UNIVERSIDADE ANHEMBI MORUMBI

GUSTAVO DE SOUZA OLIVEIRA

RODRIGO DE SOUSA LOPES

GUSTAVO CELSO TRIDA

LUCAS BAUR SANTANA

UAMBot – UMA PROPOSTA PARA CHATTERBOT ESPECIALISTA

São Paulo

2009

GUSTAVO DE SOUZA OLIVEIRA
RODRIGO DE SOUSA LOPES
GUSTAVO CELSO TRIDA
LUCAS BAUR SANTANA

UAMBot – UMA PROPOSTA PARA CHATTERBOT ESPECIALISTA

Trabalho de Conclusão de Curso apresentado como exigência parcial para a obtenção de título de Graduação do Curso de Sistemas de Informação na Universidade Anhembi Morumbi.

Orientador: Emerson dos Santos Paduan

São Paulo
2009

GUSTAVO DE SOUZA OLIVEIRA
RODRIGO DE SOUSA LOPES
GUSTAVO CELSO TRIDA
LUCAS BAUR SANTANA

UAMBot – UMA PROPOSTA PARA CHATTERBOT ESPECIALISTA

Trabalho de Conclusão de Curso apresentado como exigência parcial para a obtenção de título de Graduação do Curso de Sistemas de Informação na Universidade Anhembi Morumbi.

Aprovado em: 27 de Novembro de 2009

_____
Prof. Émerson dos Santos Paduan
Universidade Anhembi Morumbi

_____
Prof. Carlos Carneiro
Universidade Anhembi Morumbi

_____
Prof. Silvio Rocha Silva
Universidade Anhembi Morumbi

AGRADECIMENTOS

Agradecemos primeiramente a Deus que nos deu a força e a vida necessária para a conclusão deste trabalho e conseqüentemente deste curso.

Às nossas famílias, namoradas e noivas que souberam compreender todo o tempo que dispensamos nestes estudos.

Aos nossos professores, principalmente nosso professor orientador Emerson dos Santos Paduan, por compartilharem conosco seus conhecimentos e gastarem seu tempo com esta arte que é o ensino.

E por último à Universidade Anhembi Morumbi por nos proporcionar condições de concluirmos este curso com o título de bacharéis.

"Eu sou mais rápido que você. Sou mais forte que você. E com certeza, vou durar muito mais que você. Você pode pensar que eu sou o futuro, mas está errado. Você é o futuro. Se eu pudesse desejar alguma coisa, desejaria ser humano. Para saber o que significa ter sentimentos. Ter esperanças. Ter angústias. Dúvidas. Amar. Eu posso alcançar a imortalidade: basta não me desgastar. Você também pode alcançar a imortalidade: BASTA FAZER APENAS UMA COISA NOTÁVEL." (WALKER, 2006)

RESUMO

Este trabalho apresenta a construção de um *chatterbot* especialista baseado em banco de dados relacional capaz de aprender com agentes humanos. Traz o estudo de alguns dos *chatterbots* mais utilizados atualmente. O trabalho analisa conceitos de Inteligência Artificial utilizados na construção de *chatterbots*, principalmente o UAMBot, mostrando como essas técnicas são empregadas no desenvolvimento proposto. A proposta do trabalho é utilizar este chatterbot no *website* da Universidade Anhembi Morumbi, na sessão de atendimento ao aluno, onde o software deverá substituir, em médio prazo, agentes humanos na realização do atendimento. Espera-se com esta proposta reduzir custos e proporcionar maior disponibilidade no serviço de atendimento aos alunos.

Palavras-chaves: Chatterbot; Sistemas Especialistas; Inteligência Artificial.

ABSTRACT

This paper presents the construction of a specialist chatterbot based on relational data base in which is able to learn with human agents. It brings resource about some of the mostly used chatterbots currently. The paper analyzes Artificial Intelligence concepts used to build chatterbots, manly the UAMBot, showing how these techniques are used in the development propose. The propose of this paper is to use this chatterbot in the University Anhembi Morumbi‘s website, in the students area, where the software should replace in the middle term, human agents in activities related to assisting students. Expected with this propose to reduce costs and to offer high availability in the service of assisting students.

Keywords: Chatterbot; Specialized systems; Artificial Intelligence.

LISTA DE FIGURAS

LISTA DE SIGLAS

AIML	Artificial Intelligence Markup Language
ALICE	Artificial Linguistic Internet Computer Entitty
BOTMASTER	Robô Mestre
CONPET	Programa nacional da racionalização do uso dos derivados do petróleo e do gás natural
IA	Inteligência Artificial
JSP	Java Server Pages
MVC	Model View Control
SE	Sistemas Especialistas
UAMBOT	Universidade Anhembi Morumbi bot
UML	Unified Modeling Language
WEB	World Wide Web
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A Inteligência Artificial (I.A.) sempre despertou curiosidade nas mentes mais brilhantes que queriam entender como a capacidade humana, segundo alguns, criada por Deus, é inalcançável aos computadores. Mesmo depois de o homem ser vencido pela máquina em atividades como o Xadrez ou a resolução de problemas matemáticos complexos, é quase unânime o sentimento de que eles, os computadores, estão infinitamente longe de alcançar a criatividade, perspicácia e a inteligência emocional dos humanos. (LEVINE, 1988).

No entanto, o estudo da Inteligência Artificial nos últimos anos tem se mostrado cada vez mais comum, já que tarefas simples e repetitivas para seres humanos tornam-se não tão simples quando executadas por agentes dotados de Inteligência Artificial. O trabalho de construção desses agentes é alvo de pesquisas e objetivo de sujeitos que pretendem facilitar a vida humana em processos automatizados. (AUTORES, 2009)

Uma das áreas onde a I.A. tem parte importante no sucesso da operação é a de *chatterbots*: agentes robôs inteligentes capazes de sanar dúvidas, manter um padrão de conversação com pessoas, suprindo necessidades de respostas de um ser humano normal.

