informação.
- **tags** - termos principais associados ao objeto **GODData**.
- **timestamp** - data do **GODData**.
- **title** - Título da informação.
- **width** - largura do objeto a ser renderizado.

3.1.2 Métodos

- **addOnLayoutGrid:aGODData at: aRow at: aColumn** - adiciona um **GODData** ao **layoutGrid**.
- **addTag:** - adiciona uma nova tag à coleção de tags.
- **isGODData** - verifica se um objeto é do tipo **GODData**.
- **isWho** - informa o tipo de subclasse de **GODData**.

- **layoutGridRowSize:** `rowNumber` **columnSize:** `columnNumber` - define as dimensões do layoutGrid.
- **tagExists:** - verifica se uma determinada tag existe.

3.1.3 Exemplos

```
'Preenchendo um \goddata'
goddata := \godData~ new.
goddata title: 'titulo do objeto'.
goddata author: 'proprietario da informacao'
goddata content: 'Conteudo que faz parte da informacao'
goddata addTag: 'termo'.

'Definindo o tamanho do layoutGrid e atribuindo objetos \goddata~ a este grid'
goddata {layoutGridRowSize: 5 columnSize: 3.
goddata addOnLayoutGrid: goddata1 at: 1 at: 1.
goddata addOnLayoutGrid: goddata1 at: 3 at: 3.

'Adicionando uma tag a colecao'
goddata addTag: 'termo'.
```

3.2 ConfigurationOfGOD

Classe responsável pelo gerenciamento dos módulos do projeto. Possui módulos para fazer o download de versões dos módulos e dependências do projeto, assim como a realização de backup do repositório.

3.2.1 Métodos

- **backupPackages:** - realiza o backup dos módulos do projeto.
- **baseline01:** - inclui as descrições dos pacotes e dependências do projeto.

3.2.2 Exemplos

```
'Baixando a ultima versao dos pacotes do projeto'
(\configGod~ project version: '0.1-baseline01') load.
```

4 Banco de dados (GODBases)

O módulo de banco de dados orientado a objetos (GODBases) foi elaborado com o intuito de compor o Projeto GOD e cumprir os requisitos da disciplina de Programação Orientada a Objetos. Tem por objetivo fornecer um banco de dados capaz de armazenar informações sobre o projeto GOD, possibilitando operações de inserção, remoção e busca. Seu desenvolvimento foi realizado no Squeak Smalltalk, através do banco de dados orientado a objetos Magma.

Contato: Lucy Mansilla (lucyacm@ime.usp.br) e Silvia Scheunemann Silva (silviass@ime.usp.br).

4.1 Magma

Magma é uma biblioteca de banco de dados orientado a objetos disponível no Squeak, no modo monusuário (single-user) e também multiusuário (arquitetura cliente-servidor). Os pacotes e métodos disponíveis podem ser obtidos através do download na página:

- <http://map.squeak.org/packagesbyname>.

4.2 Classes do GODBases

O pacote GODBases contém os principais métodos do banco de dados orientado a objetos do projeto GOD. A seguir fazemos uma pequena descrição sobre cada uma de suas classes.

4.2.1 GDBServerController

A classe GDBControllerServer permite fazer transações diferentes no servidor do banco de dados, bem como criar e eliminar um repositório.

- Para criar um repositório padrão, execute a seguinte linha no workspace:
GDBServerController createRepository.
- Para criar um repositório específico, execute:
GDBServerController createRepository: 'nomeDoRepositorio'.
- Para inicializar o servidor com um repositório padrão, execute:
GDBServerController startServer.
- Para inicializar o servidor com um repositório específico, execute:
GDBServerController startServer:'nomeDoRepositorio'.
- Para saber se o servidor foi iniciado, execute: GDBServerController isStarted.
esse comando retorna 'true' se o servidor estiver inicializado.

Uma vez inicializado o servidor, salve como uma imagem-servidor. Assim, para fechar o servidor, só precisa fechar a imagem salva, e quando você abri-lo novamente, o servidor já estará funcionando.

- Uma outra opção para fechar o servidor, é executar:
GDBServerController stopServer.
- Para apagar um repositório específico, execute:
GDBServerController deleteRepository:'nomeDoRepositorio'

4.2.2 GDBClientController

A classe GDBClientController permite fazer a conexão de um usuário/ cliente com o servidor do banco de dados. Primeiro é necessário inicializar o servidor em uma imagem separada, como indicado anteriormente, em seguida, executar as seguintes instruções:

- Para conectar um cliente com o banco de dados, execute o seguinte:
`GDBClientController initializeSession`

Uma vez inicializada a sessão do usuário é possível fazer qualquer transação no banco de dados, ou seja, é possível fazer uso dos métodos contidos nas classes: GDBDatabase, GDBData e GDBConference.

- Para atualizar qualquer alteração de outro cliente, antes de qualquer transação execute:
`GDBClientController refresh`.
- Para alterar a sessão do cliente, execute o seguinte:
`GDBClientController refresh: 'nomeDeOutroCliente'`.

Uma vez que não se precise de fazer nenhuma transação a mais no banco de dados, só temos que desconectar a sessão do cliente, para isso execute o seguinte:

`GDBClientController release`.

4.2.3 GDBDatabase

A classe GDBDatabase permite criar o banco de dados. Nossa banco de dados é representado por um dicionário que armazena duas coleções, uma coleção do tipo MagmaCollection para armazenar os objetos GODData e uma outra do tipo OrderedCollection para armazenar uma coleção que contém objetos GCConference.

- Para criar o banco de dados, execute:
`GDBDatabase createDatabase`
- Se você quiser criar apenas o banco de dados para armazenar objetos GODData, execute:
`GDBData createGODDataCollection`.
- Se você quiser criar apenas o banco de dados para armazenar a coleção que vai conter objetos GCConference, execute:
`GDBConference createConferenceCollection`.

4.2.4 GDBData

A classe GDBData permite fazer transações diferentes no banco de dados para objetos GODData.

- Para inserir um novo objeto GODData no banco de dados, execute:
GDBData add: meuGODData.
- Para remover um objeto GODData do banco de dados, execute:
GDBData remove: meuGODData.
- Para atualizar um objeto GODData por um outro objeto GODData, execute:
GDBData update: meuAntigoGODData with: meuNovoGODData.
- Se você quiser apagar todos os objetos GODData armazenados no banco de dados, execute:
GDBData reloadData.
- Para buscar um objeto GODData usando parte do nome do autor, execute:
GDBData searchAuthor: 'parteDoNomeDoAutor'.
- Para buscar um objeto GODData usando parte do nome do titulo, execute:
GDBData searchTitle: 'parteDoNomeDoTitulo'.
- Para buscar um objeto GODData usando um bloco, execute:
GDBData searchFor: umBloco.
onde um bloco deve ter a forma:
[: *objeto|objetoatributoDoObjetoGODData* = *valorDeComparacaoParaABusca*].
- Para buscar um objeto GODData usando seu ID, execute:
GDBData searchById: umIdDoTipoInteiro.
- Para fazer uma busca exata pelo nome do autor, execute:
GDBData exactSearchByAuthor: 'nomeExatoDoAutor'.
- Para fazer uma busca exata pelo titulo, execute:
GDBData exactSearchByTitle: 'nomeExatoDoTitulo'.

4.2.5 GDBConference

A classe GDBConference permite fazer diferentes transações no banco de dados para objetos GC-Conference.

- Para inserir uma nova coleção que armazena objetos GCConference, execute:
GDBConference saveConferences: minhaColeçãoGCConference.
- Para recuperar essa coleção armazenada no banco de dados, execute:
GDBConference loadConferences.
- Se você quiser apagar essa coleção armazenada no banco de dados, execute:
GDBConference resetConferenceData.

5 E-mails (GODEmail)

Pacote responsável pelo envio e recebimento de e-mails, tendo suporte a conexões seguras. É formado por 4 classes: uma abstrata (GODService), duas que implementam os serviços de envio e recebimento de e-mails (GODSender e GODReceiver, respectivamente) e uma que implementa o cliente do protocolo Post Office Protocol 3 (POP3) utilizando conexão segura (MAILSecurePOP3Client). Possui o pacote SqueakSSL como dependência externa. Para o envio de e-mails usando Secure Socket Layer (SSL) é utilizada a classe SecureSMTPClient já implementada no SqueakSSL-SMTP.

Para receber e enviar emails pode ser necessário configurar a conta de emails para permitir receber/enviar emails via POP3/SMTP.

Em caso de dúvidas, entre em contato com *Luiz Armesto*: *luiz.armesto@gmail.com*.

5.1 MAILService

Classe abstrata que reúne o código compartilhado pela MAILSender e pela MAILReceiver. Os métodos de instância estão organizados em três categorias diferentes:

- **Métodos de acesso (accessing):**
 - **hostname** - Getter para hostname.
 - **hostname:** - Setter para hostname.
 - **password:** - Setter para password.
 - **port** - Getter para port.
 - **port:** - Setter para a port.
 - **user** - Getter para user.
 - **user:** - Setter para user.
- **Métodos de inicialização (initialize-release):**
 - **initialize** - Faz uma chamada para o método initializeConnectionInfo.
 - **initializeConnectionInfo** - Inicializa os dados pertinentes à conexão do usuário.
- **Métodos Privados (private):**
 - **validateConnectionInfo** - Considera válida uma conexão na qual os dados estejam preenchidos adequadamente. Retorna um erro caso exista algum campo vazio.

Os métodos estáticos estão organizados em duas categorias diferentes:

- **Métodos de criação (instance creation):**
 - **connectTo:identifiedAs:password:** - Construtor que recebe o host do servidor, nome de usuário e senha e cria uma instância já configurada. É utilizada a porta padrão definida pela classe filha.

- **connectTo:port:identifiedAs:password:** - Construtor que recebe o host do servidor, porta, nome de usuário e senha e cria uma instância já configurada.

- **Métodos de acesso (accessing):**

- **defaultPortNumber** - Método que deve ser implementado pelas classes filhas para definir a porta padrão usada para conectar aos servidores caso o cliente não a configure.

5.2 MAILSender

Classe responsável pelo envio de e-mail utilizando o protocolo POP3. Herda de MAILService. Os métodos de instância desta classe estão organizados em quatro categorias diferentes:

- **Métodos de acesso (accessing):**

- **smtpClient:** - Setter para um cliente do protocolo Simple Mail Transfer Protocol (SMTP)

- **Métodos de inicialização (initialize-release):**

- **initializeConnectionInfo** - Realiza um chamada para o método initializeConnectionInfo da classe superior (MAILService), considerando que é um cliente SMTP.

- **Métodos de requisição (requests):**

- **send:to:** - Envia um GODData para um destino (o destino é uma string contendo o e-mail). Internamente, ele realiza os seguintes processos relacionados ao envio: validação da conexão, checagem do dado (se o conteúdo está vazio), conversão de GODData para mensagem de e-mail, inicialização do cliente SMTP e, finalmente, o próprio envio.

- **Métodos privados (private):**

- **getClientClass:** - Se a variável de instância já estiver iniciada (smtpClient), a usará. Caso contrário, busca obter a classe correta do cliente a partir do numero da porta. Por padrão, este método tentará usar uma conexão segura, chamando a classe SecureSMTPClient. Retorna o cliente.

Já os métodos estáticos estão organizados em três categorias diferentes:

- **Métodos de acesso (accessing):**

- **defaultPortNumber** - Getter para a porta padrão usada quando nenhuma for especificada pelo cliente (465).

- **Métodos de conversão (converting):**

- **convertToHTMLEmail:recipient:** - Recebe um objeto GODData e um endereço de e-mail e retorna uma mensagem com o conteúdo do objeto, destinada ao e-mail fornecido. Utiliza o formato HTML.

- **convertToSimpleEmail:recipient:** - Recebe um objeto GODData e um endereço de e-mail e retorna uma mensagem com o conteúdo do objeto, destinada ao e-mail fornecido. O conteúdo é criado em texto puro.
- **Métodos de exemplo (example):**
 - **example** - Contem um exemplo simples de como usar a classe para enviar e-mails.

5.3 MAILReceiver

Classe responsável pelo recebimento de emails utilizando o protocolo SMTP. Herda de MAILService. Os métodos de instância desta classe estão organizados em quatro categorias diferentes:

- **Métodos de acesso (accessing):**
 - **pop3Client:** - Setter de um cliente POP3.
- **Métodos de inicialização (initialize-release):**
 - **initialize** - Faz uma chamada para o método initialize da super classe (MAILService) e inicializa os objetos relacionados ao recebimento de e-mails.
 - **initializeConnectionInfo** - Faz uma chamada para o método initializeConnectionInfo da super classe (MAILService) para um cliente POP3.
- **Métodos de requisição (requests):**
 - **receive** - Este método conecta ao servidor, recebe os novos e-mails e os converte para GODData. Internamente, ele realiza os seguintes processos relacionados ao recebimento: validação da conexão, inicialização do cliente POP3, realização do login do cliente, obtenção das mensagens e o fechamento da conexão.
- **Métodos privados (private):**
 - **getClientClass:** - Se a variável de instância já estiver iniciada (pop3Client), a usará. Caso contrário, busca obter a classe correta do cliente a partir do numero da porta. Por padrão, este método tentará usar uma conexão segura, chamando a classe MAILSecurePOP3Client. Retorna o cliente.

Os métodos estáticos estão organizados em três categorias diferentes:

- **Métodos de acesso (accessing):**
 - **defaultPortNumber** - Getter para a porta padrão usada quando nenhuma for especificada pelo cliente (995).
- **Métodos de conversão (converting):**
 - **convertToGODData:** - Recebe uma mensagem de e-mail e devolve um GODData preenchido.

- **getContentFrom:** - Devolve o conteúdo principal de uma mensagem de e-mail. Caso a mensagem seja multipart, tenta pegar por padrão o conteúdo em texto puro, se não encontrar tenta em HTML. Utilizada pelo convertToGODData.
- **getField:From:** - Devolve o conteúdo de um determinado campo da mensagem de e-mail fornecida. Utilizado internamente pelo convertToGODData.
- **getTimestampFrom:** - Devolve a data da mensagem de e-mail. Usado pelo convertToGODData.

- **Métodos de exemplo (example):**

- **example** - Contem um exemplo simples de como usar a classe para receber e-mails.

5.4 MAILSecurePOP3Client

Classe responsável pela implementação do protocolo POP3 a partir do SqueakSSL para criar uma conexão segura. Herda da classe POP3Client do próprio Squeak e sobrescreve o método que cria a conexão com o servidor.

- **Possui apenas um método privado (private):**

- **ensureConnection** - Cria conexão com o servidor do mesmo modo que a classe POP3Client faria, com a única diferença de utilizar o SqueakSSL para permitir uma conexão segura.

5.5 Exemplos

```
'Recebendo emails'
rcv := MAILReceiver new.
emailsRec := OrderedCollection new.
rcv hostname: 'pop.server.com'.
rcv user: 'username@server.com'.
rcv port: 110.
rcv password: 'password'.
emailsRec := rcv receive.

'Enviando email'
gdata := GODData new.
gdata author: 'GODEmail';
    content: 'The content of a GODData';
    title: 'GODData Test'.

snd := MAILSender new.
snd hostname: 'smtp.server.com'.
snd user: 'username@server.com'.
snd password: 'password'.
snd port: 465.
snd send: gdata1 to: 'anotheruser@server.com'.
```

6 Filtros (GODFilter)

Grupo: Carlos Ribas, Yoshio Mori, Larissa Moraes

Contato: larissam@ime.usp.br

Uma das necessidades do projeto GOD era filtrar os objetos da classe GODData para serem utilizados por aplicações específicas. Nesse contexto, foi criado o pacote GODFilter que visa criar métodos para atender a necessidade de filtros dos grupos de aplicações.

O pacote GODFilter contém três classes, FILTMainFilter, FILTTextFilter e FILTConferenceFilter, sendo que a primeira é uma superclasse e as duas últimas são subclasses da primeira.

Além disso, foram implementados testes de unidade para cada classe do GODFilter, sendo criadas as classes FILTMainFilterTest, FILTTextFilterTest e FILTConferenceFilterTest. Os testes facilitam a manutenção, pois indicam se a unidade ainda está funcional após a realização de alterações no código.

Nas subseções abaixo serão detalhadas as características de cada classe.

6.1 FILTMainFilter

A classe FILTMainFilter possui métodos para a realização de filtros em coleções de GODData que podem ser utilizados por qualquer aplicação. Foram criados quatro métodos com as seguintes assinaturas:

- ***filter:collectionOfGodData byContent:content*** - filtra uma coleção de objetos da classe GODData pelo seu conteúdo, ou seja, pelo atributo *content*.
- ***filter:collectionOfGodData byOrigin:origin*** - filtra uma coleção de objetos da classe GODData pela sua origem, ou seja, pelo atributo *origin*.
- ***filter:collectionOfGodData byTags:tags*** - filtra uma coleção de objetos da classe GODData pelas suas tags, ou seja, pelo atributo *tags*.
- ***filter:collectionOfGodData byTitle:title*** - filtra uma coleção de objetos da classe GODData pelo seu título, ou seja, pelo atributo *title*.

Como foi utilizado o método ***match:text*** de Smalltalk para encontrar as strings nos atributos dos objetos de GODData, pode ser passado como parâmetro cas* para encontrar strings como: casa, casado ou casamento, por exemplo.

6.2 FILTMainFilterTest

Esta classe possui um método chamado *setUp* que contém uma coleção de *GODData*. Estes dados são utilizados em todos os testes da classe *FILTMainFilterTest*. Os seguintes testes foram implementados:

- *testFilterByTag*: testa a realização de filtro de GODData por Tags.
- *testFilterByContent*: testa a realização de filtro de GODData por Conteúdo.
- *testFilterByOrigin*: testa a realização de filtro de GODData por Origem.
- *testFilterByTitle*: testa a realização de filtro de GODData por Título.

6.3 FILTTextFilter

A classe FILTTextFilter foi criada para atender os requisitos dos grupos de aplicação que precisavam filtrar uma página HTML. Como o HTML nada mais é que um texto contendo tags, esse filtro passou a servir para textos em geral.

Para isso, foi criado um método que recebe um texto como parâmetro e duas marcações, inicial e final. Esse método retorna uma coleção de strings que foram encontradas entre a marca inicial e a marca final. Lembrando que a string retornada não pode conter nenhuma das marcações.

O método criado tem como assinatura:

filterText: text between: startMark and: endMark.

6.4 FILTTextFilterTest

Esta classe possui um método chamado *setUp*. Este método contém uma variável chamada "*text*" que recebe um texto em formato HTML e outra variável chamada "*newtext*" que recebe um texto simples. Além disso, este método possui três *OrderedCollection* com os resultados que são esperados para os testes. Os seguintes testes foram implementados:

- *testFilterTextBetweenAnd*: este teste verifica o conteúdo de um texto que se encontra entre duas *strings*. A primeira *string* indica o início do filtro e a segunda *string* determina o término do filtro. O resultado da busca deve ser igual ao valor armazenado nas *OrderedCollections* presentes no método *setUp*.
- *testFilterTextBetweenAndWithTextEmpty*: este teste realiza um filtro em uma variável que está vazia. Logo, espera-se que o resultado disso seja uma *OrderedCollection* vazia.

