Faculdade de Tecnologia Senac RS
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

ROTEIRO DO RELATÓRIO DE PROJETO PARCIAL

IDENTIFICAÇÃO

Aluno: Felipe Nascimento de Moura
Endereço Residencial: R. Duque de Caxias,278, apto. 102
Bairro: Centro CEP: 90010-180
Cidade: Porto Alegre UF: RS
Telefone(s): 51 – 3024 5696
E-mail(s): 51 – 8425 5530

Título:

Desenvolvimento de um interpretador de linguagens de auto nível, modularizado e escalável, capaz de ser integrado com diferentes linguagens e bancos de dados.

Orientador:

Guilherme Bertoni Machado

Apresentação Geral:

Este documento contextualiza o desenvolvimento de um interpretador de linguagens de auto nível, modularizado e escalável, capaz de ser integrado com diferentes linguagens e bancos de dados e usado por meio de diferentes interfaces.

O sistema é todo desenvolvido baseando-se em uma estrutura de API, a qual pode ser consumida por diferentes interfaces, incluindo requisições HTTP, o que permitirá a terceiros o desenvolvimento de sua própria interface, esteja ela rodando em um browser ou não, consumindo a API em forma de serviço (baseado no modelo SaaS, com referências ao padrão REST).

Um dos focos deste projeto *Open Source* é lembrar que a mente humana tem a habilidade de criar, inovar ou tornar algo artístico, enquanto as máquinas são infinitamente superiores em tratamento lógico e matemático. Isso nos remete à conclusão de que devemos deixar para as máquinas todas as tarefas que se possa automatizar, liberando tempo para a mente humana trabalhar em tarefas que exijam suas habilidades. Trata-se de fazer um melhor uso do que se tem ao alcance em cada uma das pontas.

O modelo de desenvolvimento atual já é relativamente antigo e tende a evoluir muito brevemente para uma forma ainda mais próxima da comunicação interpessoal humana, o que inspira o desenvolvimento deste projeto.

Para a realização deste trabalho será utilizado o conceito de *linguagem de programação discreta*, criado pelo autor deste documento, que o cita desta forma:

> “Uma linguagem de programação discreta consiste em uma linguagem baseada em grupos variáveis de padrões e instruções com alguns marcadores padrões com a abilidade de se adaptar conforme seu uso fazendo uma transformação da forma original de alto nível para uma forma padrão em mais baixo nível”

Este conceito prevê que a linguagem de programação deva seguir grupos independentes de regras, sendo passível de agregação para novos grupos, ou adaptação para grupos depreciados. Isto deverá tornar a própria linguagem de programação capaz de aprender e se adaptar conforme o uso sofrendo variações inclusive conforme tradições ou costumes da região onde está sendo empregada.

Tal interpretador deverá manter alguns princípios do projeto já existente denominado theWebMind, também desenvolvido pelo autor deste trabalho de conclusão. Trata-se de uma nova versão, mais robusta e com mais embasamento científico, de forma mais escalável e receptiva à futuras alterações.

Definição do Problema:

Este projeto busca solucionar o seguinte problema:

O desenvolvimento de softwares vem há muito se adaptando e enriquecendo, adotando novas tecnologias e equipamentos cada vez mais acessíveis. O crescimento da demanda por novas aplicações é inegável e embora haja o desenvolvimento de inúmeras aplicações inovadoras nas mais diversas áreas, ainda há, e sempre haverá uma grande chamada por softwares comerciais, empresariais ou focados em determinados nichos do mercado. O que se observa é justamente a adoção de um padrão contínuo, estudado em faculdades e cursos sobre como implementar regras e padrões no desenvolvimento de software. Porém, se ao desenvolver um software, implementamos muitos padrões e regras, devemos pensar seriamente no fato de termos máquinas hoje em dia com muito maior capacidade para implementar tais regras. Eis o ponto onde a preocupação por uma melhor utilização do conhecimento e habilidades artísticas e relacionadas a criatividade, por parte do cérebro humano. O tempo investido na padronização e implementação de regras durante o desenvolvimento de um determinado software poderia estar sendo aplicado em outras áreas, ou mesmo, aprimorando as já existentes.

O desenvolvimento das primeiras versões deste projeto já provou que com o processamento de uma máquina comum, é possível computar muito rapidamente tais padrões, de forma quase instantânea e confiável.

Para a evolução dos softwares, é impressindível que mudemos a forma como pensamos e desenvolvemos aplicações.

As novas metodologias para desenvolvimento, novas técnicas e technologias, além de artefatos para análise e equipamentos hoje disponíveis e a acessibilidade das informações oferecem um grande parque de diversões quanto a forma como desenvolveremos o próximo aplicativo, e cabe a nós, analistas e desenvolvedores decidir qual a melhor abordagem. Uma evolução nestas áreas é inevitável.

Seria possível, um código extremamente simples baseado em uma linguagem semi-natural, desenvolver uma aplicação capaz de gerar um sistema completo, incluindo o seu banco de dados, classes, formulários, documentação, etc?

Baseando-se na questão acima, os seguintes desafios devem ser resolvidos:

- Cadastrar novos projetos a serem desenvolvidos, bem como novos membros que atuarão no desenvolvimento dos mesmos. Poder-se-á especificar o SGBD que se deseja usar para o desenvolvimento, o idioma a ser utilizado para programação e também a linguagem ou framework a serem usados para geração de um resultado final e funcional.
- Desenvolver uma estrutura de interpretação de Português e Inglês, idiomas nos quais o usuário deverá ter liberdade para escrever, bastando seguir pequenos padrões para ter uma aplicação funcional gerada, ou mesmo documentação e diagramação. Esta

plataforma deverá estar preparada para conectar em um SGBD e gerar a base de dados conforme o necessário.
- Criar uma interface a fim de viabilizar o próprio usuário a criar seus próprios *plugins* ou módulos geradores de código e documentação, possibilitando o crescimento do projeto através da comunidade *open source*.