Este trabalho efetuará análises sobre as principais técnicas utilizadas na construção desse tipo de aplicação, demonstrando a importância da utilização de cada uma delas. Analisará alguns dos principais *chatterbots* demonstrando o funcionamento de cada um.

Diante da complexa, porém instigante área, o trabalho irá propor o desenvolvimento de um chatterbot especialista, que terá conhecimento sobre um único tema, mas que saberá responder questões completas, anteriormente elaboradas por humanos, sabendo entender as necessidades do usuário e visando sempre a satisfação do mesmo com o serviço prestado.

1.1 OBJETIVOS

O objetivo geral deste trabalho é propor o desenvolvimento de um chatterbot especialista para uso na Universidade Anhembi Morumbi.

O trabalho demonstrará os passos para o desenvolvimento utilizando uma base em banco de dados relacional. O funcionamento do sistema usará conceitos de Inteligência Artificial que serão abordados em capítulos posteriores.

O sistema proposto realizará o atendimento online aos alunos no site da instituição, substituindo o atendimento realizado via chat por atendentes humanos pelo atendimento automatizado do sistema.

O trabalho será um guia para o desenvolvimento dessa aplicação, demonstrando as técnicas utilizadas.

1.2 JUSTIFICATIVA

O crescimento de atendimentos online aumentou a insatisfação de usuários dessas ferramentas com as opções existentes devido ao despreparo dos atendentes, falta de informação e treinamentos, agilidade, disponibilidade. Os problemas dos usuários algumas vezes não são resolvidos. O trabalho descreve uma opção importante na melhoria da qualidade, automatização e redução de custos para esse tipo de caso.

Uma pesquisa informal realizada entre alguns alunos da Universidade Anhembi Morumbi, que se encontra anexa ao trabalho como apêndice A, aponta que mais de setenta por cento dos alunos não está plenamente satisfeita com o atendimento prestado pela universidade.

O fato de um robô ter um conceito de padronização dificilmente alcançado por seres humanos, torna essa solução algo promissor para o compromisso das empresas com a satisfação plena do cliente em um atendimento.

Pode ser destacado também o fato de que boa parte da imagem que as empresas mantêm frente aos usuários vem de quando estes têm problemas. Um cliente acaba por formar para si a imagem de uma empresa quando é atendido por ela na tentativa de resolução de algum problema. (AUTORES, 2009)

Portanto, prezando pela qualidade no atendimento das empresas o trabalho propõe uma forma de atendimento mais sofisticada.

Portanto, prezando pela qualidade no atendimento das empresas o trabalho propõe uma forma de atendimento mais sofisticada.

1.3 ABRANGÊNCIA

O trabalho tem como área de estudo os conceitos de I.A. necessários para o desenvolvimento do sistema, estudos de outros *chatterbots*, demonstrando vantagens e

desvantagens de cada um e uma breve comparação entre estes e o chatterbot proposto, o UAMBot.

O trabalho se limita a demonstrar testes realizados a título de prova de conceito onde não serão preocupações, a interface e a usabilidade.

O trabalho se compromete a testar as técnicas de inteligência artificial propostas, demonstrando a importância das mesmas para o alcance do objetivo do trabalho.

1.4 ESTRUTURA DO TRABALHO

No capítulo 2 são abordados os conceitos de I.A. utilizados na proposta para o desenvolvimento do UAMBot e do módulo de aprendizagem que será testado no trabalho.

No capítulo 3 são abordados o conceito de chatterbot e os principais *chatterbots* da atualidade, efetuando uma comparação entre os mesmos e o UAMBot, demonstrando seus pontos fortes e fracos.

No capítulo 4 será apresentado de forma mais detalhada o mais popular chatterbot utilizado na atualidade, o ALICE. Este que foi um dos primeiros *chatterbots* a ser desenvolvido e que até hoje mantém características muito à frente de outros *chatterbots* do mercado.

No capítulo 5 será abordada a proposta do trabalho: o UAMBot, um robô para utilização no atendimento via chat a usuários do site da Universidade Anhembi Morumbi. Serão apresentadas as técnicas de desenvolvimento demonstrando sua ligação com os conceitos de I.A. apresentados. Será demonstrada a arquitetura de funcionamento do sistema e testes de algoritmos criados, principalmente o de aprendizagem. Serão apresentados trechos do código desenvolvido no protótipo e explicações quanto ao seu funcionamento.

2 INTELIGÊNCIA ARTIFICIAL

Inteligência artificial (I.A.) é a maneira de fazer um computador pensar inteligentemente (LEVINE, 1988).

Para ser considerado um ser inteligente, o ser deve raciocinar, ou seja, determinar ações em cima de razões, ter percepção a casos diferentes, se adaptar de forma física e social, sentir e aprender. (STERNBERG, 1992)

Baseando-se nesse princípio, o objetivo é simular essas capacidades de forma artificial. Sendo assim, um indivíduo considerado inteligente como o homem, ao desenvolver algo que chegue perto dos seus próprios fundamentos, cria artificialmente sua imagem lógica.

2.1 SISTEMAS ESPECIALISTAS

Sistema Especialista (SE) é uma aplicação da I.A. São denominados assim porque possuem o conhecimento de um especialista na área de aplicação dentro de um programa. (STERNBERG, 1992)

Os engenheiros ou analistas de conhecimento seriam, então, os analistas de sistemas em Inteligência Artificial, que juntamente com o técnico designado pela empresa usuária, criariam o Sistema Especialista. (RICH, 1988)

Toda área profissional requer um especialista. É o caso da indústria, educação, medicina, comércio e da tecnologia. Sua utilização se sobressai especialmente em sistemas de apoio à decisão.