6.5 FILTConferenceFilter

A classe FILTConferenceFilter foi criada a pedido do grupo da aplicação God's Call, pois eles precisavam filtrar os dados de todas as conferências que eles tinham para saber quais delas atendiam à busca do usuário.

Para isso, foi criado um método que recebe como parâmetro uma coleção de objetos da classe GODConference e uma chave de busca representada por um objeto da classe GCSearchFilter e retorna uma coleção de GODConference considerando as seguintes condições:

- keywords de GODConference contém keywords de GCSearchFilter;
- categories de GODConference contém categories de GCSearchFilter;
- deadlineDate de GODConference \leq deadlineDate de GCSearchFilter;
- startDate de GODConference \geq startDate de GCSearchFilter;
- endDate de GODConference \leq endDate de GCSearchFilter;

O método desconsidera os atributos do objeto GODConference ou do GCSearchFilter que são vazios, deixando de fazer a comparação entre os objetos.

O método criado tem como assinatura:

filter:collectionOfGodConference byGCSearchFilter:searchFilter

6.6 FILTConferenceFilterTest

Esta classe possui um método chamado *setUp* que contém uma coleção de *GCConference*. Estes dados são utilizados em todos os testes da classe *FILTConferenceFilterTest*. Os seguintes testes foram implementados:

- *testFilterBadObject*: testa uma coleção que possui objetos incorretos. Neste caso, o método deve ignorar os parâmetros que possuem valores incorretos. O resultado deste teste são todas as conferências que possuem a *keyword* Automaton, pois este é único parâmetro que possui um valor válido.
- *testFilterByGCSearchFilter*: teste com valores específicos. Busca as conferências cadastradas no método *setUp* que atendem aos parâmetros passados.
- *testFilterByGCSearchFilter02*: teste com valores específicos. Busca as conferências cadastradas no método *setUp* que atendem aos parâmetros passados.
- *testFilterByGCSearchFilter03*: teste com valores específicos. Busca as conferências cadastradas no método *setUp* que atendem aos parâmetros passados. Neste teste não foi passado o parâmetro *categories*.
- *testFilterByGCSearchFilter04*: teste com valores específicos. Busca as conferências cadastradas no método *setUp* que atendam aos parâmetros passados. Esse teste procura por conferências que possuem duas *keywords* específicas.
- *testFilterByGCSearchFilterWrongType*: verifica se o método gera exceção ao receber um atributo diferente da classe GCSearchFilter.
- *testFilterWrongType*: verifica se o método gera exceção ao receber um atributo diferente da classe OrderedCollection.

6.7 Exemplos

```
'Filtering GODConference objects'
gcfilt:= GCSearchFilter new.
gcfilt keywords: (Set with: 'software'; with: 'engineering').
gcfilt deadlineDate: (Date readFromString: '01 Jan 2014').

fconf:= FILTConferenceFilter new.
fconf filter: collectionOfConferences byGCSearchFilter: gcfilt.
```

```
'Filtering text between tags'  
txtDiv:= '<div> blablabla bla <div> XXX <div> YYY </div> </div><a>content</a><div> ZZZ </  
div></div>' .  
ftf:=FILTTTextFilter new.  
ftf filterText: txtDiv between: 'div>' and: '<' .  
  
'Filtering text in a GODData collection'  
f1:=FILTMainFilter new.  
f1 filter: collectionOfGODData byTitle: 'A handbook for dummies'.  
f1 filter: collectionOfGODData byContent: '*text*' .
```

7 Gráficos (GODGraphGenerator)

O módulo do gerador de gráficos recebe um Dictionary ou uma OrderedCollection e devolve um gráfico correspondente. O gráfico pode ser um objeto Morph ou uma imagem PNG. Quanto ao tipo de gráfico, pode ser gerado um gráfico de barras ou de linhas. Em caso de dúvidas, entre em contato com *Renan Teruo Carneiro*: renanteruoc@gmail.com.

7.1 Classes

- *GGGraph*: classe abstrata que representa um gráfico, ela por sua vez é um Morph do Squeak. Possui métodos para definir os nomes dos eixos do gráfico. Também é possível definir as dimensões do gráfico a ser gerado. O gráfico pode ser gerado no World do Squeak ou salvo como um arquivo PNG.
- *GGBarGraph* e *GGLineGraph*: subclasses de GGGraph que representam o tipo do gráfico. Recebem como entrada um Dictionary.
- *GGGraphOrderedCollectionDecorator* esta subclasse de GGGraph recebe uma OrderedCollection

A única especialização dos gráficos de barra e linha é a forma como eles são desenhados, todo o resto da estrutura é comum para todos os gráficos.

7.2 Entrada

O Dictionary com os dados de entrada precisa ter o formato: *palavra -> número*. Não é recomendado a inclusão de novas palavras no apóis o gráfico ter sido gerado.

Apesar dos campos de labels serem opcionais, é recomendado o seu uso, pois aumenta o comprimento dos eixos e gera mais espaço para o desenho do gráfico.

Outro tipo de dado de entrada pode ser uma OrderedCollection composta por duas OrderedCollections com o seguinte formato: *OrderedCollection(palavra), OrderedCollection(valor)*. Esse tipo de coleção foi definido para tornar possível ordenar os dados de acordo com a necessidade do gráfico a ser gerado. O tipo do gráfico é definido pela mensagem *delegate*, e "arrumamos" a entrada dentro do generateGraph dessa classe (GGGraphOrderedCollectionDecorator).

7.3 Saída

A saída é uma instância do tipo do gráfico, e todos os gráficos são subclasses de Morph (mais detalhes na seção 1.5).

Esse objeto pode ser exportado para um arquivo de imagem PNG, que será salvo em `$(squeak_dir)/Contents/Resources`, ou aberto no Squeak world.

Caso seja necessário mudar a extensão do arquivo de imagem, a saída pode receber qualquer mensagem destinada a objetos do tipo Morph.

7.4 Exemplo de uso

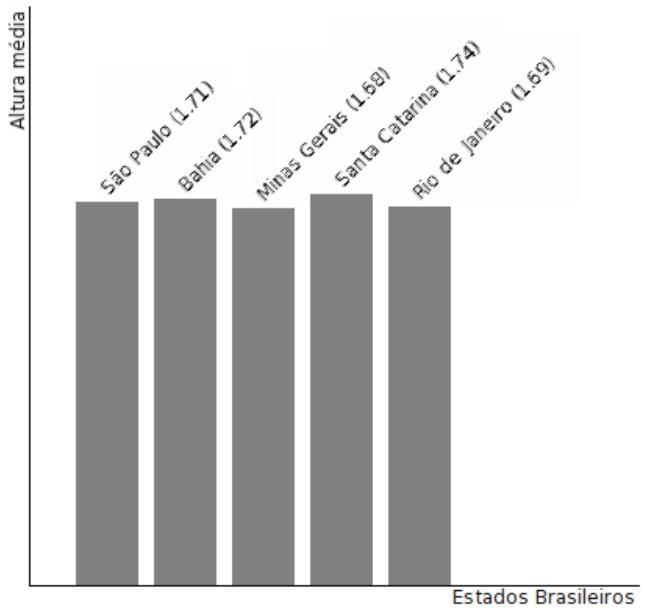
O código a seguir exemplifica a rotina para gerar o gráfico:

```
| dictionary |
```

```
graph := GGBarGraph new.
graph horizontalLabel: 'Estados Brasileiros'.
graph verticalLabel: 'Altura média'.
```

```
dictionary := Dictionary new.
dictionary at: 'São Paulo' put: 1.71;
at: 'Rio de Janeiro' put: 1.69;
at: 'Santa Catarina' put: 1.74;
at: 'Bahia' put: 1.72;
at: 'Minas Gerais' put: 1.68.
```

```
graph generateGraph: dictionary.
graph openInWorld.
graph color: Color gray.
graph exportToFile: 'estados_altura'.
```



Rotina:

1. Instanciar o gráfico que se quer gerar, `GGBarGraph` ou `GGLineGraph`
2. Adicionar os labels utilizando as mensagens `horizontalLabel` e `verticalLabel`
3. Instanciar um Dictionary e adicionar as palavras e valores
4. Gerar o gráfico com a mensagem `generateGraph`
5. Exportar ou abrir com `exportToFile` e `openInWorld`

OBS: Uma restrição do gráfico de linhas é que as chaves no Dictionary precisam ser números não negativos. Recomendamos estejam no intervalo de 0 mod (100).

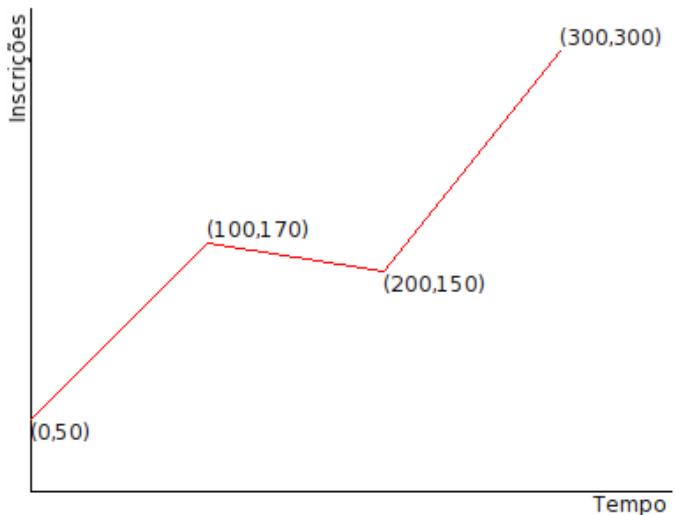
```

| dictionary |
graph := GGLineGraph new.
graph horizontalLabel: 'Tempo'.
graph verticalLabel: 'Inscrições'.

dictionary := Dictionary new.
dictionary at: 0 put: 50;
at: 100 put: 170;
at: 200 put: 150;
at: 300 put: 300.

graph generateGraph: dictionary.
graph openInWorld.
graph color: Color red.
graph exportToFile: 'tempo_inscricoes'

```



8 Web (GODWeb)

GODWeb é o módulo responsável pela geração das interfaces web do GOD. O **GODWeb** é baseado no framework Seaside. Usando o **GODWeb** é possível gerar páginas contendo diversos componentes como formulários, tabelas, imagens e texto, praticamente sem ter que escrever código do Seaside. Possui também classes para importar arquivos que serão usados nas páginas, como imagens e arquivos CSS. O **GODWeb** também fornece uma classe para obter o conteúdo HTML de URLs. Por fim, o módulo contém as classes que formam a página principal do GOD.

A seguir há uma descrição das classes principais, assim como uma breve descrição das classes que representam os *brushes* (tags HTML), como por exemplo brushes para texto, imagem, formulários, entre outros. Em caso de dúvidas, entre em contato com *Higor: hamario [at] ime.usp.br*.

8.1 WEBPage

Classe responsável pela renderização das páginas web. Contém uma lista de elementos que serão renderizados para construir uma página web.

8.1.1 Atributos

- **html** - Conteúdo dos elementos web que serão renderizados.
- **title** - Título da página.
- **elements** - Lista de elementos da página.

8.1.2 Métodos

- **add:** - Adiciona um elemento à página.
- **render:** - Chama a renderização do título e dos elementos.
- **renderElements:** - Renderiza a coleção de elementos.
- **renderTitle:** - Renderiza o título da página.

8.1.3 Exemplos

```
'Criando um WEBPage e incluindo diversos elementos'
page:= WEBPage new.
page initialize.
page title: 'Main page'.
page add: aParagraph.
page add: aHtmlText.
page add: anImage.
page add: aSpreadsheet.
^page.
```

8.2 WEBElement

Classe que representa a estrutura básica de um elemento HTML. Usada para criar novos elementos.

8.2.1 Atributos

- **html** - conteúdo dos elementos web que serão renderizados.

8.2.2 Métodos

- **render** - método que contém o código a ser renderizado. Deve ser chamado pelas suas subclasse. O método **render** deve conter o código do Seaside que renderiza o elemento.

8.2.3 Classes de uma WEBPage

As classes descritas abaixo representam os elementos básicos de uma página HTML.

- **WEBForm** - representa um formulário, mais detalhes da classe são mostrados em 8.3.
- **WEBFormElement** - elementos que fazem parte de um formulário. Mais detalhes sobre cada um dos elementos são mostrados em 8.4
- **WEBHtmlText** - representa um texto de uma página. Permite adicionar tags html ao texto renderizado.
- **WEBImage** - representa uma imagem a ser renderizada. Deve-se passar o local da imagem em **initialize:**. Pode-se também alterar a altura e largura da imagem.
- **WEBMorph** - representa um objeto do tipo Morph, passado como parâmetro em **initialize:**. Também é possível redefinir sua altura e largura.
- **WEBParagraph** - um parágrafo de texto da página.
- **WEBSpreadsheet** - uma tabela que recebe um SSSpreadsheet para renderização na página. WEBSpreadsheet pode receber adicionalmente uma lista de imagens e/ou uma lista de links para serem renderizados como colunas adicionais da tabela. Para isso, essas listas devem ser do mesmo tamanho do número de linhas do objeto SSSpreadsheet. Para isso, há diferentes métodos de inicialização que permitem iniciar uma tabela simples, uma tabela com uma lista de links, uma tabela com uma lista de imagens ou ainda uma tabela com ambas as listas. É possível ainda definir se a tabela possui cabeçalho (primeira linha de SSSpreadsheet), definir a largura das colunas e o tamanho da borda das linhas e das células.

```
'Incluindo um texto com html na pagina.'
iHtmlText := WEBHtmlText new.
iHtmlText value:'Texto <br> <strong>formatado</strong> </br> em HTML'.

'Adicionando uma imagem.'
iImage := WEBImage new initialize: 'caminho/do/arquivo.png'.
```

```

iImage height: 200.
iImage width: 400.

'Criando uma tabela com cabecalho contendo uma coluna com imagens e outra com links.'
iWebSpreadsheet := WEBSpreadSheet new initialize: spreadSheet withImages: imageList
    withLinks: linkList.
iWebSpreadsheet sheetNumber: 1.
iWebSpreadsheet header: true.
iWebSpreadsheet width: 1000.
iWebSpreadsheet cellBorder: 1.
iWebSpreadsheet rowBorder: 0.
iWebSpreadsheet imageHeader: 'Icons'.
iWebSpreadsheet linkHeader: 'Web Page'.

```

8.3 WEBForm

Classe que representa um formulário web. Os elementos de um formulário, que são subclasses de **WEBFormElement**, devem ser adicionados nesta classe para serem utilizados.

8.3.1 Atributos

- **elements** - lista de elementos do formulário.

8.3.2 Métodos

- **add:** - adiciona os elementos do formulário.
- **render:** - método que renderiza a coleção de elementos do formulário.
- **save** - esse método recebe o objeto da classe responsável pela renderização do formulário. Os atributos dessa classe representam os campos do formulário, que podem então ser usados pela aplicação para realizar o processamento dos dados.

8.3.3 Exemplos

```

'Criando um WEBForm e adicionando elementos.'
iForm := WEBForm new.
iForm initialize.
iForm add: anInputText.
iForm add: aDate.
iForm add: aCheckbox.
iForm add: aRadio.

```

8.4 WEBFormElement

Classe que representa um elemento básico de um formulário. Possui os atributos **label** e **value**. Deve ser estendida pelos elementos que serão usados em um formulário.

8.4.1 Classes de um formulário

Foram desenvolvidas classes que representam os elementos básicos de um formulário, essas classes são listadas abaixo.

- **WEBAnchor** - representa um link para uma url.
- **WEBCheckBox** - recebe e renderiza uma coleção de items do tipo checkbox.
- **WEBCheckBoxItems** - representa cada item de um checkbox.
- **WEBDate** - campo para seleção de data.
- **WEBDropDownList** - classe que recebe itens do tipo string para serem incluídos em um elemento do tipo dropdown list.
- **WEBFileUpload** - classe que recebe um arquivo para upload.
- **WEBInputText** - recebe uma string.
- **WEBRadioButton** - recebe e renderiza uma coleção de itens do tipo radio button.
- **WEBRadioButtonItem** - representa cada item de um radio group.
- **WEBSubmitButton** - classe que representa o botão de submit.

8.4.2 Exemplos

```
'Instanciando um WEBDropDownList'
iDrop := WEBDropDownList new.
iDrop label: 'Choose your preferred animal'.
iDrop add: 'Dog'.
iDrop add: 'Pig'.
iDrop add: 'Sheep'.

'Criando itens de um checkbox e adicionando ao WEBCheckBox'
iCheckboxItem1 := WEBCheckBoxItem new.
iCheckboxItem1 label: 'Apples'.
iCheckboxItem1 value: false.

iCheckboxItem2 := WEBCheckBoxItem new.
iCheckboxItem2 label: 'Bananas'.
iCheckboxItem2 value: false.

iCheckboxItem3 := WEBCheckBoxItem new.
iCheckboxItem3 label: 'Oranges'.
iCheckboxItem3 value: false.

iCheckbox := WEBCheckBox new.
```

```
iCheckbox label: 'Some fruit options:'.
iCheckbox add: iCheckboxItem1.
iCheckbox add: iCheckboxItem2.
iCheckbox add: iCheckboxItem3.
```

8.5 WEBFileLibrary

Classe que permite armazenar conteúdo de arquivos texto ou binários para serem usados nas aplicações. Dessa forma, pode-se armazenar as imagens de uma aplicação ou seu CSS, tornando a aplicação independente de arquivos locais e ainda gerenciando possíveis mudanças através de sua inclusão no repositório do projeto.

8.5.1 Métodos

- **add: aFilePath** - adiciona o arquivo à classe, que passa a ser um método de WEBFileLibrary. O nome do método será o nome seguido da extensão do arquivo com a primeira letra da extensão em maiúsculo (Ex: arquivoTxt).

8.5.2 Exemplos

'Adicionando uma imagem em WEBFileLibrary'

```
WEBFileLibrary add: 'caminho/da/imagem.png'.
```

'Usando a imagem adicionada'

```
iImage := WEBImage new initialize: WEBFileLibrary / 'imagem.png'.
```

8.6 WEBGodHome

Classe que é o componente principal da aplicação GOD. As demais aplicações são definidas como links a partir desta página principal.

8.6.1 Atributos

- **MainArea:** - atributo estático que define o componente que será renderizado na área principal do GOD.

8.6.2 Métodos

- **renderContentOn:** - chama os demais métodos de renderização da página principal do GOD.
- **renderFooterOn:** - renderiza o rodapé.
- **renderHeaderOn:** - renderiza o cabeçalho do GOD, que funciona como um menu para chamar as aplicações.
- **renderMainOn:** - renderiza a área principal, na qual as aplicações são renderizadas.

8.7 WEBHomeView

Classe que representa um componente que serve como uma página inicial para a área principal do projeto GOD.

8.8 WEBPageFetcher

Classe que realiza a captura do conteúdo HTML de páginas.

8.8.1 Métodos

- **fetch: anURL** - retorna o string com o HTML da URL passada como parâmetro.

8.8.2 Exemplos

```
'Obtendo o html de uma url'
```

```
WEBPageFetcher fetch:'url'.
```

8.9 Usando o GODWeb

Para usar o **GODWeb** é necessário criar uma classe (controladora) para fazer a renderização das páginas a serem criadas e controlar o fluxo de envio/recebimento dos dados a serem processados. A classe controladora deve receber em sua inicialização o componente que representa a página que será renderizada. Essa classe controladora deve também conter atributos que representem os elementos de **GODWeb** e que irão armazenar as informações preenchidas em um formulário para serem processadas.