A plataforma precisará ter baixo acoplamento para que possa manter uma boa escalabilidade de serviços e módulos.

Objetivos:

- Estrutura padronizada e organizada para tratamento posterior a partir do código criado em alto nível;
- API para integração com a estrutura gerada, proporcionando meio para mineração e interação com os dados armazenados;
- Ferramenta para manipulação de projetos e usuários;

Análise de Tecnologias e Ferramentas:

Linguagens de programação usadas: PHP, PL/SQL, XML, shellscript, Javascript, CSS, HTML5;

Ferramentas para o desenvolvimento: Netbeans 6.9, Postgres, MySQL, SQLite;

Extensões: PHPPDO, FInfo, Mcrypt, XDebug, PHPUnit, ReadLine, SQLite, PHP-cli.

Sistemas operacionais: Ubuntu 10, Windows XP/Seven/Server, MacOSX;

Browsers a serem suportados: Firefox 3.6+, Safari 5+, Chrome 5.0+, Opera 10.0+;

Versionador: GIT;

Licensa: MIT Open Source license

Descrição da Solução:

O sistema será todo composto por módulos capazes de interagir uns com os outros, mas também capazes de oferecer funcionalidades independentemente do resto da estrutura. Estes módulos deve interagir conforme o necessário, mas não devem afetar o sistema como um todo.

Haverá um módulo específico para a interpretação do código escrito, tal módulo será dividido em outros submódulos, como validador léxico, flexionador de substantivos, identificador de tokens, interpretador de padrões e regras e analisador de estatísticas e probabilidades. Desta forma, podemos integrá-lo a um outro grupo, capaz de usar as informações e estruturas tratadas para fazer levantamentos e deduções, além de utilizar os mesmos para a sua aprendizagem.

Outro grupo de classes deverá ser responsável pela integração com banco de dados e padronização, bem como normalização do mesmo.

Em outro conjunto de classes, deverá haver o tratamento para sua API, de forma a possibilitar acesso de aplicações externas a fim de fazer uma mineração nos dados levantados pelo novo motor do theWebMind.

Na solução proposta, módulos podem ser considerados pacotes de classes e as requisições podem vir tanto via linha de comando, quanto por HTTP simulando o uso de REST.

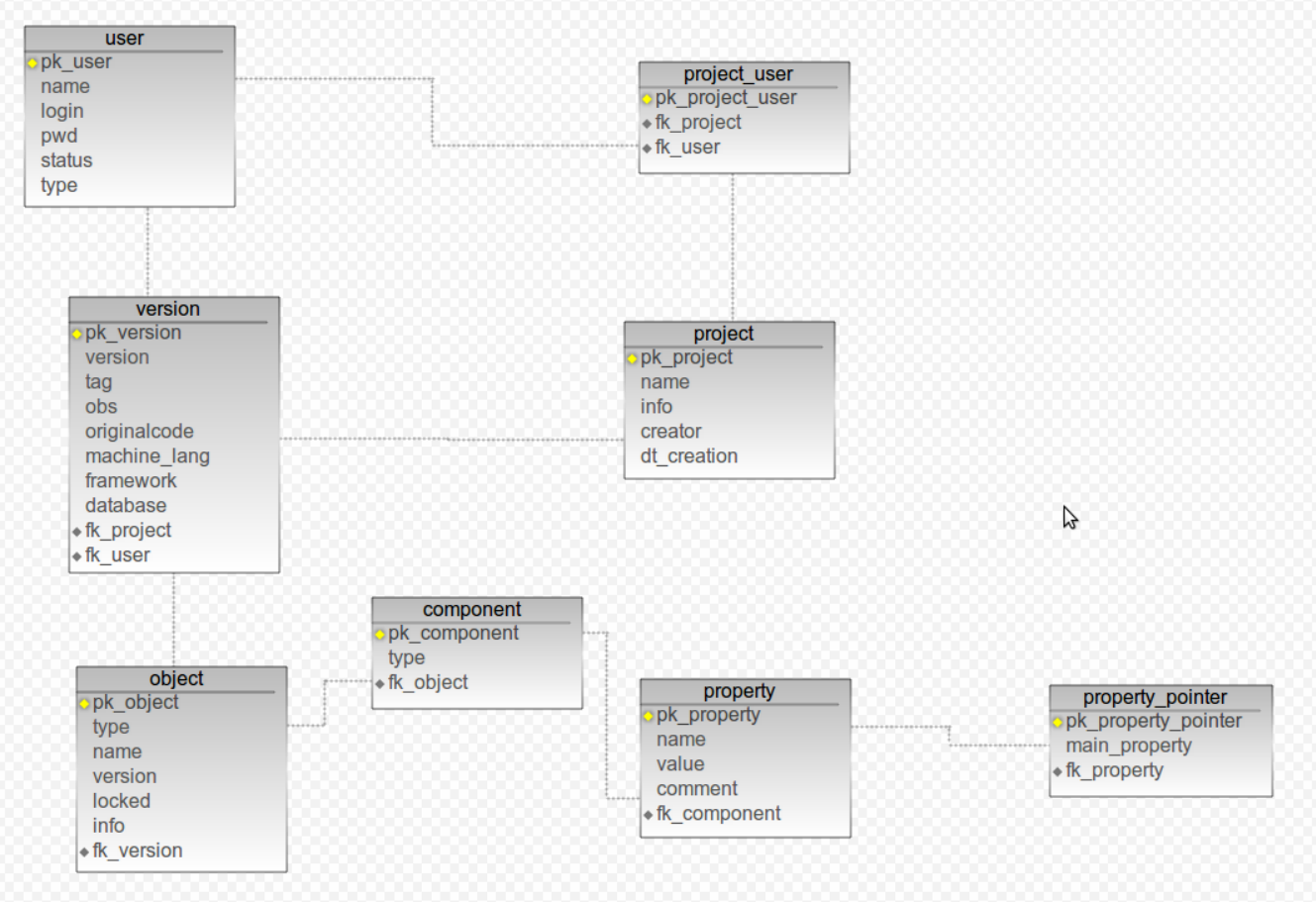
Rest foi escolhido como modelo para conexões via HTTP por ser de muito rápida aprendizagem e manter um padrão que a própria estrutura do protocolo HTTP oferece