Os Sistemas Especialistas são baseados em regras, possuem conhecimento intensivo do domínio da aplicação, foram construídos por especialistas humanos e podem empregar aprendizado automático ou até mesmo criar raciocínio. Os Sistemas Especialistas tem dificuldade em lidar com conhecimento de senso comum, orientados à reutilização do conhecimento. (FÁVERO, 2009)

Quando é desenvolvido um processo de raciocínio de um sistema especialista, o sistema verifica qual a importância dos fatos que encontra, comparando-os com as informações já contidas no seu conhecimento sobre essas hipóteses. O processo formula novas hipóteses e verifica novos fatos que influenciam o processo de raciocínio. Este raciocínio é sempre baseado em seu conhecimento prévio acumulado. Um especialista, com esse processo de raciocínio, pode não chegar a uma decisão se os fatos de que dispõe para aplicar o seu conhecimento prévio não forem suficientes. É possível que, por este motivo, ele

possa chegar a uma conclusão errada, porém, este erro é justificado em função dos fatos encontrados e do conhecimento acumulado.

Os sistemas especialistas, para serem eficazes, precisam ter uma base de conhecimento e um processo de raciocínio consistente.

Os principais passos para a criação de uma base de conhecimento são: identificação e a conceituação. (CANUTO, 2008)

A identificação consiste em levantar pontos que serão relativamente importantes para consolidação da base de conhecimento.

A conceituação trabalha diretamente com o conhecimento do especialista. Esta etapa possui um alto nível de dificuldade pelo fato de que o conhecimento especializado é rico e complexo. Isso se deve ao fato de que o especialista domina os processos de sua área de atuação e muitas vezes se torna critico no detalhamento de seu conhecimento. Em um projeto de sistemas especialistas, onde são envolvidos mais de um especialista detalhando os processos, poderá ser comum o aparecimento de problemas na elaboração da base de conhecimento já que opiniões serão divididas podendo acarretar conclusões não definitivas.

A engenharia do conhecimento estuda como construir uma base de conhecimento. Essa engenharia fundamenta a aquisição de conhecimento em três etapas: Nível de conhecimento, Nível lógico e Nível de máquina.

Nível de conhecimento: é a aquisição de conhecimento em estado puro, linguagem natural, exemplo: táxi automático: A ponte Princesa Isabel Liga a Rua da Imperatriz à Rua Nova.

Nível lógico: Define-se na formalização, conhecimento codificado em sentenças, linguagem formal, exemplo: sentença lógica, liga Ponte - PI, RI, RN.

Nível de máquina: A implementação é a estrutura de dados representando as sentenças do nível lógico, exemplo: listas, tabelas, objetos entre outros. (CANUTO, 2008)

O processo de aquisição de conhecimento é outra fase importante na formação de uma base de conhecimento consistente. Este conhecimento pode ser adquirido de muitas maneiras como, livros, pesquisas e entrevistas com especialistas.

Alguns dos principais objetivos de se utilizar sistemas especialistas são: ter velocidade na determinação dos problemas, fundamentar decisões em uma base de conhecimento, ter segurança, ter um decréscimo na dependência de pessoal específico, integrar esse tipo de sistema com outras ferramentas e evitar interpretação humana de regras operacionais. (FÁVERO, 2009).

Alguns dos problemas enfrentados pelos Sistemas Especialistas são: a fragilidade, a falta de metaconhecimento e a validação. (DEPARTAMENTO DE INFORMÁTICA DA UNIVERSIDADE ESTADUAL DE MARINGÁ, 2004).

Os sistemas se tornam frágeis por somente terem acesso a conhecimentos específicos do seu domínio, portanto, não possuem conhecimentos mais genéricos quando alguma outra necessidade surge.

Quando o sistema não possui conhecimentos sofisticados sobre sua operação, ele não consegue desenvolver raciocínios lógicos sobre seu próprio escopo.

A validação de um sistema especialista esbarra na dificuldade de medição de seu desempenho, pois não se sabe quantificar o uso de conhecimento. (FÁVERO, 2009).

Algumas aplicações dos sistemas especialistas são: Diagnóstico, suporte on-line, controle de processos, controle de vôo, identificação de padrões difusos, medicina digital e aconselhamento jurídico.

Algumas das propriedades dos sistemas especialistas são: Autonomia, habilidade social, reatividade, iniciativa, continuidade temporal e orientação a objetivos.

2.2 TESTE DE TURING

A inteligência é a habilidade de obter um desempenho de nível humano em todas as tarefas cognitivas de forma a enganar um interrogador humano. (TURING, 1950).

Já em 1936, o matemático Alan Turing se questionava sobre os conceitos de inteligência e sobre a possibilidade de algum dia as máquinas poderem chegar a um nível de inteligência parecida com a dos humanos, já que as características de inteligência são atribuídas somente a seres humanos.

Assim, levantou diversos questionamentos, preocupações, e chegou a uma proposta para testar a veracidade de uma máquina quanto a sua inteligência artificial em 1950, denominada Teste de Turing. O teste consistia em colocar um humano através de um dispositivo de entrada e outro de saída, em locais onde não tivessem um contato direto e observar se o ser humano ao interagir com o outro lado, não conseguiria distinguir se estava interagindo com a máquina ou com outro ser humano, caso isso acontecesse, a máquina passaria no teste de Turing.

O computador seria interrogado por um humano através de algum tipo de rede, que na época, Turing sugeriu o teletipo.

O teste requer a capacidade de processamento de linguagem natural, representação do conhecimento, raciocínio automatizado e aprendizado de máquina.

As idéias de Turing são pontos determinantes até hoje na evolução da tecnologia da I.A., tanto em soluções reais como na ficção cientifica. Através desses fundamentos, vários especialistas têm suas crenças, alguns duvidam dos algoritmos de I.A., já que acreditam que todo algoritmo tem uma limitação, esses são denominados cientistas da inteligência artificial fraca. Outros, o grupo da inteligência artificial forte, acredita que não há um limite para esse assunto e que logo haverá ficção transformada em realidade.(LEVINE, 1988).

2.3 MINERAÇÃO DE TEXTO

A mineração de textos ou mineração de dados do texto é o processo de extração de informação de documentos. (ARANHA, 2006).

Com a mineração de textos é possível fazer uma análise geral do texto de entrada e considerar, através das técnicas aplicadas, somente o que é importante para a geração do resultado esperado.