Essa classe controladora deve possuir um método **render**, que vai conter a página inicial da aplicação. Essa classe pode ser usada para criar métodos de renderização para diversas páginas que existam para a aplicação. Para cada página da aplicação deve haver uma classe componente (sub-classe de WAComponent). Essa classe deve conter o método **renderContentOn:**. Esse método é chamado pelo próprio seaside para renderizar a página. Dentro do método deve-se instanciar a classe controladora.

Um componente que representa a página inicial da aplicação deve conter um método **initialize** para registrar a aplicação no servidor web (conhecido como comanche). O método **renderContentOn:** desse componente inicial deve instanciar a classe controladora passando-se como parâmetro para permitir que a classe controladora chame outros componentes. Portanto, a classe controladora deve ter um atributo que recebe esse componente inicial. No **renderContentOn:** deve-se também chamar o método **render** da classe controladora.

A classe controladora deve conter um método chamado **save**, que vai receber a instância da própria classe controladora com os valores preenchidos do formulário. O método **save** fica responsável então por chamar o fluxo que vai processar os dados, assim como pela chamada do componente que irá renderizar o resultado do processamento.

A classe controladora deve chamar o método **callback:**, da classe **WEBSubmitButton** pertencente ao formulário, passando ela mesma (a classe controladora) para permitir que **WEBSubmitButton** chame o seu método **save**.

No Seaside, os dados de um formulário são passados como um objeto com seus atributos modificados, e não via GET ou POST como tradicionalmente é feito na maioria dos frameworks web. Por isso a necessidade de passar o objeto da classe controladora para o **WEBSubmitButton**. Nos exemplos a seguir são apresentados exemplos dos métodos **save** de uma classe controladora e **renderContentOn:** de uma classe componente.

8.9.1 Exemplos

'Código do método renderContentOn: da classe componente que representa uma página que contém um formulário. GODApp eh o nome da classe controladora'

```
|page|
page := GODApp new initialize: self; render.
page render: html.
```

'Código do método save de uma classe controladora. O atributo component eh a classe que contém o formulário, que por sua vez chama o componente WEBResponse que irá renderizar o resultado do processamento. A classe controladora eh passada como parâmetro para que WEBResponse obtenha os dados preenchidos no formulário'

```
component call: (WEBResponse new object: self).
```

'Código do método estatico initialize de uma classe componente que representa a página inicial de uma aplicação'

```
WAAdmin register: self asApplicationAt: 'webapplication'
```

9 Planilhas (GODSpreadsheet)

O grupo de Spreadsheet preocupou-se, principalmente, em prover funcionalidades de entrada e saída de planilhas, mas também implementou alguns métodos que auxiliem a manipulação das mesmas. Em caso de dúvida, entrar em contato com Diogo Haruki (*haruki arroba ime ponto usp ponto br*)

9.1 Estrutura

Nesta seção, explicaremos a organização das principais classes do GODSPREADSHEET. Quase todos os objetos das classes apresentadas aqui possuem uma referência para seu pai (*parent*)¹.

SSCELL	Classe que corresponde a uma célula da planilha. Assumimos que a célula tem sempre um conteúdo texto (string), e consideramos como célula vazia aquelas que possuem value = nil .
SSRow	Classe que corresponde a uma linha da tabela. Cada linha contém uma coleção de células, que foi mantida encapsulada. Dessa forma, podemos garantir a coerencia entre os elementos dentro dessa lista e a referência dos pais desses elementos.
SSSheet	Classe que corresponde a uma tabela. Uma tabela é formada por uma coleção de linhas (SSRow), deve possuir um nome e, assim como em SSRow, a coleção de linhas é encapsulada para garantir a coesão.
SSpreadsheetData	Classe que corresponde a uma coleção de tabelas. Cada tabela é indexada numericamente, mas há a possibilidade de acessá-las a partir de seu nome. A coleção de tabelas (SSSheet) também é encapsulada, para manter a coesão entre os diversos objetos.

9.1.1 Indexação em todos os níveis

Uma breve explicação dos modos de acesso aos filhos de cada classe.

Em SSSpreadsheetData, as SSSheets estão indexadas numericamente, mas podem ser acessadas por seu nome.

Em SSSheet, as SSRows estão indexadas numericamente, e seu acesso se dá unicamente pelo número da linha.

Em SSRow, as SSCells estão indexadas numericamente, mas podem ser acessadas pelo nome da coluna ('A', 'B', ..., 'Z', 'AA', ...).

9.2 How to do

Nesta seção, vamos tentar mostrar o que pode ser feito² com o módulo e um modo de fazê-lo.

¹Com exceção apenas da SSSpreadsheetData, que não tem pai

²coisas básicas que podem ser feitas

9.2.1 Abrir arquivos

A abertura de arquivos pelo módulo GODSPREADSHEET é algo bem simples. Basta utilizar o método de classe `fromFile:` da classe `SSSPREADSHEETDATA`.

Os formatos de arquivo suportados atualmente são: `csv`, `ods`, `xlsx`.

```
csvSSData := SSSpreadsheetData fromFile: 'path/to/file.csv'.
odsSSData := SSSpreadsheetData fromFile: 'path/to/file.ods'.
xlsxSSData := SSSpreadsheetData fromFile: 'path/to/file.xlsx'.
```

9.2.2 Salvar arquivos

Salvar arquivos também é tarefa fácil para o módulo GODSPREADSHEET. Para isto, basta usar o método de instância `toFile:` de um objeto de `SSSPREADSHEETDATA`. Dessa forma, suportamos saída para arquivo `ods`.

```
ssData toFile: 'path/to/file.ods'.
```

Também podemos exportar uma `SSSHEET` para `csv`. Isso se deve ao fato de `csv` guardar uma planilha simples, e não uma coleção de planilhas como os outros formatos. Para isso, podemos usar o método de instância `exportToCSV:` de um objeto de `SSSHEET`.

```
sheet := ssData getSheet: 1.
sheet exportToCSV: 'path/to/file.csv'.
```

9.2.3 Criar uma `SSSpreadsheetData` e populá-la

Caso seja necessário, também é possível criar uma `SSSPREADSHEETDATA` e populá-la manualmente de forma bem fácil. Primeiramente, para criar uma planilha (um objeto `SSSHEET`):

```
ssdata := SSSpreadsheetData new.
sheet := ssdata createSheetWithName: 'sheet'.
```

Depois de criada podemos popular a planilha de vários jeitos. O mais simples é criando célula por célula diretamente:

```
cell1 := sheet createCellAtRow: 1 atColumnIndex: 1 withValue: 'valor'.
cell2 := sheet createCellAtRow: 1 atColumn: 'B' withValue: 'valor2'.
```

Mas também podemos criar uma `SSRow`, e definir as células pela linha:

```
row := sheet createRow.
row := sheet createRowAtIndex: 3.
row createCellAtColumn: 'A' withValue: 'valor'.
row createCellAtColumnIndex: 2 withValue: 'valor2'.
```

Ou também podemos popular uma `SSSHEET` diretamente com valores de um tabela `smalltalk`, que é uma coleção de *linhas*, onde cada *linha* é uma coleção com os valores das suas células. Também é possível popular a planilha com um dicionário `smalltalk` - nesse caso, cada par (`chave, valor`) do dicionário vira uma linha da planilha, com as chaves na coluna “A” e os valores na coluna “B”.

```
sheet fillFromCollection: collection.
sheet fillFromDictionary: dictionary.
```

9.2.4 Executar alguma ação em todas as células da planilha

As classes SSSpreadsheetData, SSsheet e SSRow contém coleções das classes seguintes, mas cada uma encapsula essa coleção para manter coesão.

Porém, além das classes permitirem acesso de algum elemento específico dessas coleções por mensagens específicas de cada classe, elas também permitem que o usuário itere pelas coleções usando a mensagem `do:` do objeto.

Partindo de um SSSpreadsheetData é trivial iterar por todas células de uma planilha:

```
sheet := ssData getSheet: 1.  
sheet do: [ :row |  
  row do: [ :cell |  
    Transcript show: (cell value).  
  ].  
].
```

9.2.5 Diferentes modos de acessar uma célula

Temos acesso às células em diferentes níveis. Tanto em SSsheet como em SSRow

```
cell1 := sheet getCellAtRow: 1 atColumn: 'A'  
cell2 := sheet getCellAtRow: 1 atColumnIndex: 3  
  
row := sheet.getRow: 6  
cell3 := row getCell: 'C'  
cell4 := row getCellAtIndex: 2
```

10 Processadores (GODProcessors)

O módulo **GODProcessors** realiza o processamento de coleções de dados. Possui duas classes principais, a **PCSStatisticsCalculator** que realiza os principais cálculos estatísticos e a **PCSTagger** que realiza classificação de texto através de um método estatístico de recuperação de informação conhecido como tf-idf. Possui ainda duas classes auxiliares, a **PCSPreprocessor** que realiza tratamento de strings e a **PCSGODTagger** uma fachada para objetos **GODData**.

Em caso de dúvidas, entre em contato com *Thiago: tdsimao [at] ime.usp.br*.

10.1 PCSStatisticsCalculator

Esta classe calcula medidas estatísticas de tendência central e dispersão para um determinada coleção. Possui ainda um método para contagem de palavras.

10.1.1 Métodos

Todos os cálculos dessa classe podem ser realizados de duas formas: com ou sem bloco. A chamada sem bloco realiza os cálculos sobre o valores absolutos da coleção, enquanto a chamada com bloco executa os cálculos sobre o resultado da execução do bloco nos elementos da coleção.

Os cálculos que a classe realiza são:

- **average**: média
- **median**: mediana
- **std**: desvio padrão
- **var**: variância

Além dos cálculos estatísticos, possui um método para contagem de palavras

- **countIn: aString occurrencesOf: aWord**: conta o número de ocorrências da palavra **aWord** na string **aString**.

10.1.2 Exemplos

Para utilizar os métodos sobre uma coleção de números.

```
pcs := PCSStatisticsCalculator new.  
pcs average: aCollection.  
pcs median: aCollection.  
pcs std: aCollection.  
pcs var: aCollection.
```

Pode-se utilizar os métodos sobre os atributos de coleção uma coleção de objetos, passando um bloco como parâmetro.

```

pcs := PCSStatisticsCalculator new.
pcs average: aCollection key: [ :x | x width].
pcs median: aCollection key: [ :x | x width].
pcs std: aCollection key: [ :x | x width].
pcs var: aCollection key: [ :x | x width].

```

10.2 PCSTagger

Essa classe utiliza uma técnica estatística de recuperação de informação conhecida como **tf-idf** (term frequency-inverse document frequency). Esse método de treinamento não supervisionado recebe uma coleção de *strings* para treinamento e retorna os termos mais relevantes em relação a toda a coleção. Detalhes sobre essa técnica podem ser encontrados no livro Introduction to Information Retrieval³

10.2.1 Variáveis de Instância

- **dictIdf**: Dictionary<idf> – um dicionário de *idf*(raridade do termo na coleção)
- **maxTf**: float [0..1] – frequência máxima dos termos da string a ser considerada.
- **minTf**: float [0..1] – frequência mínima dos termos da string a ser considerada.
- **minRelevance**: float [0..1] – define o menor *tf_idf* (relevância) a ser considerada.

10.2.2 Métodos

- **createDictIdf: aStringCollection**: cria o dictIdf.
- **getMoreRelevantsOf: aString**: recupera os termos mais relevantes de aString.

10.2.3 Exemplos

Para utilizar os métodos sobre uma coleção de *strings*.

```

pcsTagger := PCSTagger new.
pcsTagger createDictIdf: aStringCollection.
bag := Bag new.
bag := pcsTagger getMoreRelevantsOf: aString.

```

10.3 PCSGODTagger

Essa classe é uma fachada da classe PCSTagger para objetos GDDData. A classe PCSGODTaggerExample mostra um exemplo completo de uso dessa classe.

É importante notar que é possível alterar o PCSTagger segundo necessário.

³<http://nlp.stanford.edu/IR-book/>

10.3.1 Variáveis de Instância

- **tagger:** PCSTagger – objeto da classe PCSTagger que treina com uma coleção de strings recuperadas de uma coleção de GODData e recupera os principais termos dessa *String*.

10.3.2 Métodos

- **training:** aGODDataCollection: realiza o treinamento com o atributo content dos objetos de aGODDataCollection.
- **addTagsTo:** aGODData: define tags para o atributo tags de aGODData
- **tagCollection:** aGODDataCollection: atalho para realizar o treinamento sobre uma coleção e em seguida adicionar tags a todos os objetos da mesma.

10.3.3 Exemplos

Para adicionar *tags* a um objeto GODData.

```
pcstagger := PCSGODTagger new.  
pcstagger training: aGODDataCollection.  
pcstagger addTagsTo: aGODData.
```

Para adicionar *tags* a todos os elementos de uma coleção de GODData é possível chamar o método *tagCollection*.

```
pcstagger := PCSGODTagger new.  
pcstagger tagCollection: aGODDataCollection.
```

10.4 PCSPreprocessor

Essa classe realiza operações comuns de pré-processamento de *strings*. Para utilizar essa classe você deve configura-la definindo os valores de suas variáveis 10.4.1 através de seus métodos de acesso e em seguida usar o método *preprocess*: aString para recuperar uma coleção de *tokens* da string *aString*

10.4.1 Variáveis de Instância

- **fileType:** String ∈ {‘TXT’, ‘HTML’} – define o tipo de arquivo de entrada.
- **punctuation:** String – string com todos os caracteres de pontuação que serão removidos, ex: ‘,.!?’.
- **stopWords:** Set<string>– um conjunto com palavras comuns da língua que serão ignoradas.

10.4.2 Métodos

Além do método principal `preprocess`: As operações que essa classe realiza são:

- **treatType**: realiza tratamento relativo ao tipo de arquivo, exemplo para arquivos HTML remove as *tags* HTML.
- **removeStopwords**: remove as palavras comuns de uma *string*.
- **tokenizer**: quebra a *string* em uma coleção de *tokens*.

10.4.3 Exemplos

```
preprocessor := PCSPreprocessor new.  
aBag := preprocessor preprocess: aString.
```

11 Redes sociais (GODSocialNetIO)

11.1 Team

Eduardo Alexandre, Aline Borges, Leonardo Haddad, Thiago Araujo

11.2 Description

GODSocialNetIO allows others modules to communicate with the Facebook and Twitter servers. It does that by implementing both public APIs. The module is divided in two main parts: the fetchers classes and the GODData classes. The fetchers classes are responsible for the communication with the APIs. The GODData classes receive the JSON response from a request returned by a fetcher and transform it into subclasses of GODData.

It's important to mention that both Facebook and Twitter APIs need a client id and password to work, this can be done creating a specific type of user in the developer section of both APIs. Last but not least, we have classes responsible for testing all methods within the fetchers and GODDatas, keep in mind that we don't have control of the Facebook and Twitter servers, so some test may fail some times simply because the server response is not what was expected.

If you have questions, contact us: thd.araujo@gmail.com.

11.3 Classes of Fetchers

11.3.1 SNETFetcher

SNETFetcher is a abstract class that contains the common parts used by SNETFacebookFetcher and SNETTwitterFetcher. The methods are described below.

- **private** - This abstract method sends some HTTP request to some url, it will be specialized in SNETFacebookFetcher and SNETTwitterFetcher.
- **initialize** - This is the default method to initialize the class.
- **connection** - This abstract method connects to some server.

11.3.2 SNETFacebookFetcher

SNETFacebookFetcher is a class that controls all the access to the Facebook API. It sends requests to the facebook server and returns the result as a SNETFacebookUsersData object, SNETFacebookPostsData, SNETFacebookPagesData, SNETFacebookPageDescriptionData, SNETFacebookGroupsData, SNETFacebookGroupDescriptionData and SNETFacebookEventsData. The methods are described below.

- **sendRequest** - sends some HTTP request to Facebook API.
- **byId: id** - is a helper to some methods that search in the Facebook API using some id.

- **initialize** - This is the default method to initialize the class, it will initialize all the class variables.
- **connection** - connects to Facebook API.
- **events: event** - searches for events with the parameter name.
- **events: event since: since** - searches for events with the parameter name since some date.
- **events: event since: since until: until** - searches for events with the parameter name since some date until another date.
- **groupDescription: groupId** - gets group information by id.
- **groups: group** - searches for groups with the parameter name.
- **pageDescription: pageId** - gets page information by id.
- **pages: page** - searches for pages with the parameter name.
- **posts: user** - gets posts of a user.
- **posts: user since: since** - gets posts of a user since some date.
- **posts: user since: since until: until** - gets posts of a user since some date until another date.
- **users: user** - searches for users with the parameter name.

11.3.3 SNETTwitterFetcher

SNETTwitterFetcher is a class that control all the access to the Twitter API. It send requests to the twitter server and return the result as data class SNETTwitterTweetsData. The methods are described below.

- **sendRequest: url** - sends some HTTP request to Twitter API.
- **sendRequest: url method: method** - sends some HTTP request with either GET or POST method to Twitter API.
- **initialize** - This is the default method to initialize the class, it will initialize all the class and instance variables.
- **connection** - connects to Twitter API.
- **posts: user** - gets posts of a user.
- **posts: user maximum: max** - gets posts of a user specifying the maximum amount of posts.
- **tweets: text** - gets tweets from text.

- **tweets: text maximum: max** - gets tweets from text specifying the maximum amount of tweets.
- **tweets: text since: date maximum: max** - gets tweets from text since date specifying the maximum amount of tweets.
- **tweets: text since: initDate until: endDate maximum: max** - gets tweets from text since date until another date specifying the maximum amount of tweets.
- **tweets: text until: date** - gets tweets from text until date.
- **tweets: text until: date maximum: max** - gets tweets from text until date specifying the maximum amount of tweets.
- **tweetsFrom: text user: user** - gets tweets from text from user.
- **tweetsFrom: text user: user maximum: max** - gets tweets from text from user specifying the maximum amount of tweets.
- **tweetsFrom: text user: user since: date** - gets tweets from text from user since date.
- **tweetsFrom: text user: user since: date maximum: max** - gets tweets from text from user since date specifying the maximum amount of tweets.
- **tweetsFrom: text user: user since: initDate until: endDate maximum: max** - gets tweets from text from user since date until another date specifying the maximum amount of tweets.
- **tweetsFrom: text user: user until: date** - gets tweets from text from user until date.
- **tweetsFrom: text user: user until: date maximum: max** - gets tweets from text from user until date specifying the maximum amount of tweets.
- **language: newLanguage** - sets the language of the tweets.

11.4 Classes of GODDatas

11.4.1 SNETTwitterTweetsData

SNETTwitterTweetsData is a class that contains a list of SNETTwitterTweet. The methods are:

- **addTweet: json** - adds a new tweet from a json to the collection.
- **initialize** - This is the default method to initialize the class.
- **size** - shows the number of tweets in the collection.
- **tweetAt: position** - gets a tweet at a position in the collection.
- **tweets** - gets the list of tweets.