nativamente. Além disto, acaba melhor aproveitando os dados enviados e recebidos adotando o padrão Json. Todavia, não foi implementada toda a interface de RestFull, que prevê diferentes cabeçalhos para cada tipo de requisição, adotando tão somente o Post como metodo de recebimento de solicitações.

Existe a possibilidade de instalar a aplicação no servidor para um uso mais fácil por meio de linhas de comando.

Ao efetuar a instalação, uma base de dados em SQLite será iniciada.

Todos os projetos e usuários serão armazenados nesta base de dados que também ficará como responsável por manter o histórico das alterações. Nesta base de dados, cada trecho do código gerado é armazenado separadamente, como classes, metodos, propriedades, interfaces, entre outros. Estes itens vinculam-se em forma hierarquica a fim de identificar a estrutura dinamicamente, ainda mantendo um status que indica qual a versão atual.

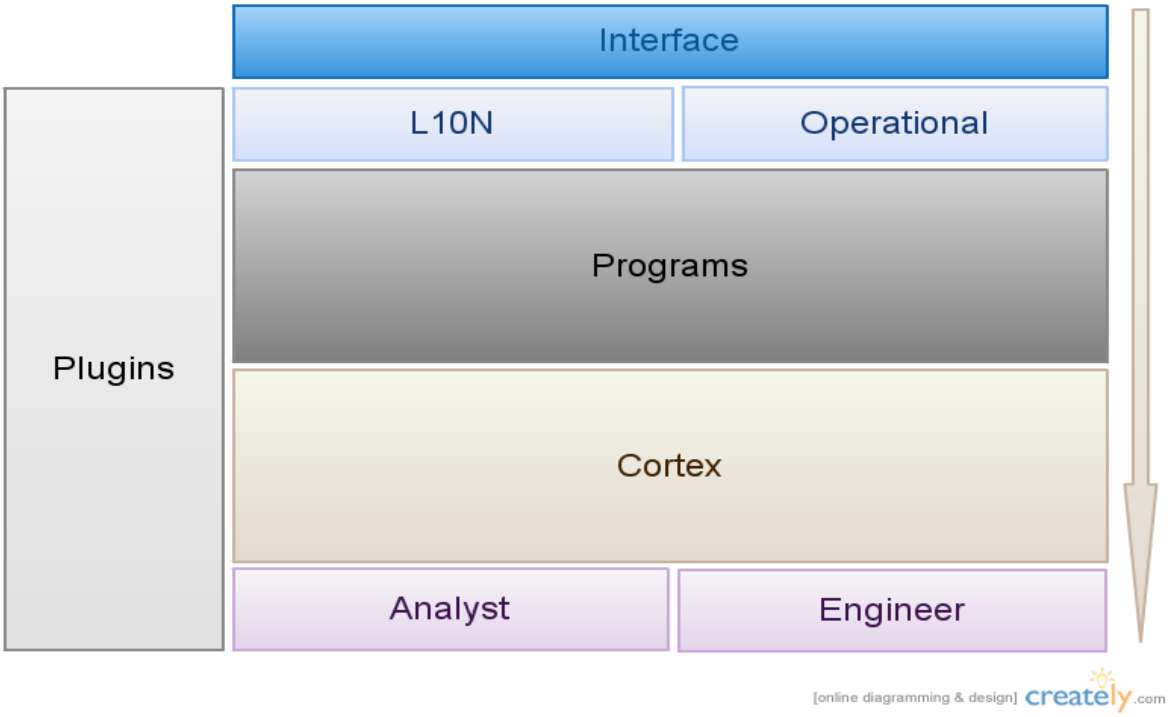


O usuário precisará autenticar-se com um login e senha antes de executar determinados comandos.

As tarefas disponíveis no sistema são dividias em dois principais níveis, a *camada de programas* onde encontram-se os comandos executados diretamente pelo usuário, e a *camada do cortex*, camada onde encontram-se as classes e conjuntos de classes responsáveis pela interpretação, tokenização, padronização e normalização das informações, e também para geração dos códigos em uma etapa final. Além destas duas camadas de funcionalidades, haverão também as classes para o controle da comunicação com o usuário implementando L10N para localization, com diferentes idiomas.