O pré-processamento da mineração é composto de fases como leitura, extração e limpeza de termos, contagem dos termos e cálculo de freqüência.

2.3.1 *Stopwords*

Stopwords são termos freqüentes em um texto e que não carregam nenhuma informação de maior relevância. As *stopwords* são compostas por palavras das seguintes classes gramaticais: artigos, preposições, conjunções, pronomes e advérbios. (DIAS, 2004).

A remoção de *stopwords* tem como objetivo excluir termos que não são importantes ao documento, ou seja, mesmo sem esses termos é possível entender uma frase ou expressão. Esta operação também pode ser considerada como uma técnica de redução do número de palavras a serem analisadas e também o número de palavras a serem armazenadas em uma base de dados. (DIAS, 2004).

Para remover as *stopwords* é necessário criar uma lista, denominada stoplist, que conterá todas as palavras consideradas irrelevantes. Este processo é necessário para retirar do texto palavras que não tem nenhuma importância, diminuindo assim o tamanho das estruturas de indexação e facilitando a mineração. (BARION, 2008).

Como exemplos de *stopwords* podem ser citados: e, é, de, que, o, os, mas e ou.

2.3.2 Forma Canônica

A forma Canônica é a diversidade de representação de uma pergunta, frase ou sentença. Entradas de perguntas distintas devem ser relacionadas a uma mesma representação de significado. (DELUCCA, 2002).

Exemplo:

A Anhembi Morumbi oferece o curso de Sistemas de Informação?

O curso de Sistemas de Informação é oferecido pela Anhembi Morumbi?

A importância em aplicar um processo de stem para formas Canônicas é descobrir o assunto que a frase esta tratando e apresentar a resposta que faz mais sentido independente de como foi questionado, evitando conflitos de linguagem natural e interpretação. (DELUCCA, 2002)

Uma forma de analisar esse tipo de situação é determinar palavras chaves, assim é possível, através destas palavras, chegar ao assunto em questão, apresentando uma resposta coerente ao questionamento.

2.3.3 Indexação

Como o volume de informação cresceu rapidamente em poucos anos, foi necessário construir estruturas de dados especializadas para facilitar o acesso à informação armazenada.

Uma estrutura de dados, bastante antiga e usada, é uma coleção de palavras ou termos selecionados associado a ponteiros que se relacionam com a informação, ou documentos, chamado índice. (COLLISON, 1972).

Indexação refere-se ao acumulo de informações dentro de um registro para construir um índice de forma ordenada e que possa ser utilizado posteriormente para facilitar uma busca. (COLLISON, 1972).

A indexação pode ser usada de muitas formas. Para utilizar essa técnica é necessário entender o seu conceito.

A ordenação é feita seguindo um padrão formado pelos processos de selecionar, ler, identificar e extrair o que é valido, inserindo na base de dados para organização de um índice para futuras utilizações.

Com essa técnica pode-se marcar sinônimos e guardar erros de digitação, assim, quando forem inseridos na entrada, terão tratamentos específicos, economizando tempo e processamento.

3 CHATTERBOTS

O termo *chatterbot* surgiu da junção das palavras *chatter*, a pessoa que conversa, e da palavra *bot*, abreviatura de *robot*, ou seja, um robô, em forma de *software*, que conversa com pessoas. (TEIXEIRA, 2007).

Chatterbot é um *software* de computador que faz a simulação de uma conversa entre seres humanos. Tem como principal objetivo manter um diálogo em conversa natural de forma que as pessoas que estão interagindo com o robô, tenham a impressão de estar conversando com outra pessoa humana. Após os questionamentos realizados por humanos, o programa busca respostas plausíveis em uma base de conhecimento para respondê-lo.

3.1 TIPOS DE CHATTERBOTS

Existem diversos tipos de *chatterbots* que são usados para fins específicos.

Os *Academic Bots* são relacionados a assuntos acadêmicos, como sites de professores ou laboratórios de pesquisa.

Os *Design Bots*, usados em planejamento, possuem ferramentas e habilidades para a produção de outros bots e agentes inteligentes.

Os *Commerce Bots* desempenham atividades de comércio na Internet.

Os *Fun Bots* permitem aos usuários diversão através de jogos, ambientes virtuais, previsões e personagens de realidade virtual.

Os *Government Bots* buscam informações em sites governamentais.

Os *Knowledge Bots* congregam agentes inteligentes, de informação, de laboratório, cibernéticos, de web, e muitas ferramentas inteligentes de busca.

Os *News Bots* criam jornais personalizados e clips de artigos de jornais de todo o mundo.

Os *Search Bots* realizam buscas na Internet.

Os *Shopping Bots* fazem compras e comparações de preços para internautas.

Os *Stock Bots* monitoram o mercado de ações e enviam mensagens sobre os últimos preços e tendências.

Os *Update Bots* informam sobre novidades e materiais atualizados na internet. Avisam ao usuário quando um site especificado foi atualizado ou modificado.

3.2 ELIZA

O ELIZA foi o primeiro *chatterbot* da *web*. Foi criado em 1966 pelo alemão Joseph Weizenbaum no Instituto de Tecnologia de Massachusetts. (WEIZENBAUM, 1966).

Era um *chatterbot* simples, possuía apenas 204 linhas de código fonte, trabalhava com uma base de conhecimento pequena e não tinha uma linguagem e modelos apropriados.

O propósito de sua criação era manter um diálogo entre um paciente e um psicólogo.

Ao receber uma pergunta sobre um assunto desconhecido, apresentava uma resposta em forma de pergunta. Isto fazia com que a pessoa fornecesse mais detalhes sobre o assunto discutido, e assim, o programa poderia responder de forma inteligente e causar algum tipo de surpresa para as pessoas que estavam conversando, fazendo com que o usuário pensasse que estava sendo entendido.