11.4.2 SNETTwitterTweet

SNETTwitterTweet is a class that contains a Twitter tweet. There are getters and setters for the following variables:

- **favorited: number** - number of favorited of tweet.
- **id: idNumber** - tweet id.
- **message: text** - tweet message.
- **retweets: number** - number of retweets.

11.4.3 SNETFacebookPostsData

SNETFacebookPostsData is a class that contains a list of SNETFacebookPost. The methods are:

- **addPost: json** - adds a new post from a json to the collection.
- **initialize** - This is the default method to initialize the class.
- **size** - shows how much posts the collection contains.
- **postAt: position** - gets a post at a position in the collection.
- **posts** - gets the list of posts.

11.4.4 SNETFacebookPost

SNETFacebookPost is a class that contains a Facebook post. There are getters and setters for the following variables:

- **addComment: json** - adds a new comment to the collection.
- **caption: text** - post caption.
- **description: text** - post description.
- **likes: number** - number of likes of post.
- **message: text** - post message.
- **picture: url** - picture url from post.
- **shares: number** - post shares.
- **type: postType** - type of post.

11.4.5 SNETFacebookComment

SNETFacebookComment is a class that contains a Facebook comment. There are getters and setters for the following variables:

- **id:** `idNumber` - comment id.
- **likes:** `number` - number of likes of a comment.
- **message:** `text` - comment message.

11.4.6 SNETFacebookPagesData

SNETFacebookPagesData is a class that contains a list of SNETFacebookPage. The methods are:

- **addPage:** `json` - adds a new page from a json to the collection.
- **initialize** - This is the default method to initialize the class.
- **size** - shows the number of pages in the collection.
- **pageAt:** `position` - gets a page at a position in the collection.
- **pages** - gets the list of pages.

11.4.7 SNETFacebookPage

SNETFacebookPage is a class that contains a Facebook page. There are getters and setters for the following variables:

- **category:** `pageCategory` - page category.
- **id:** `idNumber` - page id.
- **name:** `pageName` - page name.

11.4.8 SNETFacebookPageDescriptionData

SNETFacebookPageDescriptionData is a class that contains the description of a certain Facebook page. There are getters and setters for the following variables:

- **category:** `pageCategory` - page category.
- **description:** `text` - page description.
- **id:** `idNumber` - page id.
- **name:** `pageName` - page name.
- **likes:** `number` - likes of a page.

11.4.9 SNETFacebookGroupsData

SNETFacebookGroupsData is a class that contains a list of SNETFacebookGroup. The methods are:

- **addGroup: json** - adds a new group from a json to the collection.
- **initialize** - This is the default method to initialize the class.
- **size** - shows the number of groups in the collection.
- **groupAt: position** - gets a group at a position in the collection.
- **groups** - gets the list of groups.

11.4.10 SNETFacebookGroup

SNETFacebookGroup is a class that contains a Facebook group. There are getters and setters for the following variables:

- **id: idNumber** - group id.
- **name: groupName** - group name.

11.4.11 SNETFacebookGroupDescriptionData

SNETFacebookGroupDescriptionData is a class that contains the description of a certain Facebook group. There are getters and setters for the following variables:

- **description: text** - group description.
- **id: idNumber** - group id.
- **name: groupName** - group name.

11.4.12 SNETFacebookEventsData

SNETFacebookEventsData is a class that contains a list of SNETFacebookEvent. The methods are:

- **addEvent: json** - adds a new event from a json to the collection.
- **initialize** - This is the default method to initialize the class.
- **size** - shows the number of events in the collection.
- **eventAt: position** - gets a event at a position in the collection.
- **events** - gets the list of events.

11.4.13 SNETFacebookEvent

SNETFacebookEvent is a class that contains a Facebook event. There are getters and setters for the following variables:

- **location:** `eventLocation` - event location.
- **id:** `idNumber` - event id.
- **name:** `eventName` - event name.

11.4.14 SNETFacebookUserData

SNETFacebookUsersData is a class that contains a list of SNETFacebookUser. The methods are:

- **initialize** - This is the default method to initialize the class.
- **size** - show the number of users in the collection.
- **userAt: position** - gets a user at a position in the collection.
- **users** - gets the list of users.

11.4.15 Exemplos

```
'Initializing a facebook fetcher'  
faceFetcher := SNETFacebookFetcher new.  
faceFetcher connect.  
  
'fetching by group'  
snetGroupData:= faceFetcher groups: 'rock'.  
  
'fetching by posts'  
snetPostData := faceFetcher posts: 'running'.  
  
'Initializing and getting tweets.'  
twFetcher := SNETTwitterFetcher new.  
twFetcher connect.  
twData:=twFetcher tweets: 'Obama'.
```

A more detailed information about this module can be found in the Appendix section.

12 Textos (GODTextIO)

O `GODTextIO` é o módulo integrante do GOD (Grande Organizador de Dados) que se encarrega da conversão do conteúdo arquivos texto para um objeto do tipo `GODData` e vice-versa. Atualmente, o módulo lida com arquivos `odt`, `pdf`, `rtf` e `txt`. A seguir, daremos uma descrição sobre os pré-requisitos para seu funcionamento e, logo depois, a composição do módulo, além de mostrar como lidamos com os diversos tipos de arquivos.

12.1 Equipe

João Paulo Camargo, John Gardenghi⁴, Marcello Oliveira, José Eurípedes.

12.2 Pré-requisitos

Para que este módulo funcione corretamente, é necessário

- Ter o pacote `OSProcess` instalado no Squeak. O `OSProcess` é usado para executar comandos no terminal do SO. Ele está incluído nos pacotes dependências do repositório do GOD.
- Ter o `pdflatex` e o `abiword` devidamente instalados e funcionando por chamada no terminal.

12.3 Classes

`TXTIOConverter`

É a interface com o GOD. Qualquer outro módulo que deseje usar alguma das nossas funcionalidades, deve fazer chamadas a métodos desta classe. Ela é apenas a porta de entrada do nosso módulo, portanto, as funcionalidades que ela implementa não estão codificadas nesta classe, mas apenas são feitas chamadas a métodos competentes para cada funcionalidade.

Esta classe possui apenas um atributo, que é o `types`. É um dicionário cujas chaves são os tipos suportados pelo módulo, a saber `odt`, `pdf`, `rtf` e `txt`, e os valores são as classes associadas a cada tipo de arquivo. Por outro lado, temos dez métodos. A seguir, descrevemos os que interessam a outros módulos do GOD. Logo depois, mostramos os demais métodos da classe, usados para finalidades específicas da própria classe.

1. `readFromFile: aPath type: fileType` é responsável por ler um arquivo cujo caminho é passado no argumento `aPath` e cujo tipo é passado no argumento `fileType`. Os tipos de arquivos válidos são `odt`, `pdf`, `rtf` e `txt`. Devolve um objeto do tipo `GODData`. Sua variação é o método `readFromFile: aPath`, em que o tipo do arquivo é extraído do caminho passado.
2. `write: godData toFile: aPath type: fileType append: app overwrite: ovwr` é responsável por escrever o conteúdo do objeto `godData` do tipo `GODData` em um arquivo cujo caminho é passado pelo argumento `aPath` do tipo `fileType`. Os argumentos (a) `append` e (b) `overwrite` são valores lógicos que determinam, caso o arquivo a ser escrito já exista, se (a) o novo conteúdo deve ser adicionado ao final do arquivo já existente e (b) o arquivo deve ser sobreescrito. Suas variações são quatro métodos, variando apenas a combinação dos argumentos, que são

⁴john@ime.usp.br

- `write: godData toFile: aPath,`
- `write: godData toFile: aPath type: fileType,`
- `write: godData toFile: aPath type: fileType append: app,`
- `write: godData toFile: aPath type: fileType overwrite: ovwr.`

Além desses sete métodos, temos

1. `getFileTypeFromPath: aPath` extrai o tipo do arquivo a partir do caminho passado como argumento em `aPath` e o retorna.
2. `getPossibleTypes` retorna uma lista com as chaves do dicionário `types`.
3. `initialize` inicializa o dicionário `types`.

TXTMainConverter

É a classe mãe que determina os métodos que cada classe filha deve implementar de forma ler ou escrever arquivos de tipos específicos. Possui sete atributos:

1. `append` determina se o conteúdo novo deve ser adicionado ao final de um arquivo existente (lógico),
2. `author` contém o autor do conteúdo (texto),
3. `contents` contém o conteúdo do arquivo (texto),
4. `date` contém a data (texto),
5. `overwrite` determina se o arquivo deve ser sobreescrito (lógico),
6. `path` contém o caminho do arquivo (texto),
7. `title` contém o título (texto).

Contém dois métodos principais:

1. `readFile` lê um arquivo. Nesta classe, possui uma implementação genérica, em que converte o tipo do arquivo para `odt`, usando o `abiword`, e lê este arquivo usando o leitor implementado na classe `TXTIOODtConverter`. Todavia, é sobreescrito pelas classes filhas `TXTIOODtConverter` e `TXTIOTextConverter`. O conteúdo lido do arquivo é salvo no atributo `content` na classe, e este método retorna `true`.
2. `writeFile` é apenas um método abstrato.

Além disso, temos `convertFileType`, que faz a conversão do tipo do arquivo usando o `abiword`, e os métodos de acesso aos atributos.

As classes que lidam com os tipos suportados são:

1. **TXTIOOdtConverter** lida com arquivos `odt`. Explora o padrão *OpenDocument Format (odf)*. Esse formato é uma forma compactada de alguns arquivos, dos quais nos interessam o arquivo `content.xml`. Na classe `TXTOdtConverter`, o método `readFile`, se encarrega primeiro de chamar o `extractTextXML` para extrair o conteúdo interessante (`tags office:*`) desse arquivo, primeiro descompactando o formato `odt` (através da classe `ZipArchive` do Squeak) e depois lendo como uma *stream*. Depois o `readFile` extrai a informação útil dessa *stream*, verificando as `tags`. O `writeFile`, cria um diretório temporário e os arquivos que compõem o `odt`, e escreve no `content.xml` o conteúdo (conforme o formato `xml`), compacta para `odt` e o move para o caminho desejado pelo usuário.
2. **TXTIOPdfConventer** lida com arquivos `pdf`. Escreve arquivos usando o `pdflatex` e lê usando o conversor do `abiword` implementado na classe mãe.
3. **TXTIORtfConverter** lida com arquivos `rtf`. Para a escrita, criamos algumas tags básicas do formato e escrevemos, através da `TXTTextConverter`, em um arquivo, no formato `rtf`, todavia os lê usando o conversor `abiword` implementado na classe mãe.
4. **TXTIOTextConverter** lida com arquivos `txt`. Escreve e lê os arquivos usando a classe `FileStream` do Squeak Smalltalk.

12.4 Exemplos

```
'Criando um arquivo texto a partir de um GODData'
txtio:=TXTIOConverter new.
txtio write: godData toFile:'path/to/file.txt'.

'Lendo um arquuivo ODT e adicionando a um GODData'
gOdt:=txtio readFile:'path/to/file.odt'.
```

13 Agregador de informações acadêmicas (GODAcademics)

Grupo: Ígor Bonadio, Renato Cordeiro, Ruan Costa

Contato: ibonadio@ime.usp.br

GOD Academics é um agregador de informações acadêmicas. A partir de um perfil do Google Scholar, esta aplicação constroi um relatório resumindo as informações obtidas de diversas fontes.

Atualmente GOD Academics apenas utiliza como fonte de informação o Google Scholar e o CAPES-Qualis.

13.1 Classe ACADPaper

Representa um artigo de um pesquisador.

13.1.1 Atributos de instância

- name: nome do artigo.
- coauthors: string contendo os nomes de todos os autores.
- year: ano em que o artigo foi publicado.
- journal: é o nome do periódico onde o artigo foi publicado.
- impactFactor: é o fator de impacto (estrato) do journal no qual o artigo foi publicado. Essa é uma medida do sistema webQualis da capes.
- numberOfCitations: é o número de citações do artigo.

13.1.2 Métodos de instância

- hasAttribute: aTag in: aHtmlPaper - Verifica se a informação representada por aTag existe em aHtmlPaper
- initializeFromHTML: aHtmlPaper - Monta um paper a partir de uma string contendo código html. Não deve ser usado diretamente, veja Researcher>initializeFromProfileURL.
- loadAttribute: aTag from: aHtmlPaper - Retorna o conteúdo da aTag a partir de um html.
- loadAttributesFrom: aHtmlPaper - Carrega os atributos de um paper a partir de um html.
- loadImpactFactorFrom: journal - Carrega o fator de impacto do journal passado
- numberOfCitations - Retorna o número de citações do artigo.

13.2 Classe ACADQualis

Representa a lista qualis de periódicos x fator de impacto. É um singleton. Não deve ser instanciado com new, mas sim com o método de classe singleton.

13.2.1 Atributos de classe

- uniqueInstance: guarda a única instância da classe.

13.2.2 Atributos de instância

- qualis: é um dicionário. As chaves são os periódicos e o valor é o fator de impacto do periódico.

13.2.3 Métodos de classe

- clear - Limpa o atributo de classe que guarda a instancia da classe. Faz isso atribuindo null.
- singleton - Cria uma instância da classe e a retorna.

13.2.4 Métodos de instância

- initialize - Carrega o hash qualis com a lista armazenada em listOfQualis
- journalIsSomethingLike: aJournal - Dado o nome de um periódico, procura na no hash um periódico com nome parecido. Retorna a primeira chave que dê match em um dos prefixos do nome do periódico procurado. Se não houver match, é retornado unknown.
- listOfQualis - String com a última lista lançada pela capes, com o nome dos periódicos e seus respectivos fatores de impacto
- load: aQualisText - Carrega um texto qualis (que esteja no mesmo formato de listOfQualis) no dicionário qualis.
- qualisOf: aJournal - Dado o nome de um periódico, retorna seu fator de impacto. Se não existir no dicionário, unknown é retornado.

13.3 Classe ACADResearcher

Representa um pesquisador.

13.3.1 Atributos de instância

- name: nome do pesquisador
- job: profissão do pesquisador
- papers: lista de artigos
- valid: variável booleana que indica se este é um objeto bem formado ou não.

13.3.2 Métodos de instância

- barGraphOfPapersByImpact - Retorna um gráfico com os fatores de impacto no eixo x, e a quantidade de artigos com um dado fator de impacto no eixo y.
- forEachPaperDo: aBlock - Executa o bloco passado em toda a lista de artigos.
- hasAttribute: aTag in: aHtmlPage - Verifica se a informação representada por aTag existe em aHtmlPage.
- initialize - Aloca memória para a lista de artigos.
- initializeFromHTMLPage: aHtmlPage - Monta um pesquisador extraindo as informações de aHtmlPage.
- initializeFromProfileURL: aProfileURL - Similar à initializeFromHTMLPage, mas pega a página passada em aProfileURL. Esta URL tem de ser um perfil do Google Scholar. Utilizamos um proxy para que o google scholar não bloqueie nosso acesso, pois nosso programa age como um robô
- loadAttribute: aTag from: aHtmlPage - Se a informação representada por aTag existir em aHtmlPage, retornamos ela. Caso contrário, retornamos um GODData vazio.
- loadAttributesFrom: aHtmlPage - Carrega os atributos do pesquisador a partir de aHtmlPage.
- loadPapersFrom: aHtmlPage - Carrega os artigos do pesquisador a partir de aHtmlPage.
- papersByImpactDictionary - Retorna um dicionário, onde as chaves são todos os possíveis valores de fatores de impacto e o valor é a quantidade de artigos que o pesquisador tem com cada fator de impacto.
- totalNumberOfCitations - Retorna o número total de citações ao pesquisador, isto é, a soma das citações de todos os seus artigos.

13.4 Classe ACADTag

Representa uma o início e o fim de uma TAG. Entre essas marcas encontra-se o conteúdo necessário para carregar as classes ACADResearchers e ACADPapers.

13.4.1 Atributos de instância

- star: marca de início
- end: marca de fim

13.5 Classe ACADTagPool

Representa a coleção de ACADTags necessárias para a aplicação GODAcademics. É um singleton. Não deve ser instanciado com new, mas sim com o método de classe singleton.

13.5.1 Atributos de classe

- uniqueInstance: guarda a única instância da classe.

13.5.2 Atributos de instância

- pool: um dicionário de ACADTags.
- filter: um FILTTextFilter

13.5.3 Métodos de classe

- clear - Limpa o atributo de classe que guarda a instancia da classe. Faz isso atribuindo null.
- singleton - Cria uma instância da classe e a retorna.

13.5.4 Métodos de instância

- get: aTagName - Retorna um ACADTag cujo nome é aTagName.
- getAttributes: aTagName from: aHTML - Utilizando a tag aTagName, retorna o valor encontrado em aHTML.
- initialize - Inicializa a instância.
- initializePaperTags - Adiciona ao dicionário pool todas as tags de artigos
- initializeResearcherTags - Adiciona ao dicionário pool todas as tags de pesquisadores.

13.6 Classe ACADAcademicsHome

Componente utilizado pelo Seaside para renderizar a página inicial da aplicação.

13.7 Classe ACADAcademicsProfile

Componente utilizado pelo Seaside para renderizar a página com o relatório final do pesquisador escolhido.

13.8 Classe ACADAcademics

Responsável por controlar o fluxo da aplicação GODAcademics.

13.8.1 Atributos de instância

- component: um ACADAcademicsHome ou ACADAcademicsProfile
- input: um textfield que contem a URL do perfil do Google Scholar de um pesquisador
- profileUrl: URL do perfil do Google Scholar de um pesquisador

13.8.2 Métodos de instância

- initialize: aComponent - Inicializa a instância.
- renderHome - Renderiza a página inicial
- renderProfile: researcher - Constrói o relatório do pesquisador
- save - Captura o valor atribuído ao formulário da página inicial.

13.9 Exemplos

```
'Iniciando um ACADResearcher a partir de um perfil do Google Scholar'  
researcher:= ACADResearcher new initialize.  
researcher initializeFromHTMLPage: 'http://scholar.google.com/citations?user=12SqSAsAAAAJ  
?hl=pt-BR&oi=ao'.  
  
'Gerando um grafico com os papers do pesquisador'  
researcher barGraphOfPapersByImpact.
```

14 Agregador de conferências (GOD's Call)

14.1 Equipe

Fabrício C. Machado⁵, Phablo F. S. Moura e Camila M. de Sousa.

14.2 Descrição geral

O GOD's Call é uma aplicação para o auxílio na decisão sobre qual conferência participar. Essa aplicação acessa diferentes fontes na internet (e.g. WikiCFP, ConfSearch e lista de classificação da CAPES-Qualis) e combina os dados dessas fontes para criar uma lista de conferências com informações como Deadline, Palavras-Chave, Área, Ratings, Conceito Qualis, permitindo que o usuário faça buscas nessa base de dados.

14.3 Tutorial de inicialização

14.3.1 Dependências

Na versão atual do God's Call, usamos especialmente os seguintes pacotes (e versões): `GODBases-lacm.31.mczi`, `GODCall-pfsm.92.mczi`, `GODCallTests-pfsm.36.mczi`, `GODFilter-yml.23.mczi`, `GODKernel-ER17.mczi`, `GODSpreadsheet-foa.29.mczi`, `GODWeb-has.21.mczi`.

Também usamos um pacote adicional: Regex (versão `damienpollet.17`).