Há também uma camada de *integração de plugins*. O sistema oferecerá suporte a plugins por meio de chamadas padronizadas para cada plugin, aceitando em sua descrição o que será o gatílho, o trigger, que disparará a chamada de cada plugin. Este plugins são basicamente uma classe seguindo um padrão documentado, extendendo a classe MindPlugin e implementando a interface Plugin.

Todas as demais classes estão classificadas como *operacionais*.



Abordagem de Desenvolvimento:

A metodologia adotada será uma variação da Scrum, pois não terá as reuniões diárias, porém contará com a organização de sprints e documentação de metas com muitas pequenas entregas.

Também serão usadas técnicas de TDD, para testes e implementação.

A maior parte do sistema manter-se-a com uma abordagem Orientada a Objetos, porém, implementando também funcionalidades REST e tratando as aplicações do sistema como serviço, lembrando SaaS.

As alterações feitas no projeto estarão sendo versionadas com o controlador de versões GIT, sendo assim, todo o histórico das alterações estará seguro e atualizado em http://github.com/felipenmoura juntamente com todos os logs e documentação. Por se tratar de um projeto open source, existe a possibilidade de outros colaboradores alterarem algum código ou documento, todavia, por hora apenas o autor deste artigo está cadastrado como administrador do projeto com permissões de escrita, e também, para alterações futuras por parte de novos membros, sempre constará um log seguro e preciso de cada arquivo alterado, inserido ou removido do servidor.

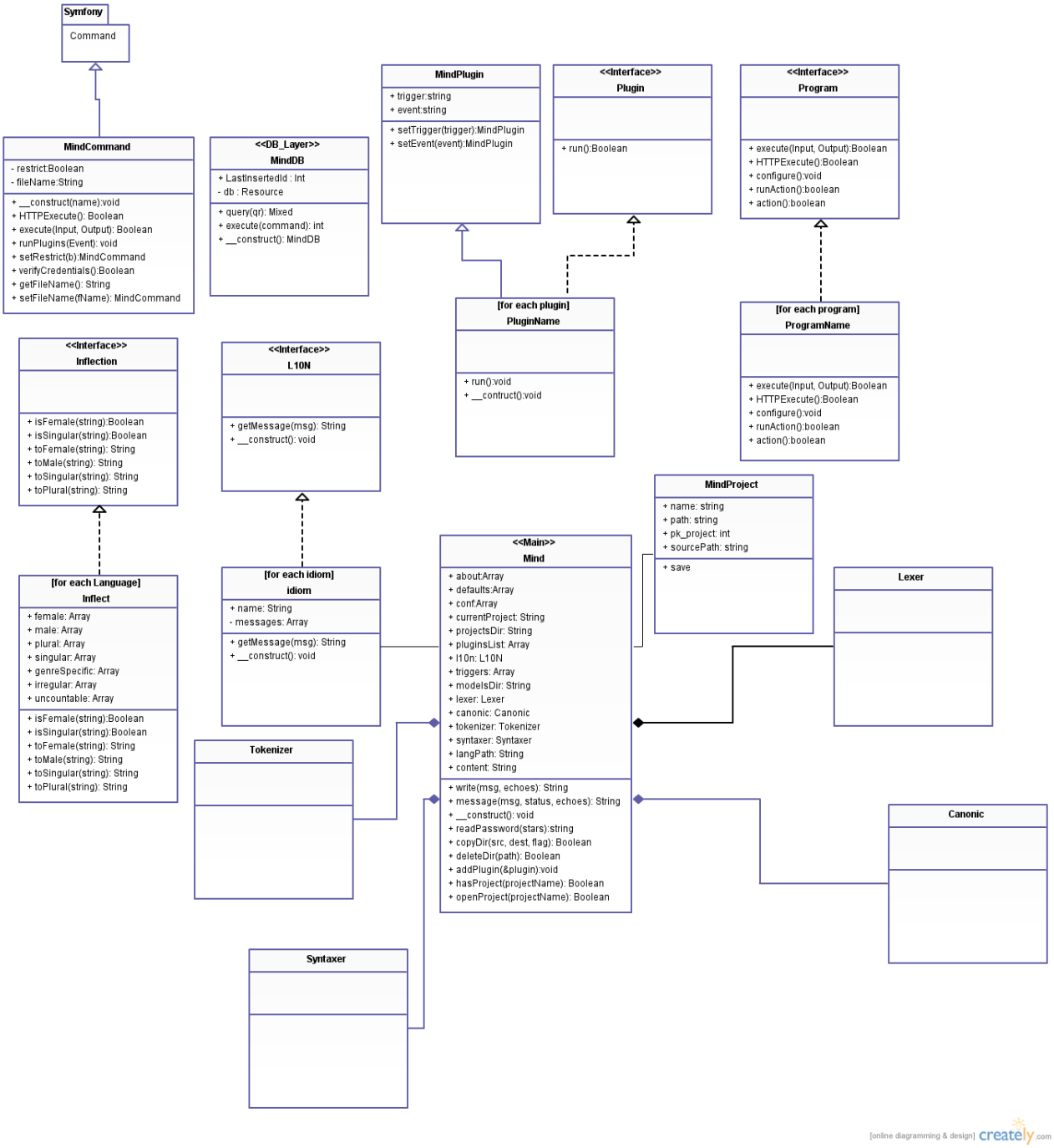
Serão adotados os comentários de Annotation, ou seja, todas as classes, packages/namespaces, interfaces e demais blocos de código estarão devidamente comentados seguindo um padrão que possibilita a geração automática de documentação, ao término do desenvolvimento, o que agilizará a documentação final da API e também das próprias classes.