Abaixo, um exemplo de conversa entre um usuário e o ELIZA:

ELIZA: "*So, how are you doing today?*"
("Então, como você se sente hoje?")
Usuário: "*I am tired."*
("Eu estou cansado")
ELIZA: *"Did you come to me because you are tired?"*
("Você veio me ver porque está cansado?")
Usuário: *"Yes, I feel alone."*
("Sim, me sinto só.")
ELIZA: *"Do you often feel alone?"*
("Você se sente só constantemente?")
(ELIZA, 2009)

3.3 CYBELLE

O CYBELLE foi o primeiro *chatterbot* com base de conhecimento em português. Foi criado em 2000 pela empresa francesa Agentland. (PRIMO, 2002).

Apresentando uma imagem virtual, CYBELLE é construída para ser uma espécie de referência no mundo dos agentes. Além de conversar com ela, é possível visualizar ao mesmo tempo, informações sobre agentes virtuais. CYBELLE também está integrada a um portal sobre agentes. A navegação no portal pode ser direcionada durante a própria conversa com o

robô por ela mesma ou pelo usuário. Assim, se a entrada for: “O que exatamente você é?”, ela, além de fornecer a resposta, ainda disponibiliza uma página com informações sobre sua personalidade. Se a entrada é uma pergunta sobre ALICE, por exemplo, CYBELLE mostra informações a respeito e ainda solicita a opinião do usuário sobre o *chatterbot*.

O CYBELLE não oferece aprendizagem, o que pode ser considerado como uma limitação do sistema, pois, muitos usuários já questionaram seus criadores acerca dessa limitação. Ela possui uma base de conhecimento pré-programada e, diante de alguns questionamentos, pode se mostrar repetitiva e imprecisa em suas afirmações. (PRIMO, 2002).

Atualmente, o CYBELLE encontra-se fora do ar.

3.4 EDBOT

O Ed é um robô criado para a Petrobras, pela empresa Insite Soluções para Internet. É resultado de mais de 10 anos de pesquisa e está no ar desde o dia 20 de outubro de 2004. (INSITE, 2009). A figura 1 mostra um exemplo de conversa com o Ed:

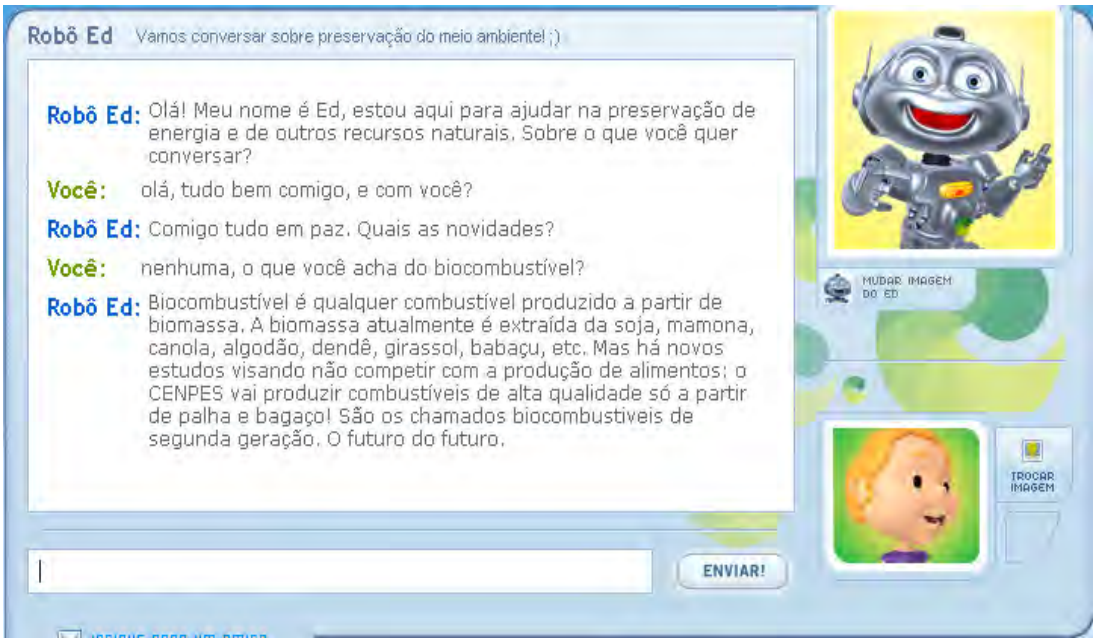


Figura 1: Exemplo de conversa com Ed. Fonte: (INSITE, 2009)

Este *bot* é capaz de conversar com um usuário como se fosse um atendente real e falar sobre assuntos ligados ao uso racional de energia, derivados de petróleo, meio ambiente, gás natural, dicas de economia, qualidade do ar, biocombustíveis, programas educacionais e fontes alternativas de energia.

Através do endereço http://www.conpet.gov.br/ed/ é possível conhecer e conversar com o Robô. O CONPET é o programa nacional da racionalização do uso dos derivados do petróleo e do gás natural e fornece o conteúdo que forma a base de conhecimento do Robô. (INSITE, 2009)

4 ALICE

O ALICE (*Artificial Linguistic Internet Computer Entity*) é um *chatterbot* criado na Lehight University por Richard S. Wallace. O desenvolvimento teve início em 1995 e foi reescrito em JAVA em 1998. (WALLACE, 2009).

O ALICE utiliza linguagem natural e inteligência artificial e é baseado em um experimento especificado por Alan M. Turing, em 1950.

4.1 FUNCIONAMENTO

O ALICE usa um padrão XML chamado AIML (*Artificial Intelligence Markup Language*) para a especificação das regras de heurística. Foi criado para processar linguagem natural, ou seja, é um programa que envolve robôs em uma conversa com os humanos.

O ALICE implementa o modelo de aprendizagem supervisionado, no qual o papel do *botmaster* é fundamental. O *botmaster* é o agente que monitora os diálogos, identifica as melhorias necessárias e cria novos conteúdos na forma de arquivos AIML, de tal forma que, as próximas respostas sejam mais apropriadas. (WALLACE, 2002).