14.3.2 Inicialização da aplicação

Na primeira vez que tentar abrir a página inicial do God's Call, o banco de dados estará vazio e a aplicação começará a rastrear as páginas da web. Essa etapa pode demorar algumas horas.

14.4 Implementação

14.4.1 Modelos do domínio

- **CCVenue** Essa classe modela as informações a respeito do local onde é realizada a conferência. Seus atributos de instância são: `place`, `city` e `country`.
- **GCRating** Nesta classe ficam informações a respeito da avaliação (ou classificação) de uma conferência. No atributo `qualis` é guardado o conceito Qualis-Capes e no atributo `hIndex` fica o Índice H da conferência.
- **GCIImportantDates** Representa as datas mais relevantes da conferência. No atributo `deadlineDate` é guardada a data limite para submissão de trabalhos. Os atributos `startDate` e `endDate` representam o período de realização (início e fim) da conferência.
- **GCConference** Essa classe modela uma conferência. Como principais atributos de instância, ela possui `acronym`, `name`, `url`, `identifier` e `content`. Claramente, os 3 primeiros atributos representam o acrônimo, o nome e a url da página da conferência. O atributo `identifier` identifica

⁵fabcm@ime.usp.br

unicamente uma conferência. Ele é gerado automaticamente a partir do acrônimo. No atributo `content` fica guardado o texto do “call for papers”. Além desses, a classe `GCConference` possui os atributos `venue` (um objeto da classe `GCVenue`), `rating` (um objeto da classe `GCRating`) e `importantDates` (um objeto da classe `GCIImportantDates`).

14.4.2 Crawling

Obtemos informações de três fontes: ConfSearch⁶, WikiCfp e lista de classificação CAPES-Qualis. Os processos de consultar e unir as informações é orquestrado pela classe `GCCrawler`. Essa é uma classe estática que consulta todas as fontes existentes segundo a lista retornada por `GCSOURCECREATOR»createAll`. Essas fontes são instâncias da classe `GCSOURCE`.

Cada instância de `GCSOURCE` contém uma referência para um arquivo de texto que contém um dicionário de categorias. A chave é o nome da categoria na fonte e o valor o nome da categoria no sistema GOD’s Call. Cada linha é um mapeamento, sendo o formato `chave=valor`. Linhas em branco resultam em uma quebra do formato e portanto possíveis erros na execução. Além do arquivo de texto, a instância possui uma referência para uma instância de um *crawler* que é capaz de buscar e transformar as conferências dessa fonte em uma `GCConference`.

Os *crawlers* tem um método comum definido por sua superclasse `GCSOURCECRAWLER»crawl:categories`, que recebe uma lista de categorias e devolve uma lista de conferências.

Atualmente temos implementado `GCCONFSEARCHCRAWLER`, `GCWIKICFP_CRAWLER` e `GCQUALIS_CRAWLER`. Os dois primeiros consultam os sites das fontes e transformam os resultados obtidos em conferências. Quanto a lista de classificação da Qualis, como ela é uma fonte que sofre poucas alterações decidimos por utilizar um arquivo csv. Esse arquivo foi adicionado ao repositório usando uma subclasse de `WAFILELIBRARY`, `GCFILELIBRARY` e está acessível pelo método `GCFILELIBRARY»qualisConferencesCSV`.

Para adicionar uma nova fonte, crie uma instância de `GCSOURCE` em `GCSOURCECREATOR` e retorne-a no método `createAll`. Você provavelmente terá que criar um novo *crawler* subclasse de `GCSOURCECRAWLER`.

14.4.3 Web Application

A aplicação possui 3 páginas web, que são responsabilidade das classes `GCFIRSTPAGE`, `GCSSECONDPAGE` e `GCTHIRDPAGE`. Estas classes delegam a renderização das páginas para os métodos `GODCALL»renderFirstPage`, `GODCALL»renderSecondPage` e `GCCONFERENCERESPONSE»renderThirdPage` que usam as classes e métodos disponibilizados pelo pacote GODWeb.

`GODCALL` representa uma sessão de usuário e é instanciado no momento que a página é acessada (veja `GCFIRSTPAGE»renderContentOn:`). No momento de sua inicialização, (veja `GODCALL»initialize:`) verifica se a base de dados está carregada e ativa o crawler, caso esteja vazia. Seus atributos armazenam os parâmetros relevantes para a sessão, em especial os parâmetros de busca que são preenchidos no formulário da primeira página.

`GCCONFERENCERESPONSE` é responsável pela renderização da terceira página, que é específica para uma conferência. É instanciado um objeto para cada conferência exibida na segunda página (veja

⁶Desde o dia 29/11 até a data desse documento o site www.confsearch.org não estava disponibilizando as conferências, todas as categorias apresenta uma lista vazia como resultado

`GODCall»renderSecondPage`) e possui um método `save`, que é ativado quando seu respectivo botão na segunda página é selecionado.

14.4.4 Testes

Implementamos 25 testes unitários que possuem cobertura de 83% do código. Praticamente todos os métodos que não possuem um teste são referentes à aplicação web.

Para o teste de alguns métodos recorremos a utilização de *mocks*. Esses são disponibilizados por meio de arquivos e permitem que métodos como o `GCConfSearchCrawler»parse:page` recebam uma página html sem a que a requisição à página web seja realizada.

Para testar o método `crawl` das fontes WikiCfp e ConfSearch substituímos a referência ao objeto `WEBPageFetcher` pelos objetos `FakeWikiCfpFetcher` e `FakeConfSearchFetcher`. Isso permite que os testes rodem sem internet e que não sejam afetados por possíveis erros no pacote GODWeb.

14.5 Exemplos

```
'Obtendo informacoes de conferencias do ConfSearch para software engineering'  
csCrawler:=GCConfSearchCrawler new.  
categCS:=Dictionary new.  
categCS at:'category' put:'software engineering'.  
csList:= csCrawler crawl: categCS.  
  
'Obtendo informacoes de conferencias do WikiCfp para computer science'  
wCrawler:=GCWikiCfpCrawler new.  
categW:=Dictionary new.  
categW at:'computer science' put:'computer science'.  
wList:= wCrawler crawl: categW.
```

15 Análise de sentimentos de consumo e político (GODSentimentAnalysis)

Grupo: Ana Luisa de Almeida Losnak, Arthur Branco Costa, Daniel Costa Bucher, Diego de Araújo Martinez Camarinha, Rafael Batista Carmo

Contato: analosnak@gmail.com

15.1 Introdução

Análise de Sentimento de Consumo: Aplicação na qual o usuário faz uma busca por determinado produto e o GOD devolve as estatísticas a respeito dele. Há dois formatos de saída: planilha ou gráfico. Nos resultados são mostrados quantas postagens nas redes sociais foram a favor, contra e neutras a respeito desse produto.

Análise de Sentimento Político: Nesta aplicação a busca consiste em dois campos: o nome de um candidato e seu partido político. É possível ainda procurar apenas pelo partido e ver as opiniões sobre ele de um modo geral. Também indica-se o formato de saída desejado (planilha ou gráfico).

A seguir descrevemos as classes do módulo, assim como os seus principais métodos.

15.2 SAInterface

Classe principal que chama todas as outras, deve ser a única utilizada por outros pacotes. Ou seja, é a única classe que seria pública.

analyse: as: é o método principal de toda a aplicação, o único método de instância dessa classe e que deve ser chamado pelo usuário do pacote.

Além do analyse, ela possui mais cinco métodos de classe, todos eles responsáveis pela criação dos dicionários de termos positivos e negativos:

15.2.1 Léxicos de Sentimento

Todas as palavras dos dicionários desta classe foram retiradas do **OpLexicon**, um léxico de sentimento para a língua portuguesa, do OntoLP, um portal de ontologias.

Após baixarmos esse léxico tivemos que tratá-lo para adequá-lo melhor ao GOD, fazendo as seguintes alterações:

- Remoção dos verbos: Os verbos dessa lista constavam apenas no infinitivo. Para sua utilização de forma eficiente, seria necessário incluir suas conjugações mais utilizadas.
- Português brasileiro: As palavras lá listadas estavam em português de Portugal, então tivemos que realizar modificações para o português brasileiro, tais como: género - gênero; projecto - projeto; tónico - tônico, etc.

15.3 SAFetcher

Devolve postagens relevantes da rede social escolhida (Twitter ou Facebook) que contenham o termo pesquisado, no caso um `tag`.

O método `getTweets` pega no máximo cem tweets (limitação da API) da classe SNETTwitterFetcher
O método `setSession` auxilia a correta implementação dos testes utilizando mock.

15.4 CSAApp

Estas classes são responsáveis por manter o conteúdo HTML da página Web da aplicação de Análise de Consumo.

`renderResponse`: este método faz a validação dos dados inseridos pelo usuário. Apenas verifica se o campo do produto não está vazio.

Os demais métodos de instância incluem: inicialização, renderização da tela do navegador e o `save` que retorna os valores de um formulário.

15.5 PSAAppl

Classes análogas às anteriores, porém tratam sobre a Análise de Sentimento Político. Os métodos adicionais da PSAAppl lidam com a validação dos dados retirados do TSE.

`validateCandidates` e `validateCandidates:using`: valida a entrada do usuário, verificando se o candidato está em alguma das listas de cargo (presidente, governador, senador, etc). Por último o método `validateParty`, que verifica se o partido está na lista de partidos válidos.

15.6 PSAPoliticalDataLoader

No pacote `accessor` encontram-se métodos que servem apenas para retornar os dicionários de candidatos por cargo, partidos e UFs. Já no pacote `initializer` localizam-se os métodos que buscam os candidatos por cargo e estado no site do TSE.

15.7 SAApp

Classe que cria e preenche a estrutura de dados desejada para ser apresentado na Web. Se o usuário escolhe “planilha”, esta classe que se comunica com a WEBSpreadSheet e formata como deve ser a planilha, no fim das contas. O mesmo ocorre quando o parâmetro passado é “Grafo”.

15.8 SAOutputGenerator

Esta classe é responsável por gerar um grafo ou uma planilha a partir dos resultados e avaliações das pesquisas sobre análise de sentimento recebidas, dependendo da escolha do usuário (`generateGraph:` e `generateSpreadsheet:`) e exporta o que foi gerado (`export:as:`).

15.9 SASentimentAnalyser

Nesta classe, ocorre a classificação das opiniões retiradas das redes sociais através da utilização dos dicionários de palavras positivas e negativas. Por enquanto, o único critério adotado foi a contagem

de cada tipo de palavra. Por exemplo, se há mais palavras positivas na postagem, ela inteira será contabilizada como positiva.

Contudo, após essa contagem e analisada a proporção entre os dois tipos de palavras, é calculado ainda um viés. Se a diferença entre palavras positivas e negativas for menor que um índice que escolhemos, o texto é classificado como neutro.

15.10 SASentimentLabel

`SASentimentLabel` nada mais é do que uma classe para armazenar e retornar os tipos possíveis de classificação (GOOD, BAD ou NEUTRAL).

15.11 SASentimentTable

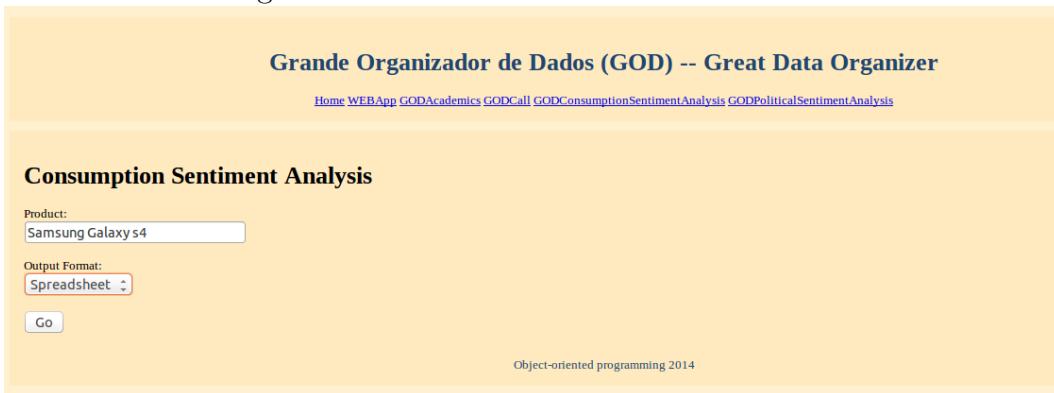
Esta classe é responsável por fazer uma tabela de sentimentos (GOOD, BAD ou NEUTRAL). Ela leva em consideração as avaliações feitas.

15.12 Rodando as Aplicações

15.12.1 Análise de Sentimento de Consumo

Para iniciar essa aplicação, basta clicar na opção do cabeçalho `GODConsumptionSentimentAnalysis`, que então surgirá a tela para pesquisar algum produto, figura 1.

Figura 1: Análise de Sentimento de Consumo



Um possível resultado dessa pesquisa seria, por exemplo, o da figura 2, mostrado a seguir:

15.12.2 Análise de Sentimento de Político

Para rodar essa aplicação é preciso selecionar a opção: `GODPoliticalSentimentAnalysis`, no cabeçalho. Assim, aparecerá a seguinte tela:

Um possível resultado dessa pesquisa seria, por exemplo, o da figura 4, mostrado a seguir:

Figura 2: Pesquisa - Galaxy s4

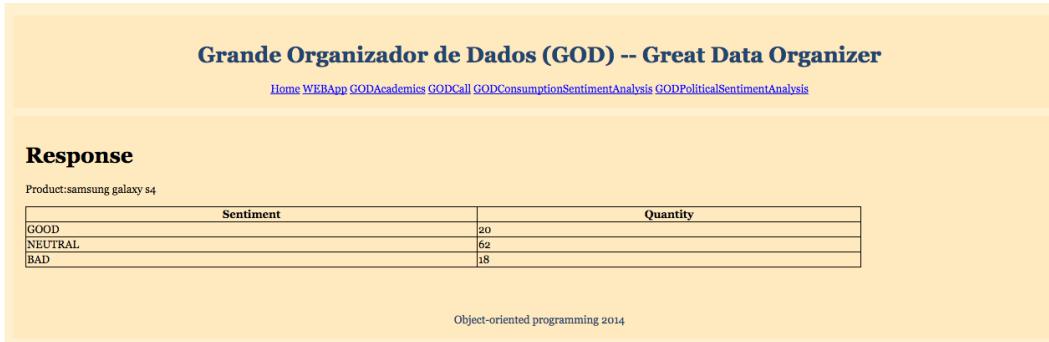
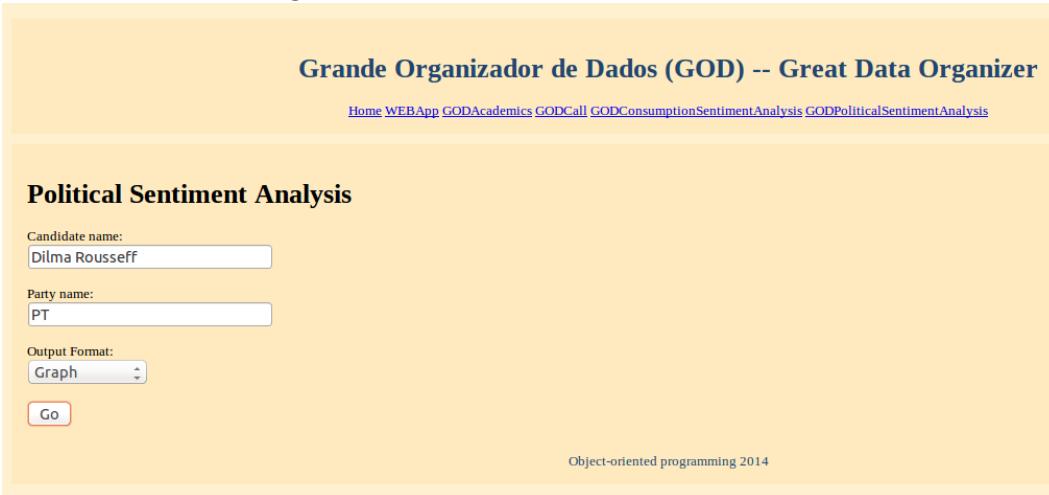


Figura 3: Análise de Sentimento Político



15.13 Exemplos de código

'Iniciando os dicionarios positivos e negativos'
SAInterface init.

'Contando palavras que aparecem no dicionario positivo a partir de uma string'
SASentimentAnalyser countOfWordsOf: (SAInterface positiveDictionary) in: 'aceita aceita
aceita aceita aceito azar'.

'Classificando uma string de acordo com a frequencia de seus termos positivos e negativos',
SASentimentAnalyser classify: 'aceita ladro chifrudo' using: (SAInterface
positiveDictionary) and: (SAInterface negativeDictionary).

Figura 4: Pesquisa - Dilma PT

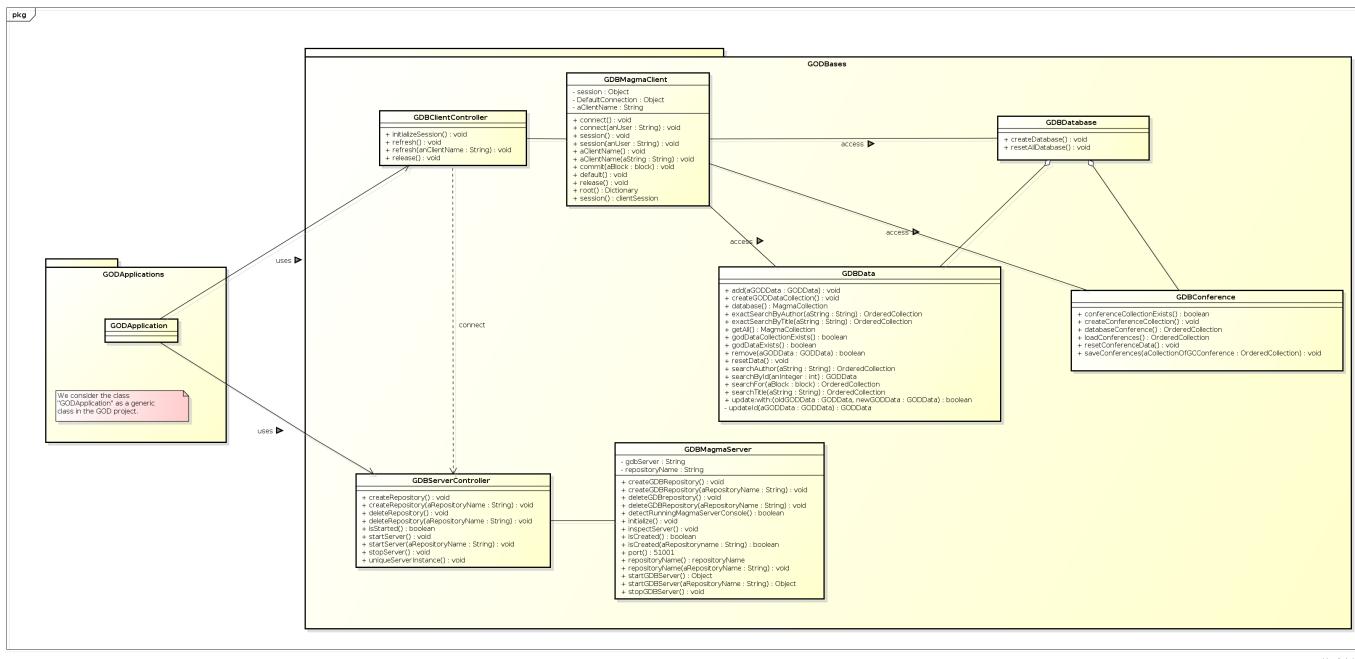


'Analisando um termo e retornando ele como spreadsheet'
political:= SAIInterface new.
sheetP:=political analyse:'Dilma' as: 'spreadsheet'.