Importante notar também que, justamente por se tratar de um projeto Open Source, grandes alterações podem ocorrer desde refatorar um módulo na análise, até reescrever classes de código, devido a possíveis intervenções no decorrer do desenvolvimento. Estas intervenções podem ocorrer em função de novas pesquisas, novos projetos a serem incorporados, novos contatos com pessoas de diversas áreas.

Há contato já iniciado com pesquisadores nas mais diversas áreas de estudo de idiomas em Inglês, Português e Espanhol, podendo-se expandir conforme novos contatos e oportunidades surjam.

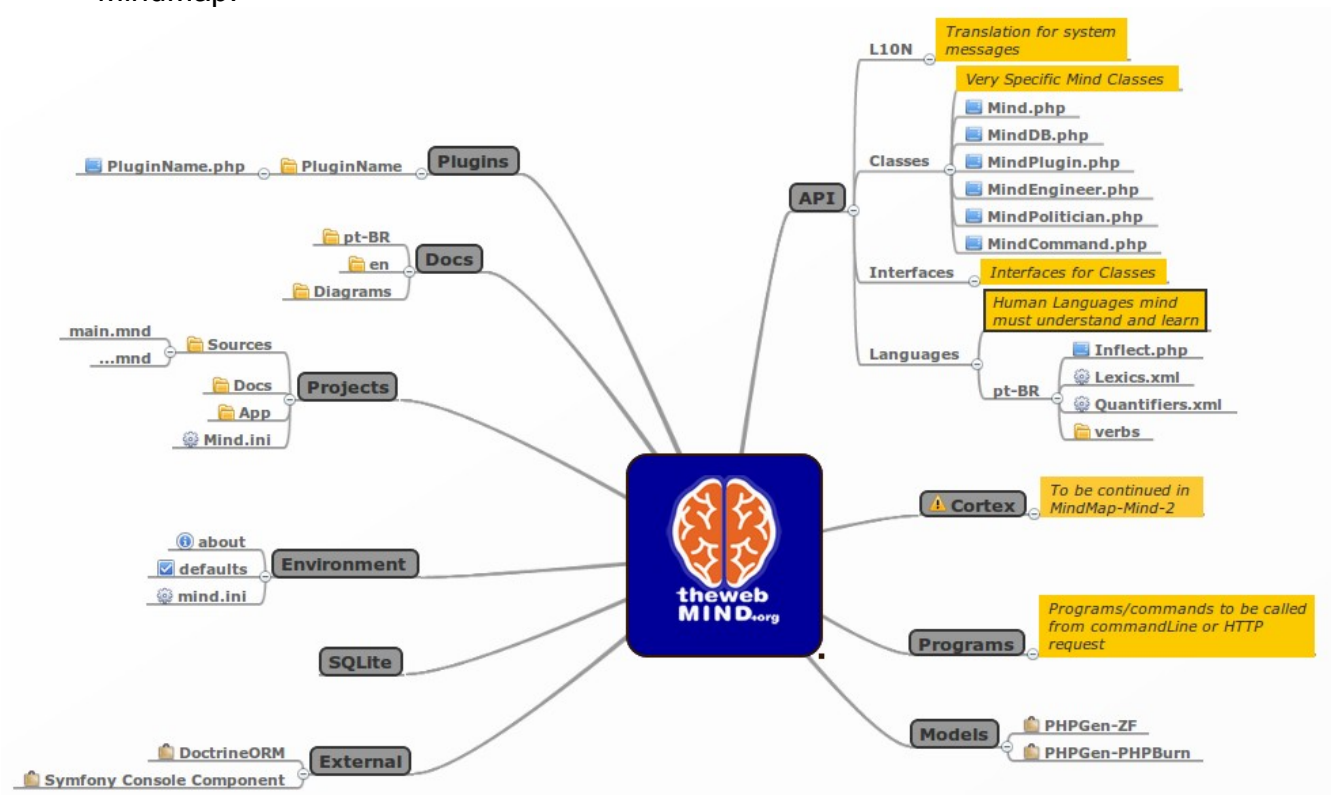
Arquitetura do Sistema:

- Diagrama parcial de Classes:

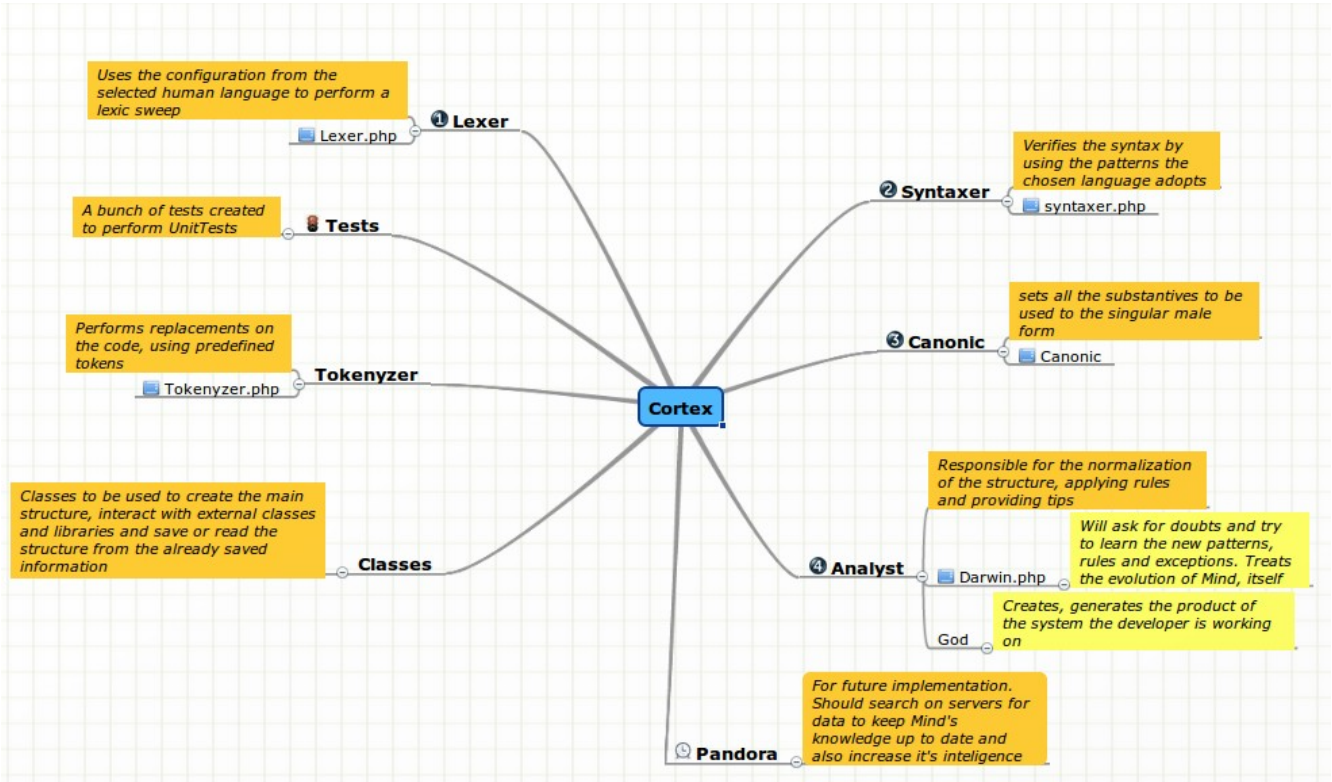


Este diagrama mostra a estrutura principal das classe e suas relações, mas alguns métodos e propriedades ainda serão adicionados conform eu desenvolvimento do sistema, e novas pesquisas e contatos.

- MindMap:



Este mapa mental mostra as funcionalidades e módulos que compões o projeto em altíssimo nível



Uma forma mais detalhada do módulo Cortex e suas taregas

- Diagrama de sequência:

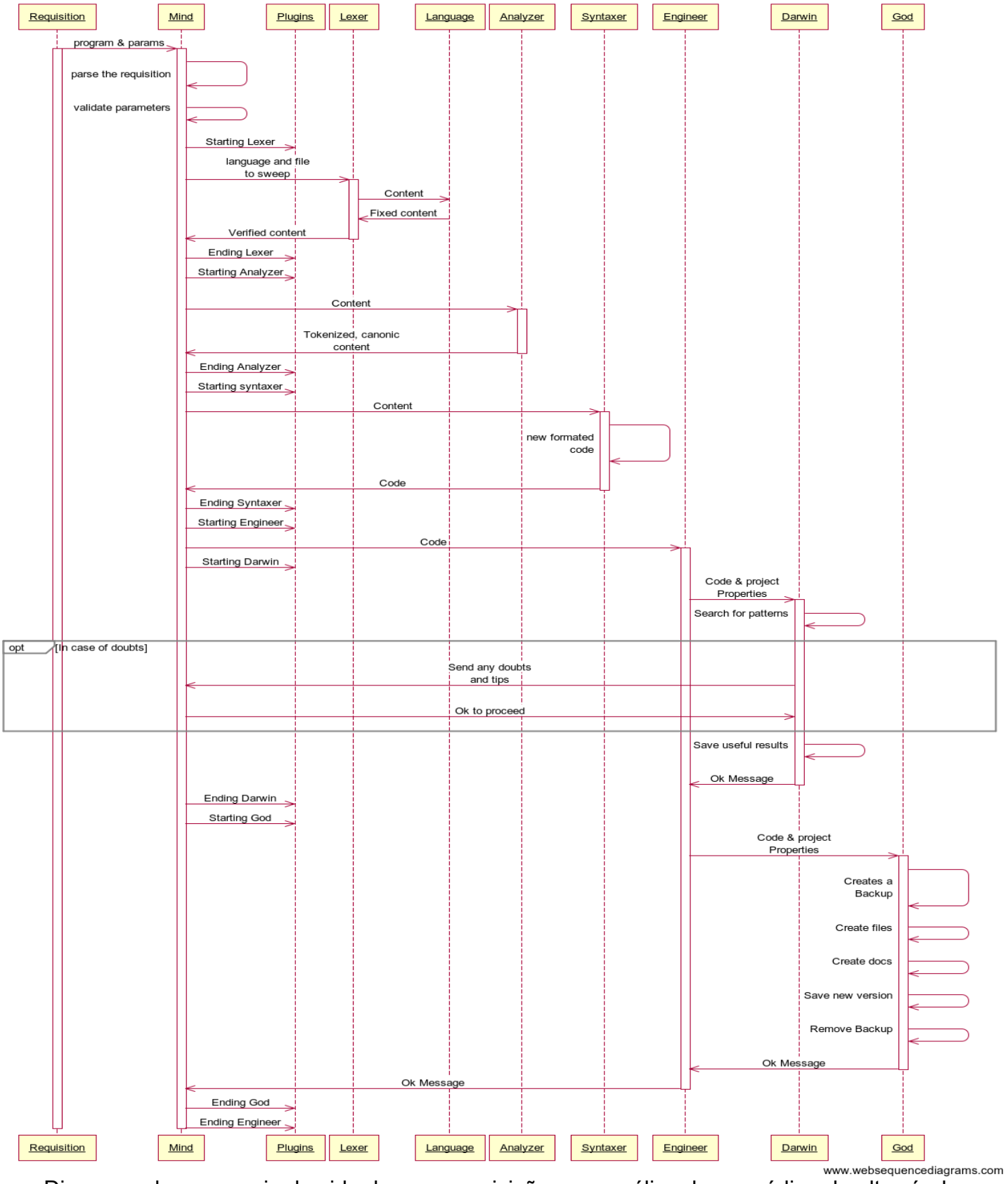
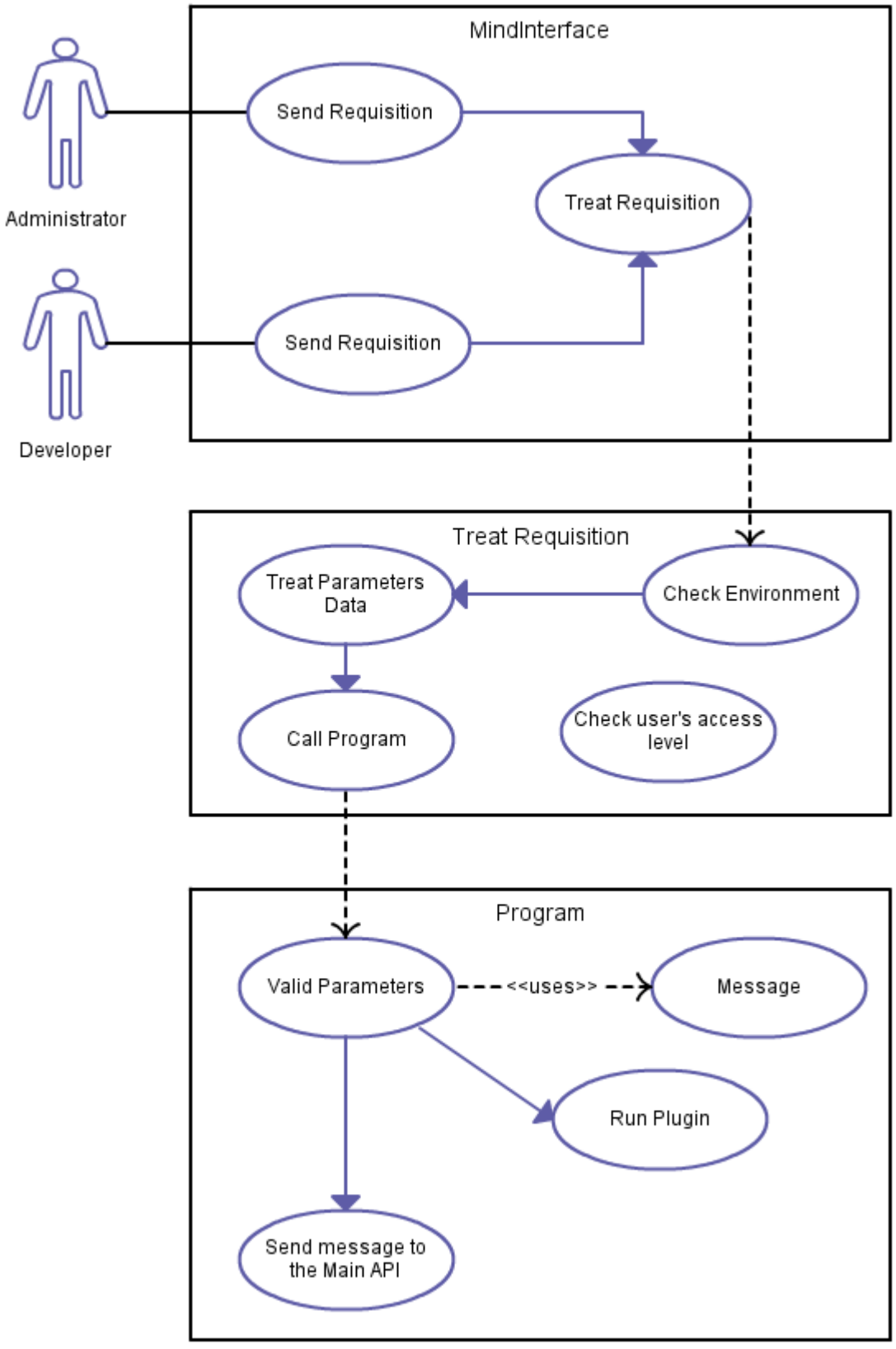


Diagrama de sequencia da vida de uma requisição para análise de um código de alto nível