Não existe nenhuma tecnologia sofisticada na concepção do ALICE. Ele não utiliza nenhuma tecnologia do tipo redes neurais, representação do conhecimento, busca em profundidade, algoritmos genéticos ou análise gramatical. (TEIXEIRA, 2005).

ALICE o primeiro programa de personalidade, ganhou o prêmio Loebner como “O computador mais humano no Teste de Turing nos concursos de 2000 e 2001”.
(WALLACE, 2003).

O ALICE é considerado o *chatterbot* mais utilizado e mais popular da atualidade e tem a maior base de conhecimento do mundo. O ALICE é uma extensão do ELIZA, mas, o ALICE é mais popular por possuir cerca de 40.000 categorias de conhecimento enquanto o ELIZA só possui 200. O ALICE possui sua fonte aberta, ou seja, permite aos usuários incrementarem o programa. (TEIXEIRA, 2005).

A maioria dos *chatterbots* foi criada utilizando o software do ALICE e utilizam a base de conhecimento implementada em AIML, fazendo apenas a tradução, a adição e a alteração de categorias.

4.2 CONHECIMENTO

Graphmaster é o nome dado ao conjunto de todos os nós que compõem o conhecimento do ALICE. Na verdade o *Graphmaster* é uma árvore que representa todo o conhecimento de um *chatterbot* baseado na tecnologia ALICE. Cada nó da árvore armazena uma única palavra ou os coringas.

Coringas são caracteres especiais que substituem partes da frase como o "_" para início de frase e o "*" para meio e fim de frase. Cada folha armazena a resposta ou os dados existentes entre as *tags*. O número de folhas da árvore equivale ao total de unidades de conhecimentos do ALICE. (TEIXEIRA, 2005).

Todo nó da árvore que contém o coringa "_", terá maior prioridade em uma busca nos nós de um mesmo nível, em seguida a prioridade será das palavras e finalmente a menor prioridade será do coringa "*".

Todas as unidades de conhecimento que começam com o coringa devem ser carregadas antes das perguntas que iniciam com palavras, e por último, devem ser carregadas as perguntas que iniciam com o coringa.

A busca de uma pergunta no *Graphmaster* é feita palavra por palavra e não pergunta por pergunta, cada palavra é armazenada em um nó da árvore.

Para localizar uma pergunta que inicie com a palavra "X", o sistema seguirá etapas conforme apresentado abaixo: (TEIXEIRA, 2005).

Caso o nó contenha o coringa, será feita a busca nos nós filhos deste, procurando palavras subseqüentes à palavra "X". Caso nenhuma palavra seja localizada, o sistema passa para a próxima etapa.

Caso o nó contenha a palavra "X", será feita a busca nos nós filhos deste, procurando palavras subseqüentes à palavra "X". Caso nenhuma palavra seja localizada, o sistema passa para a próxima etapa.

Caso o nó contenha o coringa "*", será feita a busca nos nós filhos deste, procurando palavras subseqüentes à palavra "X". Caso nenhuma palavra seja localizada o sistema retorna ao nó pai deste e substitui o valor do nó pela palavra "X".

Se o nó raiz contém o coringa "*" e logo em seguida há uma folha, então o algoritmo garante que uma resposta será fornecida.

Para localizar uma pergunta que inicie com a palavra "X", o sistema seguirá outras etapas conforme apresentado abaixo:

Caso o nó contenha o coringa "_",será feita a busca nos nós filhos deste, procurando palavras subseqüentes à palavra "X". Caso nenhuma palavra seja localizada, o sistema passa para a próxima etapa.

Caso o nó contenha a palavra "X", será feita a busca nos nós filhos deste, procurando palavras subseqüentes à palavra "X". Caso nenhuma palavra seja localizada, o sistema passa para a próxima etapa.

Caso o nó contenha o coringa "*", será feita a busca nos nós filhos deste, procurando palavras subseqüentes à palavra "X". Caso nenhuma palavra seja localizada, o sistema retorna ao nó pai deste e substitui o valor do nó pela palavra "X".

Se o nó raiz contém o coringa "*" e logo em seguida temos uma folha, então, o algoritmo garante que uma resposta será fornecida. (TEIXEIRA, 2005).

4.3 PONTOS FORTES E PONTOS FRACOS

O ALICE tem como pontos fortes sua grande base de conhecimento e sua grande percepção de inteligência que é muito parecida com a percepção humana. Outro ponto forte é a utilização da linguagem AIML, que permite uma popularização de idéias e flexibilidade de uso para leigos.

O ALICE tem como ponto fraco a forma de aprendizado, onde ainda não é possível uma atualização da base de dados em tempo real. Outro ponto fraco é a forma de tradução que, devido às diferenças entre o inglês e o português, algumas categorias foram retiradas da base de conhecimento devido à falta de tradução. Isso ocorre porque algumas expressões só existem em inglês, e não em português, além de existirem palavras em inglês que não possuem sinônimos em português. (NETO, 2004).

4.4 ARQUITETURA

ALICE é uma tecnologia de desenvolvimento que se baseia na interpretação de bases de conhecimento escritas na linguagem *Artificial Intelligence Markup Language* (AIML). (WALLACE, 2005).

Experiências com a ALICE indicam que aproximadamente 2.000 palavras atendem 95% das opções escolhidas pelas pessoas como a primeira palavra no início de uma frase, a partir da segunda, as opções diminuem bastante. (TEIXEIRA, 2005).

Com aproximadamente 41.000 unidades de conhecimento é possível estabelecer um bom diálogo. (TEIXEIRA, 2005).

A figura 2 apresenta uma visão geral do funcionamento do ALICE. A primeira etapa representa o encaminhamento da pergunta. Em seguida, o sistema realiza uma série de passos até que a pergunta fique pronta para a busca na base de conhecimento. Na segunda etapa, será feita a busca da pergunta na base de conhecimento contidas no arquivo AIML. Após a localização o sistema apresenta a resposta cadastrada.