16 Diagramas de classe do GOD

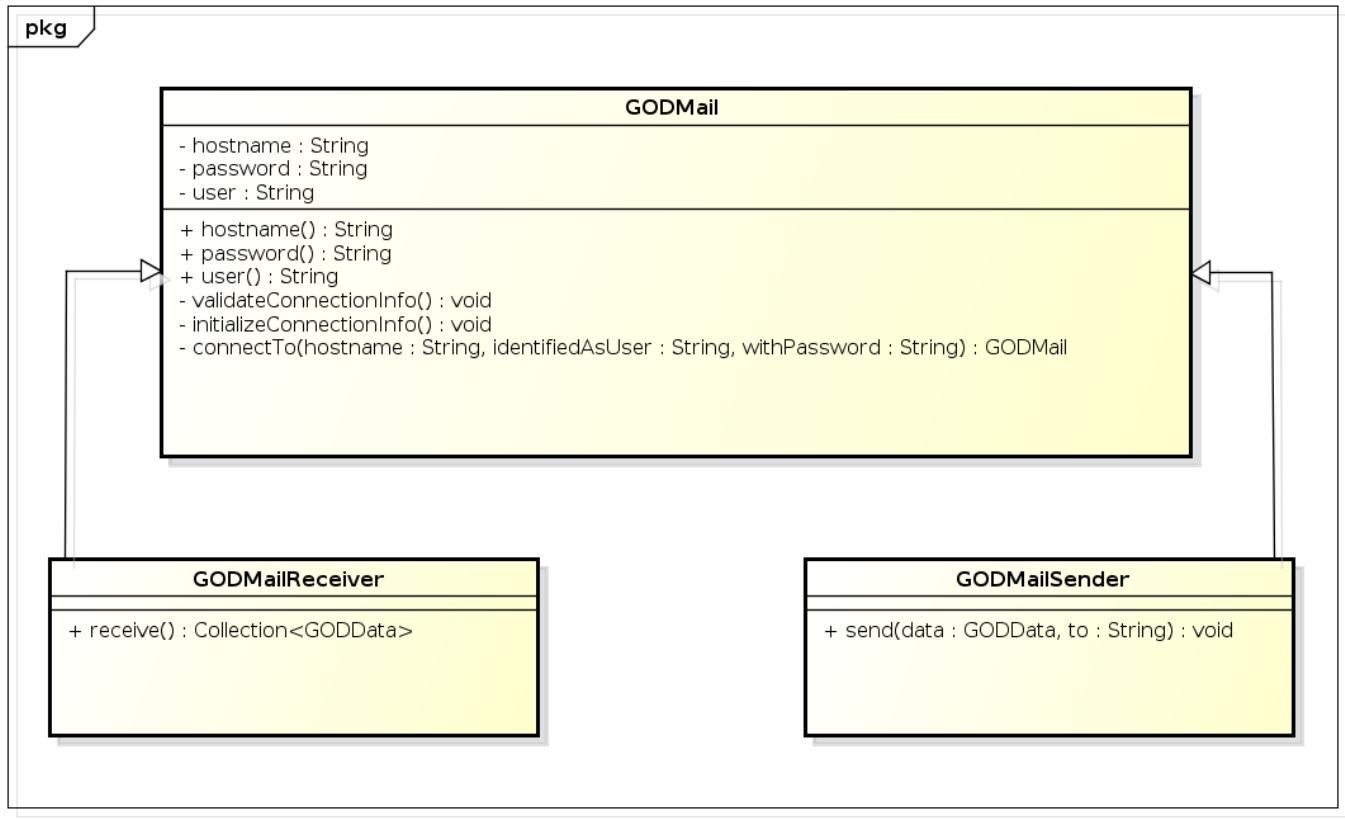
16.1 GOD

16.2 Banco de dados (GODBases)

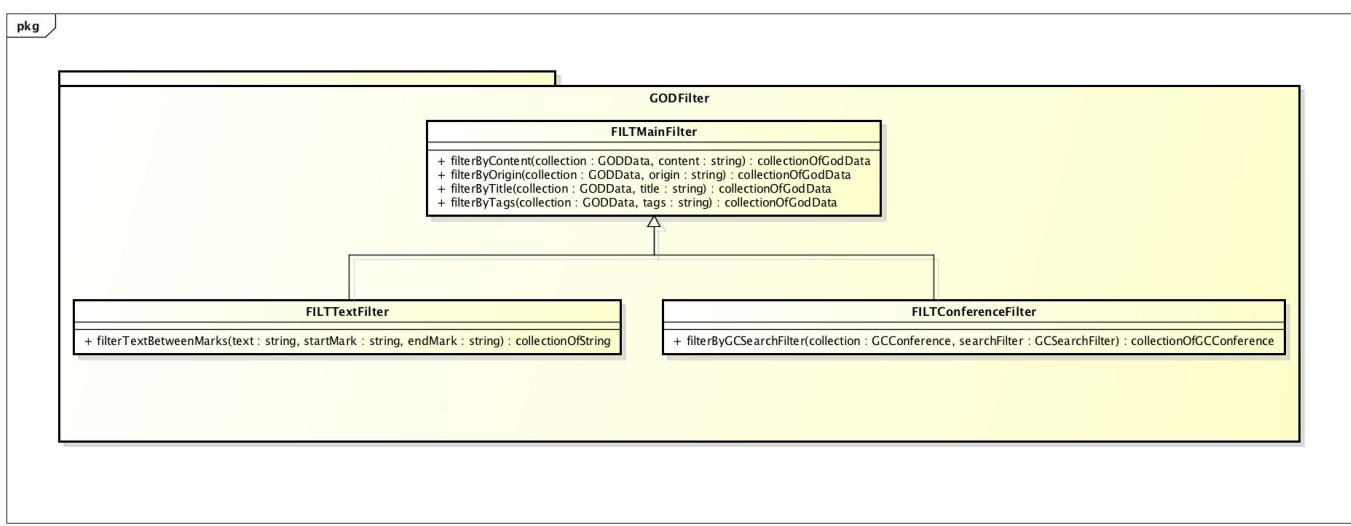


powered by Arsitekt

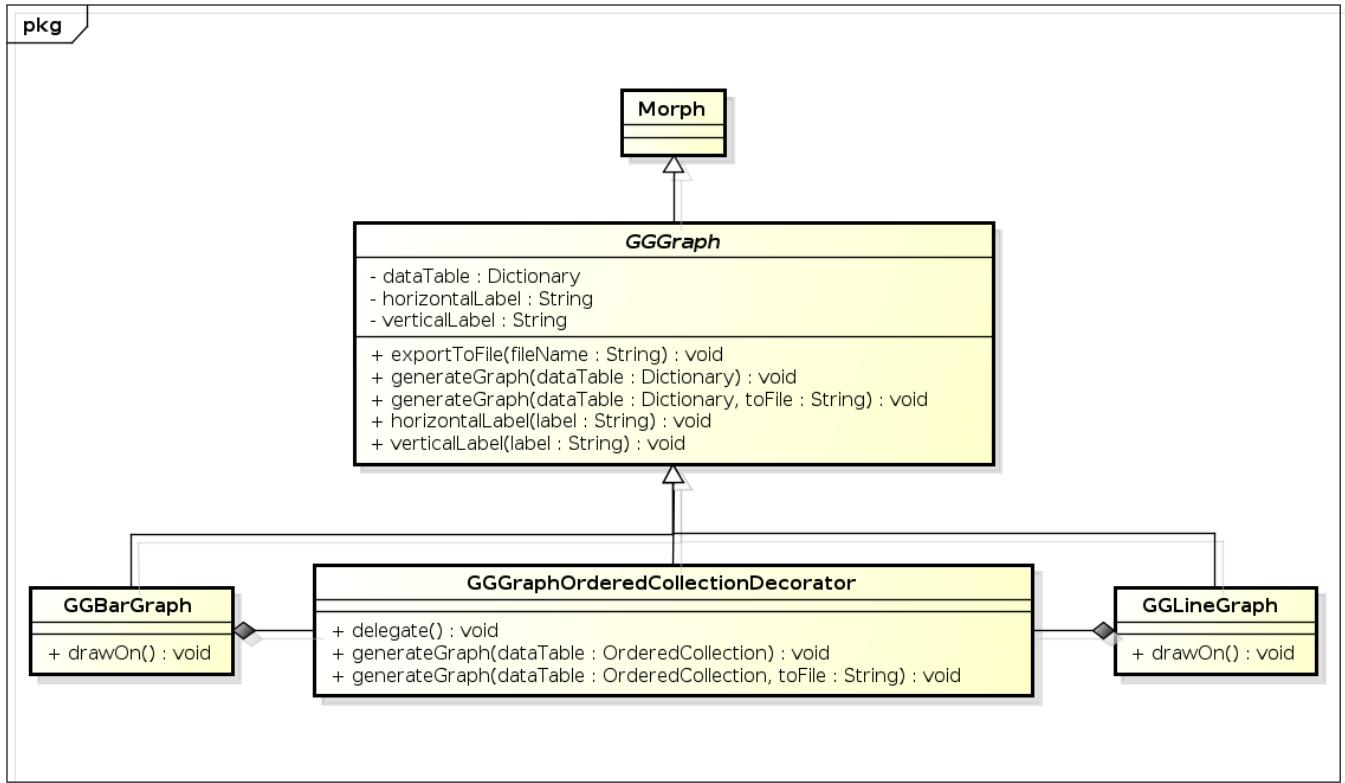
16.3 E-mails (GODEmail)



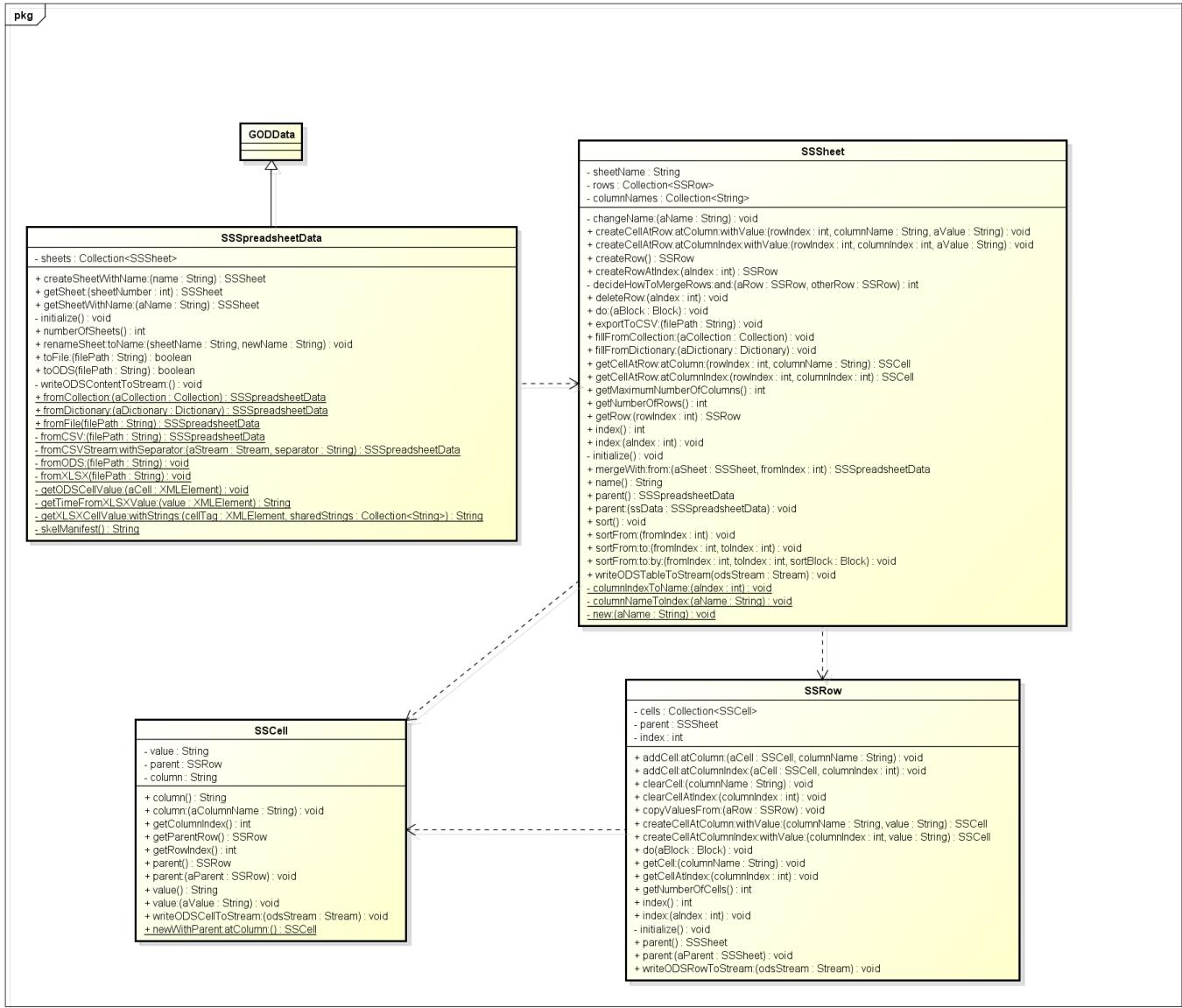
16.4 Filtros (GODFilter)