- Diagrama de atividades

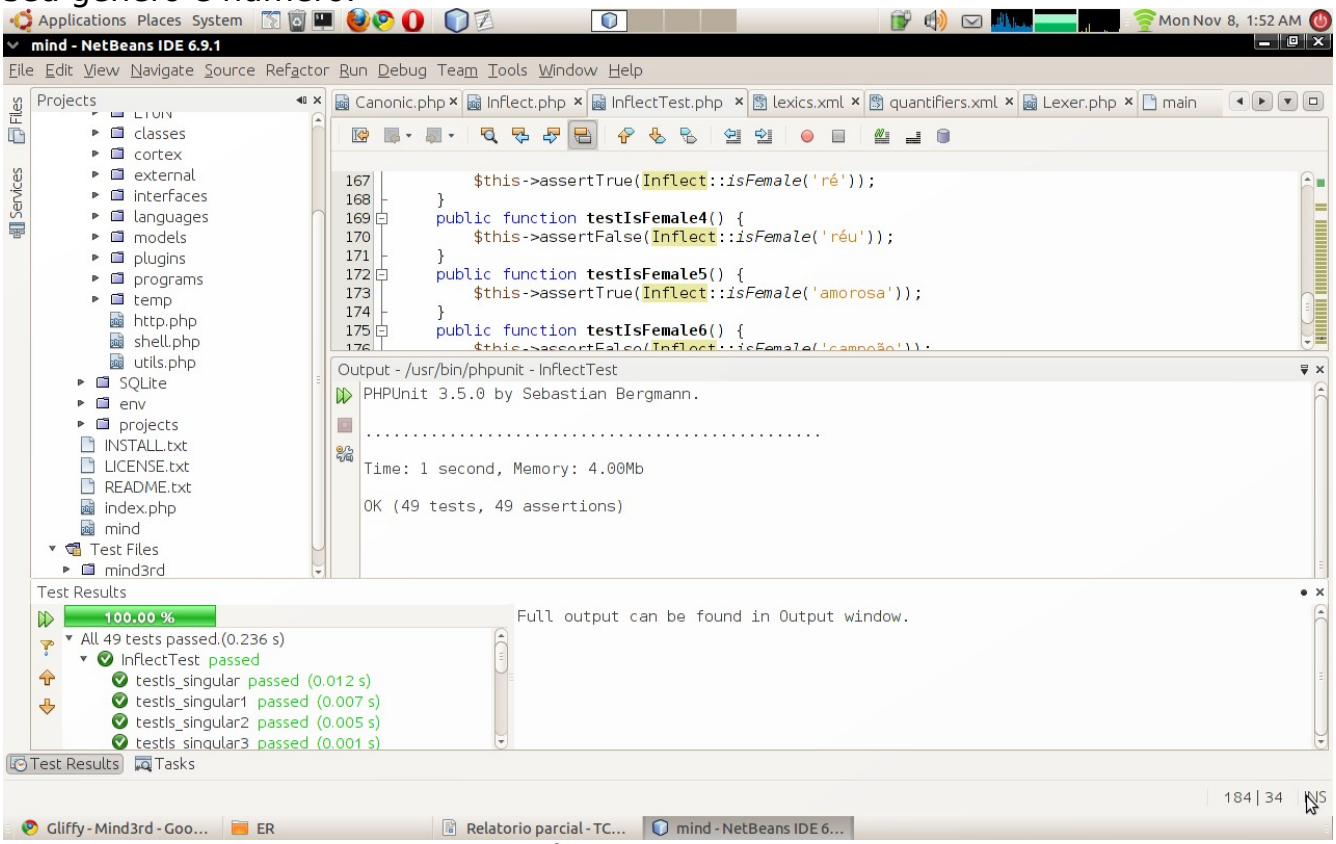


Validção:

A forma de validação a ser feita de duas maneiras.

Na primeira, haverá uma sequência de testes unitários automatizados a serem executado a fim de validar funcionalidades de classes e módulos. Num segundo momento, haverá um questionário para validação das opiniões de ao menos 10 usuário que tenham testado o novo core. Tais usuários serão analistas e desenvolvedores ativos na comunidade Open Source.

A primeira bateria de testes atinge 100% de acertos, executada sobre a classe Inflect para flexão de substantivos em português, alterando ou detectando seu gênero e número:



Esta imagem mostra o relatório dos testes executados a partir de uma IDE

Cronograma:

Atividade	Produto	Data	Detalhe
Entrega do Plano	Plano de trabalho	10/9/2010	
Início da estruturação	Treeview	15/9/2010	
Estruturação das classes	Diagrama de classes	30/9/2010	
Interfaces e testes	OO interfaces e classes de teste	20/10	Classes para testes e interfaces a serem implementadas pelas classes
Codificação parte I	Protótipo básico	10/12/2010	Gerenciador de projetos e usuários, e o básico do interpretador
Codificação parte II	Protótipo avançado	20/3/2011	Interpretador concluído
Codificação parte III	Gerador	30/4/2011	Gerador de códigos e documentação. Exemplos de integração com a API
Início das avaliações		30/4/2011	Entrega do protótipo final para os testadores
Término das avaliações	Relatórios e feedback	31/5/2010	

Referências:

-PHP, “PHP Documentation”
http://php.net

-Stallman, Richard M; Sussman, Gerald J (1977).”*Forward Reasoning and Dependency-Directed Backtracking In a System for Computer-Aided Circuit analysis*. Artificial Intelligence 9”. pp.135–196

-Micheal L. Scott, “Programming Language Programatics II”,
http://goo.gl/vZGb7

-Luiz A.M. Palazzo, “Introdução à programação Prolog”

-Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, “Copilers Principles, Techniques and Tools”

-Felipe Nascimento de Moura, “theWebMind Project Documentation”,
http://docs.thewebmind.org/

-Ted Briscoe, John Carroll, “Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars”

Componentes reutilizados:

-jQuery e jQueryUi, com alguns plugins para desenvolvimento da interface gráfica;

-Algumas bibliotecas da versão anterior do próprio sistema;

-PDO, drive para abstração da camada de banco de dados no PHP;

-SQLite e PHP-SQLite

-Symfony Console

-Biblioteca Inflection para flexão de subsantivos em inglês enconrada em http:php.net