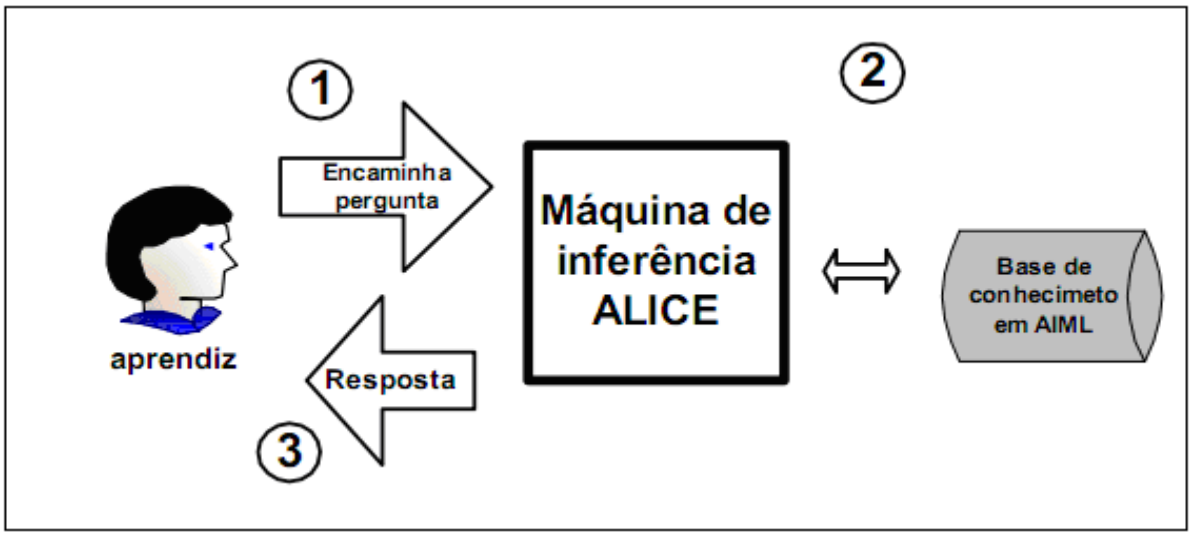


Figura 2: Arquitetura da tecnologia ALICE. Fonte: (TEIXEIRA, 2005).

4.5 AIML

AIML (*Artificial Intelligence Markup Language*) é uma linguagem desenvolvida para criar diálogos em uma linguagem natural por meio de softwares, simulando assim, inteligência humana. (WALLACE, 2009)

A linguagem AIML foi criada por Richard Wallace em uma comunidade de software livre entre os anos de 1995 e 2000. Através desta linguagem, foi criado o Alicebot. Alguns dos objetivos são: a linguagem deve ser de fácil aprendizagem e deve ser compatível com XML. (WALLACE, 2009)

A linguagem AIML é baseada em *tags* padrões e outras *tags* que podem ser customizáveis. (WALLACE, 2009)

Exemplos das principais *tags*:

<aiml>: a marca que começa e termina um documento AIML.

<category>: a etiqueta que marca uma unidade de conhecimento.

<pattern>: usada para conter um padrão simples, que corresponde ao que um usuário pode dizer ou digitar.

<template>: contém a resposta a uma entrada do usuário. (WALLACE, 2009).

5 UAMBOT

Este capítulo detalha as etapas do desenvolvimento de um protótipo. São ilustrados os principais requisitos, a especificação, a implementação, mencionando técnicas e ferramentas utilizadas, bem como a operacionalidade do protótipo.

5.1 REQUISITOS PRINCIPAIS DO PROTÓTIPO

Primeiramente serão apresentados os requisitos não funcionais. Alguns serão atendidos pelo protótipo e outros apenas pelo UAMBot.

Como requisitos não funcionais podem ser apresentados:

a) NF01: A interface com o usuário é de vital importância para o sucesso desse sistema. Esse sistema não será usado diariamente, portanto o tempo gasto pelo usuário para aprender a utilizar o software será mínimo. A interface será amigável, sendo bem receptiva a usuários iniciantes e não cansativa para usuários experientes. Esse requisito é de prioridade essencial.

b) NF02: O desempenho do sistema é uma preocupação Apesar de não ser um requisito essencial apresenta grande parte de responsabilidade no grau de satisfação do usuário. Quanto mais rápido ele for atendido, mais satisfeito ficará. Esse requisito é de prioridade importante.

c) NF03: O sistema será compatível com os mais usados navegadores de internet. É importante que o usuário não tenha problemas no acesso ao sistema usando qualquer navegador de internet. Portanto o sistema deverá ser implementado respeitando os padrões internacionais de programação WEB. Este requisito é de prioridade importante.

d) NF04: O tempo de resposta do sistema não deverá ultrapassar trinta segundos. Este requisito é de prioridade importante.

e) NF05: A base de dados deve ser protegida para acesso de apenas usuários autorizados. Este requisito é de prioridade essencial.

f) NF06: O software deve utilizar-se da sessão de autenticação do usuário no site da universidade. Deve estar em um local protegido por senha para que o acesso seja realizado apenas por alunos. Este requisito é de prioridade essencial.