16.5 Gráficos (GODGraphGenerator)

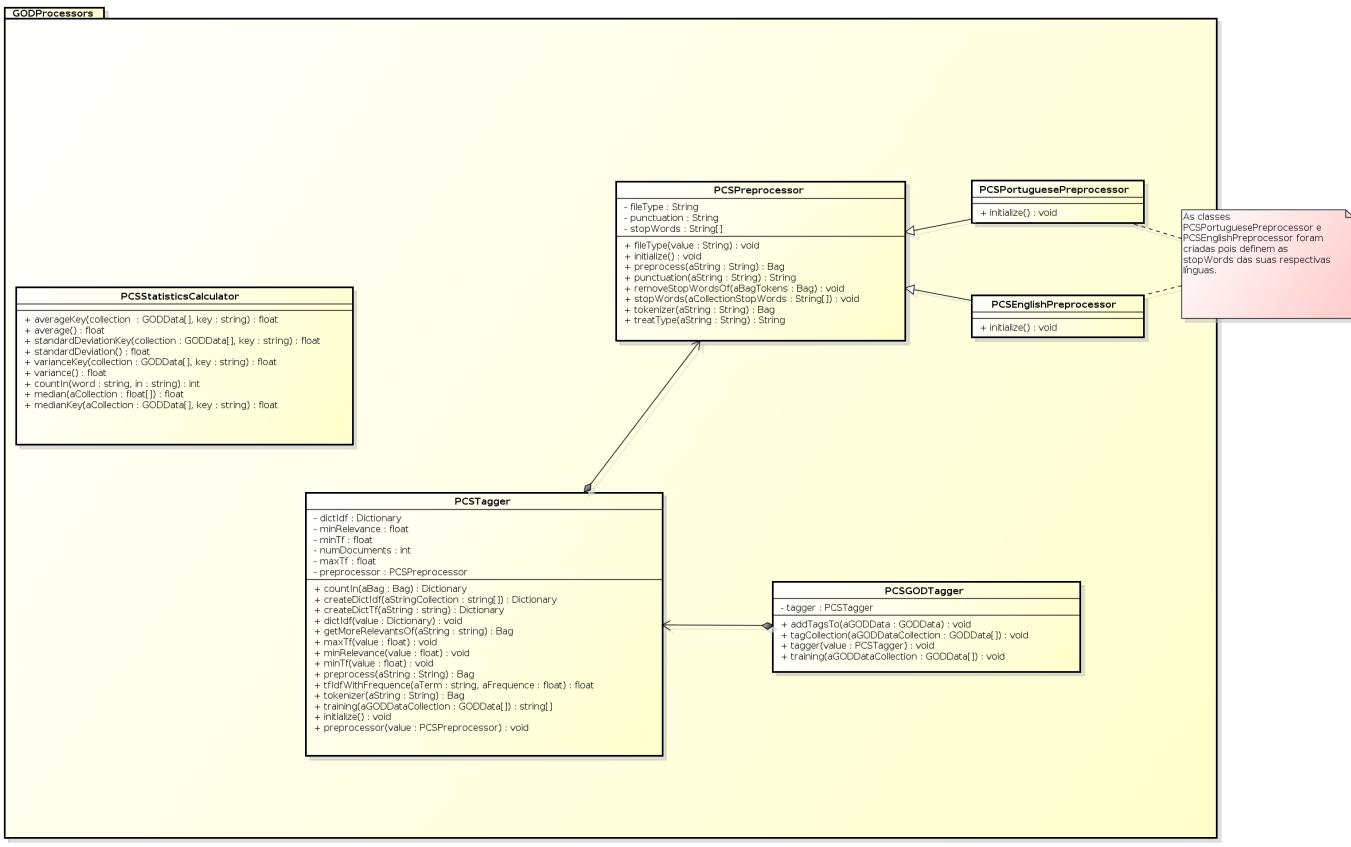


16.6 Planilhas (GODSpreadsheet)



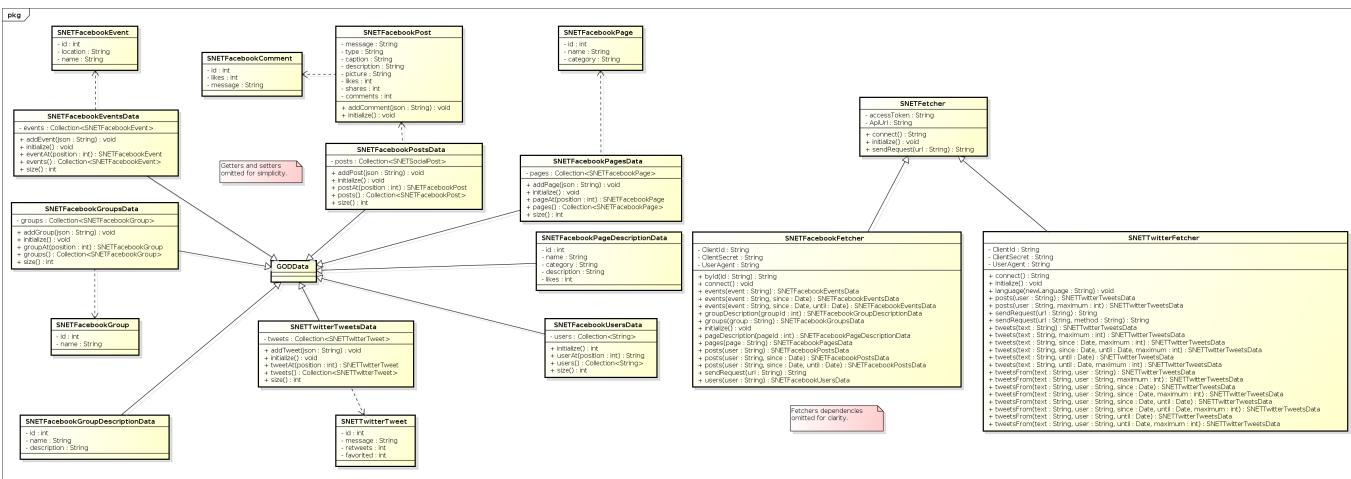
powered by Astah

16.7 Processadores (GODProcessors)



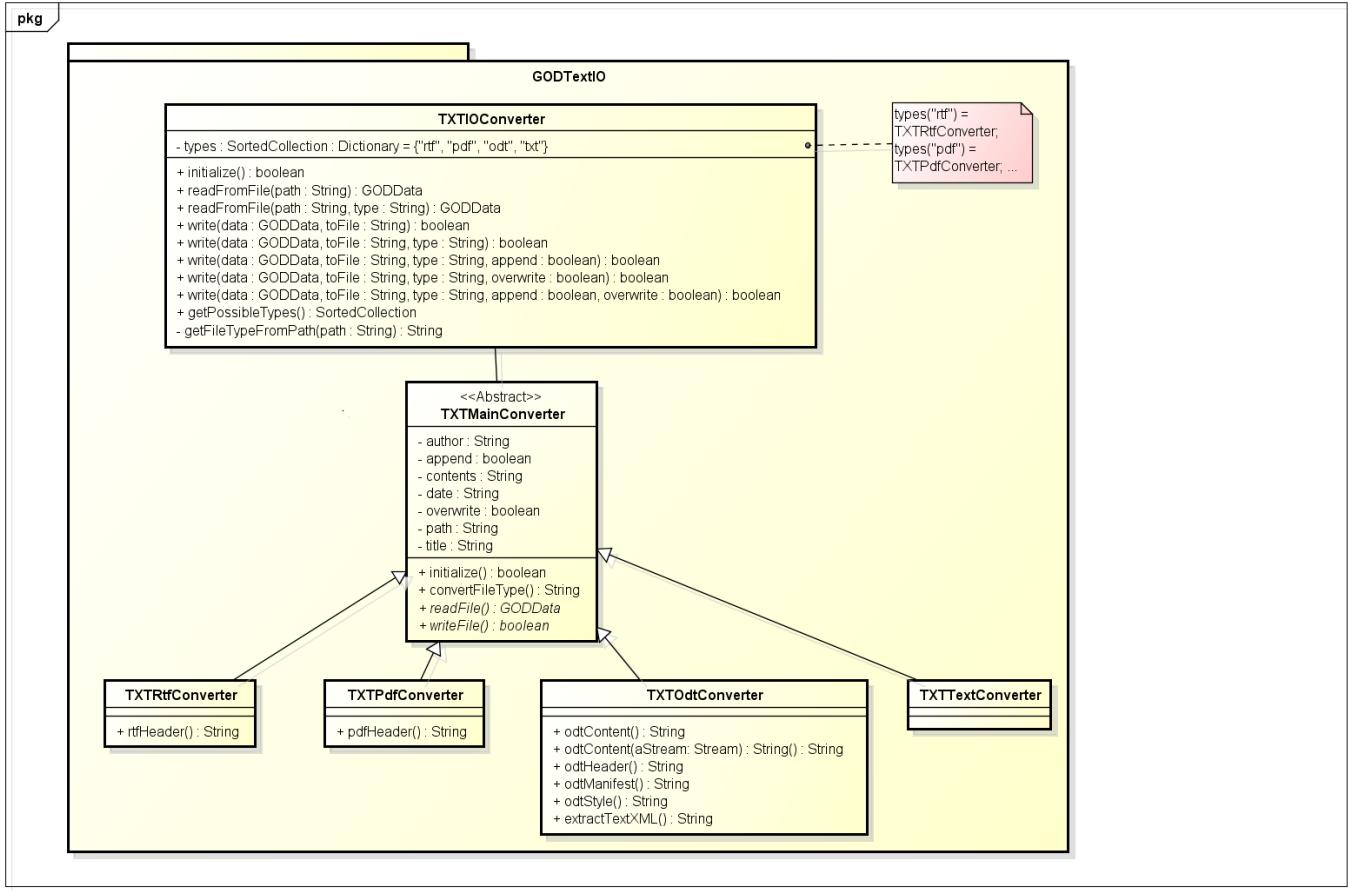
powered by Astah

16.8 Redes sociais (GODSocialNetIO)



powered by Astah

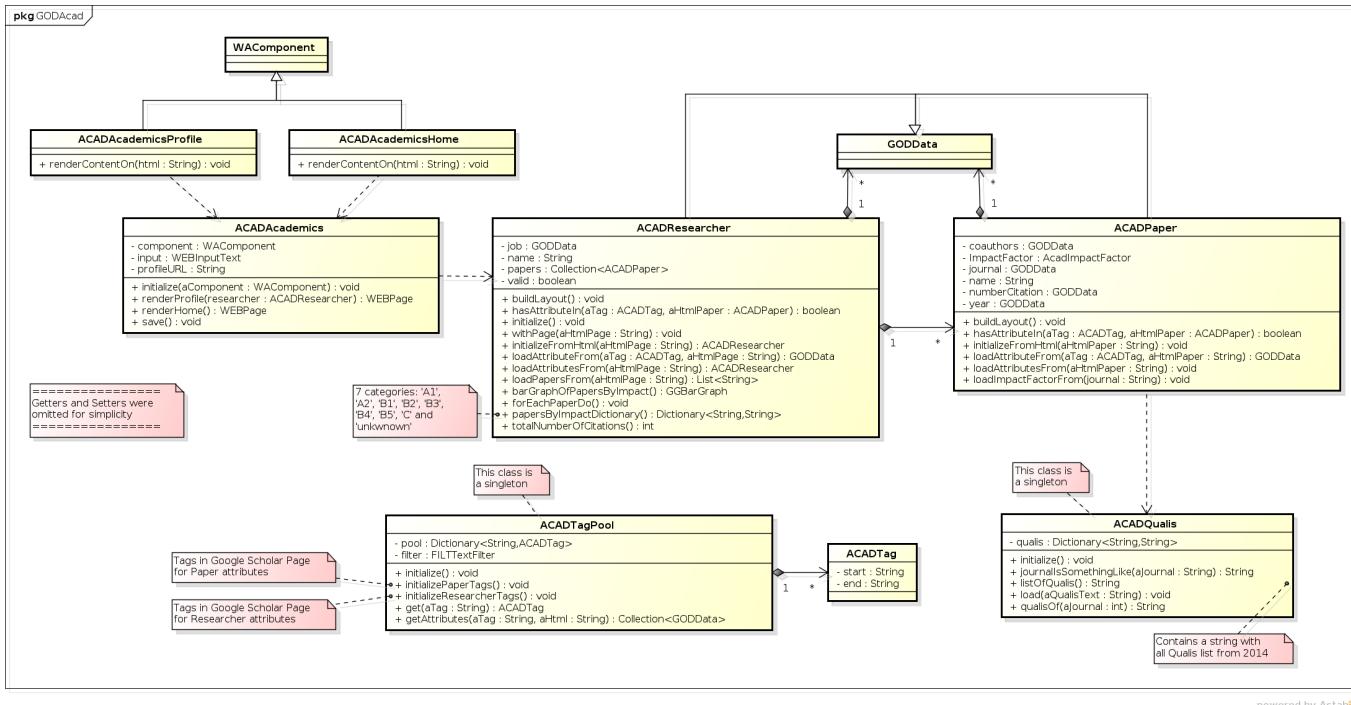
16.9 Textos (GODTextIO)



powered by Astah

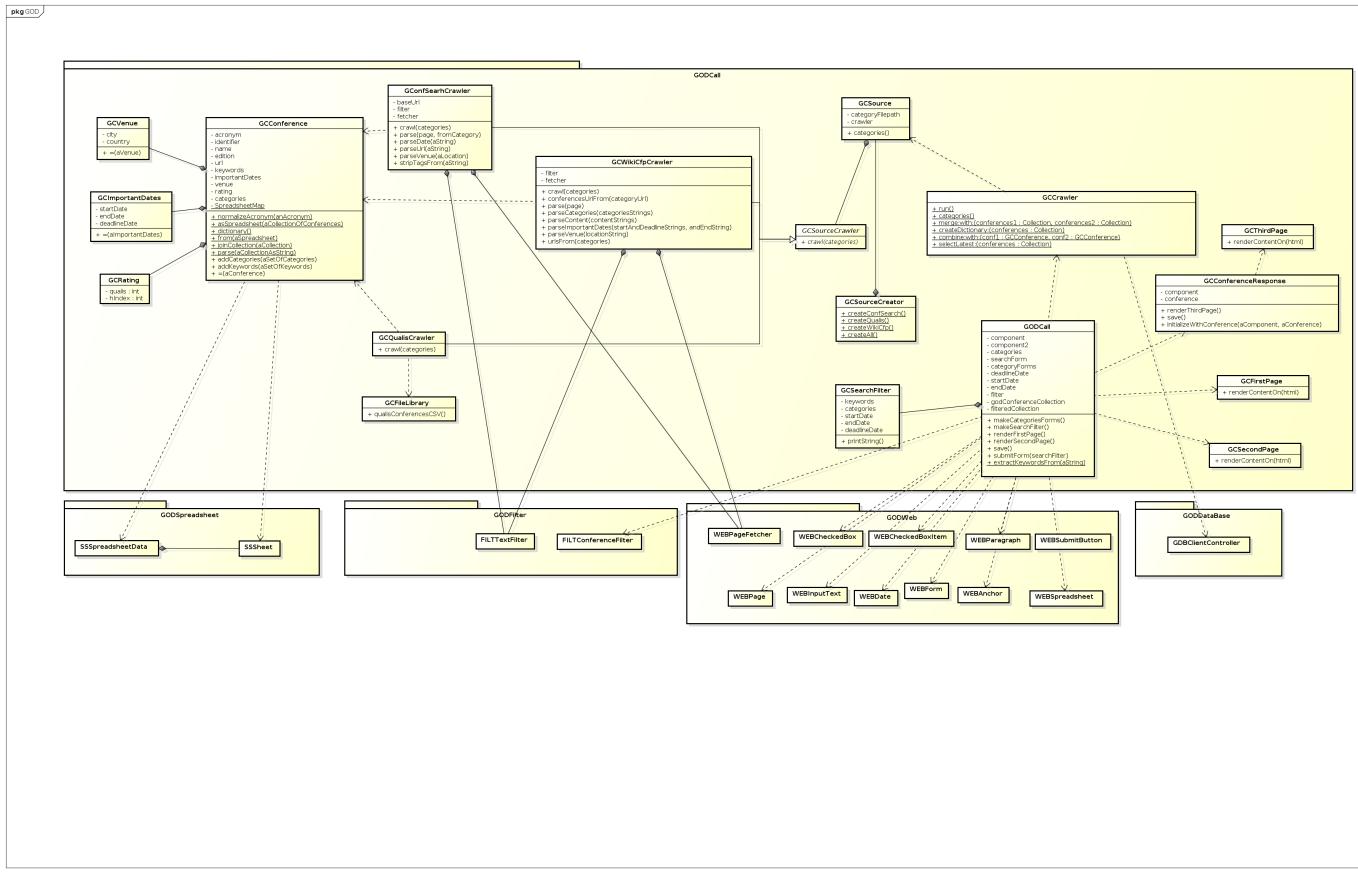
16.10 Web (GODWeb)

16.11 Agregação de conferências (GODAcademics)

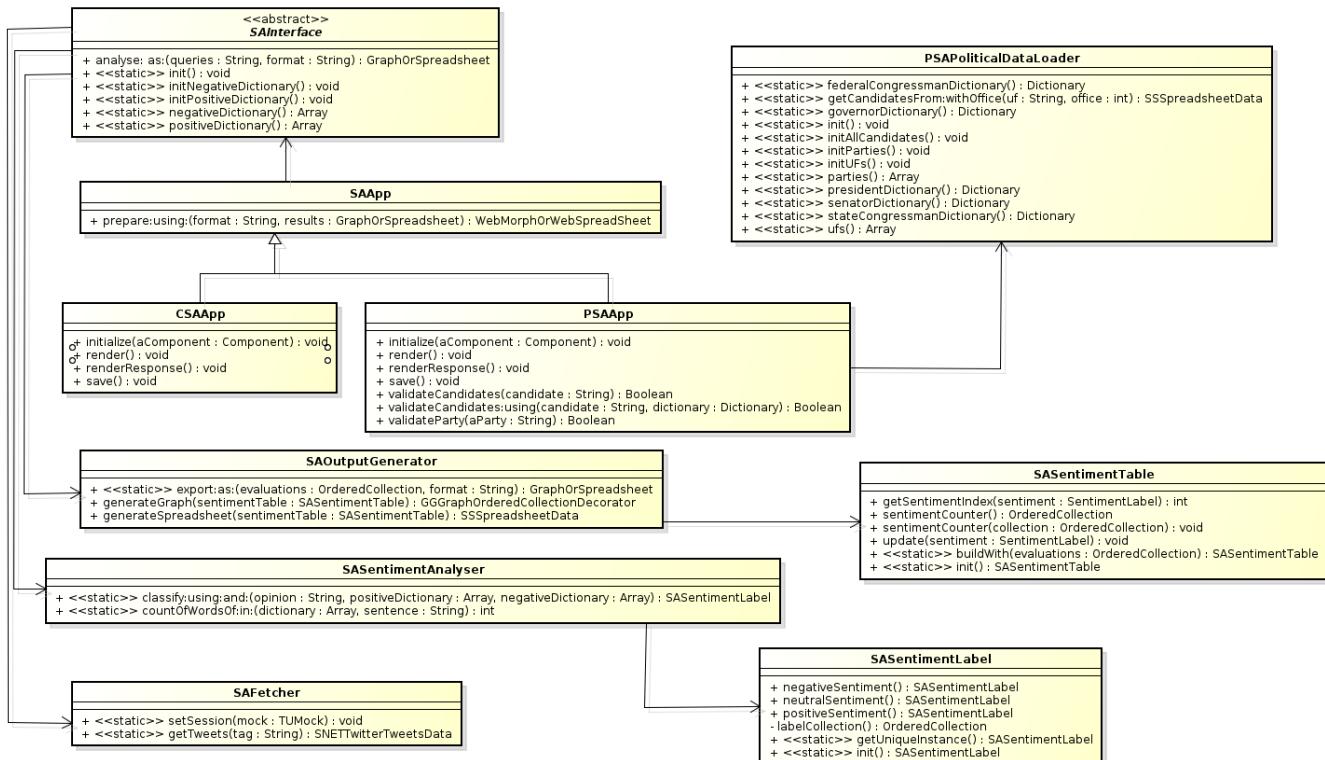


powered by Astah

16.12 Agregação de informações acadêmicas (GODsCall)



16.13 Análise de sentimentos de consumo e político (GODSentimentAnalysis)



17 Requisitos não implementados e sugestões de requisitos futuros

Alguns requisitos previstos para o projeto não foram desenvolvidos. Também foram identificados requisitos que podem ser desenvolvidos no futuro para acrescentar novas funcionalidades ao projeto. Abaixo são listados esses requisitos de acordo com cada módulo.

17.1 Banco de dados (GODBases)

1. Busca por tags de um GODData.
2. Realização de backups do banco de dados.

17.2 E-mails (GODEmail)

1. Envio de arquivos como anexos.
2. Envio de emails em HTML.

17.3 Filtros (GODFilter)

1. Realizar buscas por uma coleção de strings, retornando GODDatas que contenham todas elas.
2. Uso de expressões regulares.
3. Integração com o banco de dados.
4. Filtrar HTML: Criar um método que recebe como parâmetro um arquivo HTML e um XPath e retorna o que encontrar com aquele caminho em todo o HTML.
5. Filtro completo: Criar um método de filtro da classe FILTMainFilter que receberia como parâmetro uma coleção de GODData e uma lista de Strings e retornaria todas as coleções de GODData que contém alguma das Strings em qualquer um dos seus atributos. Ou seja, o filtro buscaria em 'content', 'tags', 'title' e 'origin' ao mesmo tempo, sem precisar especificar em qual campo buscar.

17.4 Gráficos (GODGraphGenerator)

1. Geração de gráfico de pizza.
2. Uso de um pacote de geração de gráfico para melhorar a qualidade visual.
3. Salvar os gráficos como byteArray no Squeak (usando a classe WAFileLibrary do Seaside).

17.5 Planilhas (GODSpreadsheet)

1. Escrita em XLSX.
2. Tratamento de caracteres especiais Os caracteres &, >, < e alguns outros tem uma representação diferente em um xml. Portanto, faz-se necessário um tratamento especial para eles. Talvez o método asHtml da classe String resolva.

17.6 Processadores (GODProcessors)

1. Uso do tf-idf como medidor de relevância: É possível utilizar o tf-idf (já implementado) para definir uma medida de relevância para os termos de um documento, isso pode ser interessante para ponderar a análise de um texto.
2. Extender PCSTagger para uma classe de busca de documentos: Criar um método para comparação de consulta e documentos. Esse método precisará de um vetor para cada documento e cada consulta onde as dimensões serão os termos da coleção e o valor será dado pelo tdidf daquele termo no respectivo documento ou consulta. Por fim a similaridade entre documento e consulta é dada pela distância angular dos mesmos, quanto menor a distância mais relevante aquele documento é para a consulta.
3. Stemming ou lemmatization: Na classe PCSPreprocessor criar meios para extrair o radical das palavras. Ex: 'processar' e 'processamento' devem ser tratados pelo mesmo radical, um exemplo pode ser 'processo' isso pode melhorar os resultados do PCSTagger. Esses métodos podem estar vinculados a língua, dessa forma é melhor adicioná-los nas subclasses de PCSPreprocessor.
4. Implementar novas técnicas de análise de recuperação de informação.

17.7 Redes sociais (GODSocialNetIO)

1. Twitter - Pesquisa por mais tweets: O twitter limita a quantidade de tweets retornados em 100. Existem formas de contornar isso, pois ele permite paginar o resultado. Isso é interessante quando o resultado é muito grande, e poderia ser implementado como uma melhoria.
2. FB - Usuário de teste é limitado: É preciso utilizar um usuário ao fazer algumas buscas na API do facebook. Como utilizamos um usuário de teste (criado no developer.facebook.com), ele tem acesso a poucas informações. Isso tem impacto em buscas de eventos, posts e principalmente de usuários. Talvez seja interessante utilizar um usuário real para essas buscas.
3. Twitter - Limite semanal: O twitter retorna apenas os tweets da última semana. Datas muito antigas não são retornadas. Por isso, talvez seja interessante salvar as buscas mais utilizadas semanalmente em algum banco de dados, para que pudessem ser utilizadas pelas aplicações.
4. FB - busca de usuários: A busca de usuários não retorna dados, provavelmente pois a API exige que um usuário seja autenticado e utilizado nas buscas. Portanto, só são retornados dados que esse usuário tem acesso. Como utilizamos um usuário de teste do app, ele não tem acesso a muitas coisas, e usuários reais é uma delas.

17.8 Textos (GODTextIO)

1. Leitura e escrita em DOCX.
2. Escrita de arquivos PDF sem o uso do `pdflatex`.
3. Leitura de arquivos RTF sem o uso de um *parser* externo (abiword).
4. Leitura de arquivos PDF sem o uso de um *parser* externo (abiword).

17.9 Web (GODWeb)

1. Permitir o posicionamento de componentes web em diferentes lugares de uma página: Inicialmente a ideia era usar um grid do objeto GODData para isso, mas esse grid pode ser colocado na classe WEBPage. A ideia é permitir definir um tamanho e posição para os componentes web dinamicamente.
2. Incluir a string do conteúdo html em um objeto GODData, e retornar esse objeto ao invés da String. Isso é feito na classe WEBPageFetcher.
3. Criar classes para tags html que ainda não tenham sido criadas.
4. Disponibilizar páginas para upload de arquivos.
5. Receber CSSs específicos para cada aplicação: Permite personalizar cada aplicação.

17.10 Agregação de conferências (GODAcademics)

1. Uso de tags sobre as publicações que devem ser apresentadas em uma conferência.
2. Busca mais informações: Usar outras fontes além do Google Scholar para obtenção de informações.
3. Busca fuzzy por journals: Hoje a busca é feita apenas pelos prefixos dos nomes dos journals. Seria interessante que essa busca fosse melhorada. Uma ideia é a busca fuzzy, onde, por exemplo, "plos comp biology" seria encontrado como "plos computational biology". Link: http://en.wikipedia.org/wiki/Approximate_string_matching.

17.11 Agregação de informações acadêmicas (GODsCall)

1. Crawler incremental: Criar um sistema de crawling incremental, para que possa ser feito aos poucos. A ideia é reduzir o tempo para carregar a aplicação.
2. Pesquisa em novas fontes: Enriquecer mais o banco de dados, buscando conferências em novas fontes, como o <http://academic.research.microsoft.com>.
3. Buscar novos tipos de informação: Enriquecer as informações de uma conferência, com informações sobre o local da conferência, transporte, clima... Também informações acadêmicas, como estatísticas sobre os trabalhos apresentados.

17.12 Análise de sentimentos de consumo e político (GODSentimentAnalysis)

1. Melhorar a análise de sentimentos: Atualmente a análise é feita somente pela frequência dos termos positivos e negativos em um texto.
2. Usar o método tf-idf de GODProcessors na análise de sentimentos.
3. Melhorar o layout: Deixar o leiaute mais amigável ao usuário.
4. Mostrar a planilha e o gráfico na mesma busca.
5. Adicionar cores à planilha: Usar cores para diferenciar melhor os rótulos (verde para GOOD, amarelo para NEUTRAL e vermelho para BAD).
6. Adicionar cores ao gráfico: Usar cores para diferenciar melhor os rótulos (verde para GOOD, amarelo para NEUTRAL e vermelho para BAD).
7. Melhorar as técnicas de análise de sentimento: Aprimorar a análise, avaliando relações de uma palavra com a outra.
8. Adicionar autocomplete para os candidatos e partidos: Ao invés de validar se o candidato/partido informado pelo usuário é válido somente após o envio do formulário, implementar uma opção de autocomplete no campo de busca. Assim, o usuário começa a digitar o nome de um partido/candidato e o campo vai buscando na base de dados quais opções casam com os caractéres já informados de forma que o usuário possa selecionar uma opção.
9. Criar opção de selecionar período de tempo: Permitir ao usuário selecionar o período de tempo que este quer realizar a análise. Para tanto, pode-se permitir que este especifique um intervalo de dada, ou selecione períodos pré-determinados, como por exemplo, primeiro turno das eleições do ano X. É provável que seja necessário alterar a forma como é feita a análise e coletar e persistir tweets periodicamente ao invés de buscar tweets no momento da pesquisa.
10. Adicionar técnicas alternativas de análise de sentimentos: Adicionar a opção de técnicas de análise de sentimento que não utilizem dicionários.

17.13 Sugestões de novos módulos de fontes de dados

1. Módulo para obtenção de dados a partir de web services.

17.14 Sugestões de novas aplicações

Diversas aplicações podem ser implementadas usando os recursos fornecidos pelos módulos do GOD. O uso de fontes de emails, web-sites, arquivos texto, planilhas e redes sociais permite criar aplicações para diversos domínios.

1. Análise de informações sobre dados públicos do governo.
2. Análise de sentimento de sobre empresas, usando fontes como o site reclame aqui.

3. Análise de dados da bovespa.
4. Criação de corpus com base em artigos de um domínio de interesse. A partir de um conjunto de artigos, que podem ser carregados em um repositório do GOD, pode-se permitir a verificação de frases e termos frequentes usando GODTextIO e GODProcessors.

18 Instalação

O projeto GOD é compatível com o Squeak 4.4, que pode ser obtido em <http://ftp.squeak.org/4.4/Squeak-4.4-All-in-One.zip>. O GOD roda sobre o sistema operacional Ubuntu. Isso ocorre devido a algumas dependências externas à máquina virtual do Squeak. Mais detalhes sobre a instalação do projeto serão detalhadas a seguir.

18.1 Pré-requisitos

18.1.1 Instalação do Metacello

O pacote metacello é usado para gerenciar o controle de versões do projeto GOD. Para instalá-lo digite os seguintes comandos no workspace:

```
Installer squeaksource
  project: 'MetacelloRepository';
  install: 'ConfigurationOfMetacello'.
(Smalltalk at: #ConfigurationOfMetacello) perform: #load.
```

18.1.2 Instalação do Seaside

O Seaside é o framework web usado para construir a interface visual do GOD. Para instalá-lo, acesse o menu **Apps»SqueakMap Catalog**. Em seguida digite “seaside” em Search Packages. Selecione a versão 3.0 e clique em Install.

Outra opção é fazer o download pelo Metacello. Basta executar os comandos abaixo em um workspace do Squeak.

```
Installer squeaksource
  project: 'MetacelloRepository';
  install: 'ConfigurationOfSeaside30'.
(Smalltalk at: #ConfigurationOfSeaside30) load.
```

Execute os comandos (“do it”) e aguarde a finalização do processo. Para conferir se a instalação foi feita com sucesso, clique em Apps veja se aparece a opção Seaside Control Panel.

O Seaside vem com o servidor web chamado **comanche**. Para ativar o comanche, abra **Apps»Seaside Control Panel**, clique com o botão direito do mouse em **Add adaptor**, selecione a porta e clique em **Start**.

18.1.3 Instalação do Magma

O Magma é o banco de dados orientado a objetos usado pelo GOD. Para instalar o Magma, acesse o menu **Apps»SqueakMap Catalog**. Em seguida digite “magma” em Search Packages. Selecione a versão 1.4 e clique em **Install** (ver Fig. 5 e Fig. 6).

A seguir, selecione o pacote **client** e escolha a opção **install**.(Ver Fig. 7). Repetir o passo anterior para instalar os pacotes **server** e o **test**. (Ver Fig. 8)

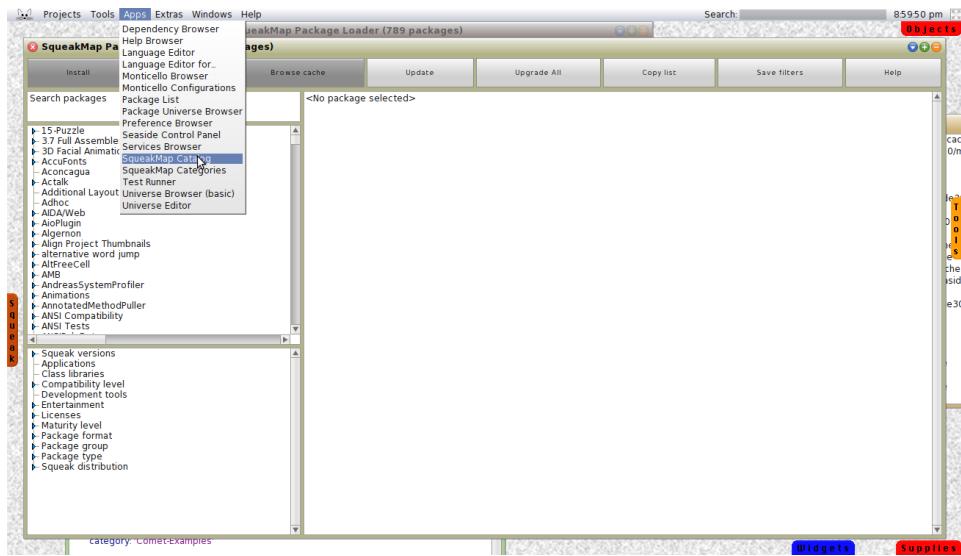


Figura 5: Instalação do magma

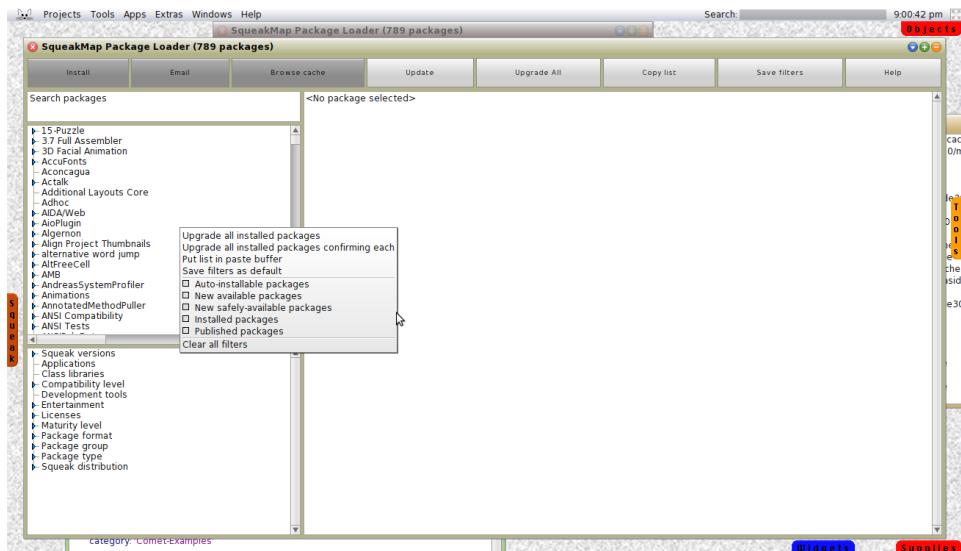


Figura 6: Instalação do magma (2)

18.1.4 SqueakSSL-bin

O SqueakSSL é um conjunto de pacotes usados para acessar conteúdo HTTPS. Alguns módulos do GOD usam o SqueakSSL: GODEmail, GODSocialNetIO e GODWeb. Esse pacote é instalado junto com os demais módulos do GOD. No entanto, é necessário instalar um arquivo binário do SqueakSSL antes de fazer a instalação do GOD. Baixe o arquivo em <https://squeakssl.googlecode.com/files/SqueakSSL-bin-0.1.5.zip>. A seguir, copie o arquivo **so.SqueakSSL** dentro dos diretórios do Squeak em **Contents/Linux-i686/lib/squeak/4.4.7-2357**.

Existem também os arquivos binários para OS X e Windows. Porém, nos testes que fizemos com o OS X o procedimento não funcionou.

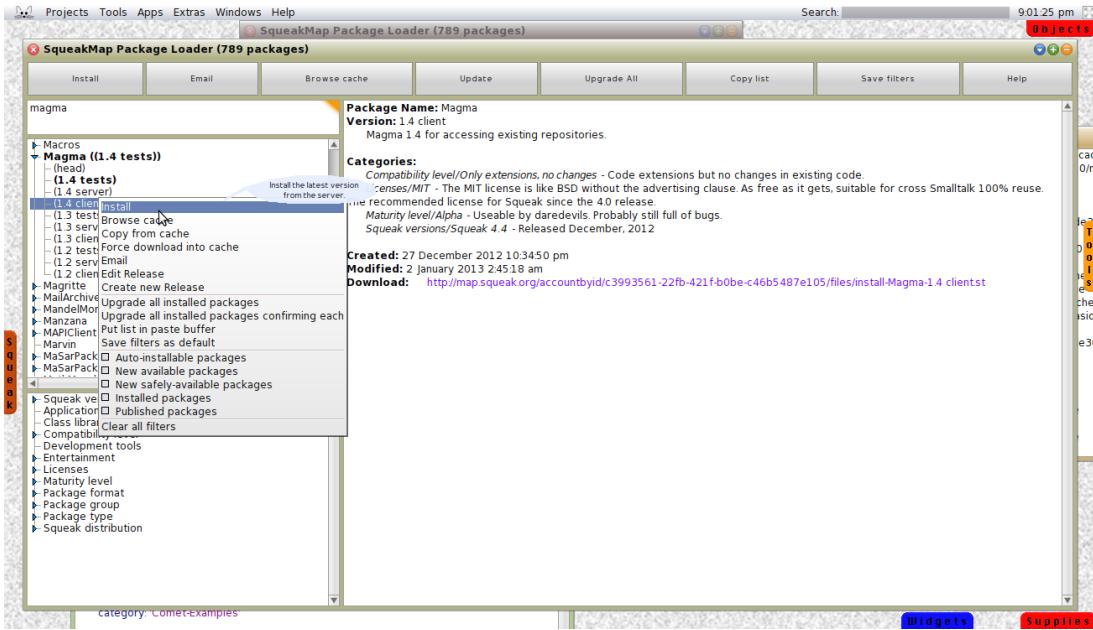


Figura 7: Instalação do magma (3)

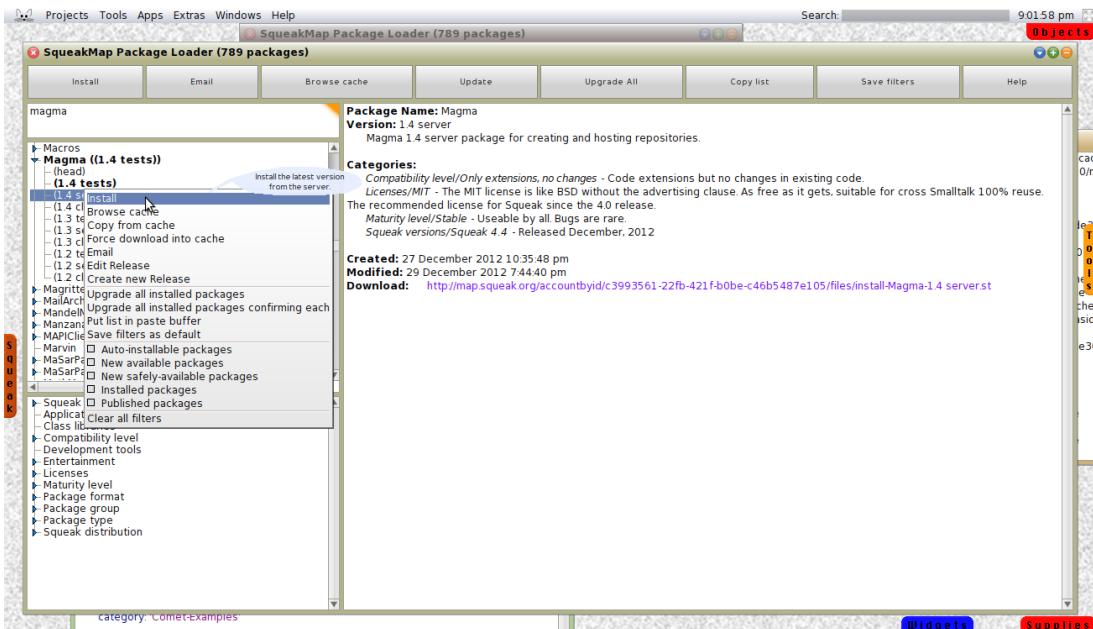


Figura 8: Instalação do magma (4)

18.1.5 pdflatex e abiword

Para realizar a leitura e escrita dos formatos de texto PDF e RTF é necessário ter instalado os programas pdflatex e abiword (em sua versão 2.9.2). Ambos programas devem estar funcionando por linha de comando. O pacote **OSProcess** faz a chamada desses programas pelo Squeak para realizar as conversões. Infelizmente não existe uma versão do abiword compatível com as versões atuais do OS X.

O Abiword pode ser obtido em <http://www.abisource.com/downloads/abiword/2.9.2/source/abiword-2.9.2.tar.gz> e o pdflatex acompanha as principais distribuições latex.

18.1.6 Repositórios do GOD

O repositório do GOD fica no SmalltalkHub <http://smalltalkhub.com/mc/higoramario/GOD/main>. Há também um repositório usado para backup do projeto no SqueakSource3 (<http://ss3.gemstone.com/ss/GOD/>).

18.1.7 Instalação do GOD

Para instalar o GOD, inclua o repositório do projeto no Monticello. Clique em **Tools»Monticello Browser»+Repository»HTTP** e digite a url. Selecione a url e clique em **open**. Na janela seguinte, selecione o pacote **GODKernel** do lado esquerdo e selecione o pacote mais atual na janela do lado direito. Clique em **load**.

Para obter todos os pacotes mais atuais do projeto, digite os seguintes comandos no **workspace**:

```
(ConfigurationOfGOD project version: '0.1-baseline') load.
```

As dependências externas necessárias para o funcionamento do GOD estão listadas abaixo. Todas elas são incluídas no download dos pacotes do GOD.

- JSON
- OSProcess (Tests-OSProcess)
- SqueakSSL (SqueakSSL-Core, SqueakSSL-SMTP, SqueakSSL-Tests)
- VB-Regex
- WebClient (WebClient-Core)

18.1.8 Inicialização do GOD

Inicie o servidor web do seaside. Em seguida, para iniciar o GOD digite no workspace o comando:

```
WEBGodHome initialize.
```

Todas as aplicações e o GOD serão registrados no Seaside e estarão disponíveis para uso.

18.1.9 Problemas identificados

Versões do Squeak: A versão 4.5 do Squeak apresentou problema com o uso do Magma. Por esse motivo optamos por usar a versão 4.4 do Squeak. Por sua vez, tivemos problemas com o uso do SqueakSSL na versão 4.4 (como descrito anteriormente), o que limita o uso do GOD ao Linux, no qual esse problema de compatibilidade pode ser resolvido.

Abiword: A versão 2.9.2 do abiword deve ser usada no projeto, já que a versão dos repositórios oficiais do Ubuntu (3.0.0) não funciona via linha de comando. Além disso, não existe versão compatível do programa para OS X em suas versões mais atuais. Esse é outro fator que limita o uso do GOD ao Linux.

GODAcademics x Google Scholar: O Google Scholar é usado pelo módulo GODAcademics como fonte de informação para os dados processados pelo módulo. Em algumas tentativas de acesso o Google reconheceu a aplicação como um robô, impedindo a obtenção dos dados.

GODSocialNetIO: As APIs do Facebbok e do Twitter possuem algumas limitações quanto às consultas que podem ser feitas. Há restrições de tempo e quantidade de posts/tweets que podem ser acessados. No facebook só é possível visualizar dados dentro do escopo do usuário que é usado para o acesso à api. Além disso, há rumores dizendo que a API do Twitter será cancelada.

GODsCall: Durante o final da fase de desenvolvimento do projeto, o ConfSearch, uma das fontes de informação do módulo, ficou indisponível, limitando as informações do GODsCall aos dados do WikiCFP.