Serão apresentados os principais requisitos funcionais do software:

a) RF01: O software deve sempre responder ao usuário, mesmo quando não tiver resposta em sua base de conhecimento. Este requisito é de prioridade essencial.
b) RF02: O software deve armazenar as perguntas que não tenham resposta para que sejam respondidas posteriormente. Este requisito é de prioridade importante.
c) RF03: O software deve desprezar na análise da frase as palavras contidas na lista de stop words. Este requisito é de prioridade essencial.
d) RF04: O software deve adicionar à base de conhecimento as perguntas que não tinham respostas e foram respondidas por atendentes. Este requisito é de prioridade importante.
e) RF05: O software deve sempre questionar o usuário quanto a sua satisfação após receber uma resposta. Este requisito é de prioridade importante.
f) RF06: O software deve sempre dar a oportunidade ao usuário de fazer uma nova pergunta caso queira. Este requisito é de prioridade essencial.
g) RF07: O software deve enviar perguntas e receber respostas de atendentes humanos para que possa responder aos usuários, no caso de perguntas que não tenham respostas na base de conhecimento. Este requisito é de prioridade essencial.
h) RF08: O software deve apresentar as respostas dos atendentes ao botmaster para que possam ser validadas para compor a base de conhecimento. Este requisito é de prioridade essencial.
i) RF09: O software deve indexar as perguntas por palavras chaves para que caso sejam formuladas distintamente por alunos possam ser respondidas normalmente. Este requisito é de prioridade essencial.
j) RF10: O software deve enviar respostas alternativas aos usuários caso não estejam satisfeitos com a primeira resposta enviada. Este requisito é de prioridade essencial. (AUTORES, 2009)

5.2 ESPECIFICAÇÃO DO PROTÓTIPO

Esta especificação foi desenvolvida utilizando a notação UML e a ferramenta utilizada foi a Microsoft Visio.

O sistema terá três atores. O usuário, que é o aluno da Universidade Anhembi Morumbi, o atendente, que será acionado quando o UAMBot não tiver a resposta para alguma pergunta e por último, o *botmaster*, que será o administrador da base de conhecimento do bot.

Serão apresentados os diagramas de casos de uso em suas formas resumidas para melhor visualização. A figura 3 é um diagrama que apresenta o usuário e suas possíveis interações com o sistema.

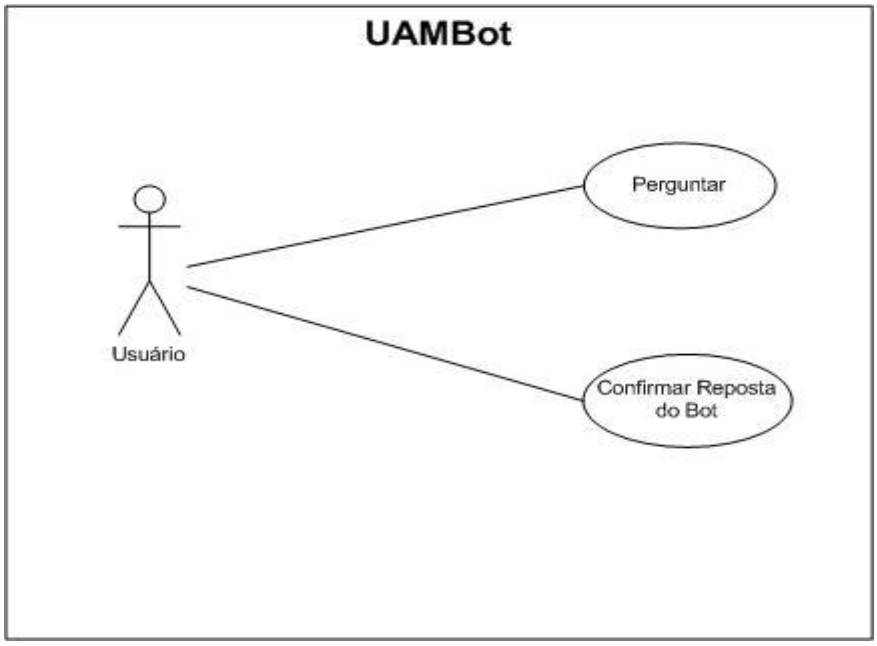


Figura 3: Usuário e possíveis interações com o sistema. Fonte: (AUTORES, 2009).

O usuário poderá interagir com o sistema realizando perguntas. Ao receber uma resposta, o usuário será questionado sobre sua satisfação quando à resposta apresentada. Essas respostas serão importantes para a melhoria da qualidade da base de conhecimento do sistema através das análises do *botmaster*. A figura 4 é um diagrama que apresenta o atendente e suas possíveis interações com o sistema.

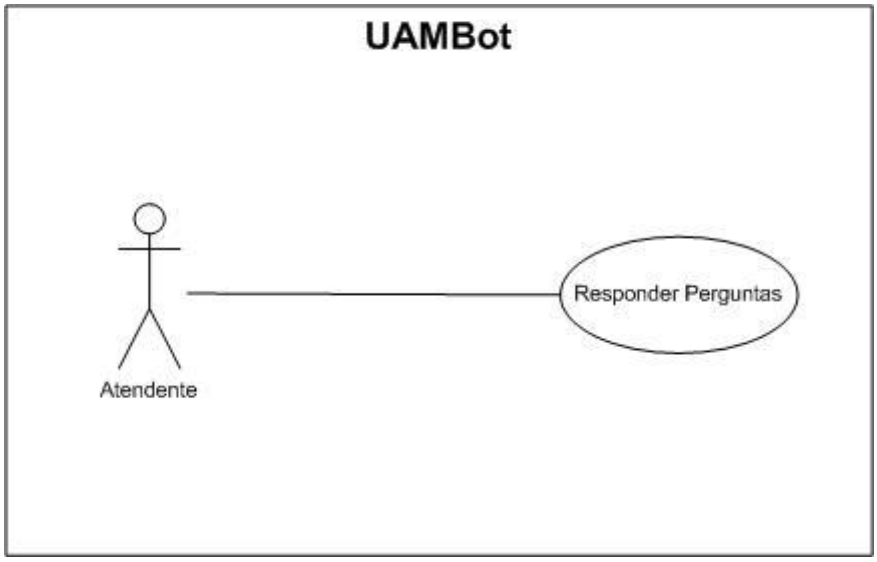


Figura 4: Atendente e possíveis interações com o sistema. Fonte: (AUTORES, 2009)

O atendente é a pessoa que auxilia o bot. Sempre que o sistema não encontrar resposta para alguma pergunta feita pelo usuário, guardará essa pergunta em uma tabela de perguntas pendentes. O atendente responderá as perguntas armazenadas nessa tabela. A figura 5 é um diagrama que apresenta o *botmaster* e suas possíveis interações com o sistema.

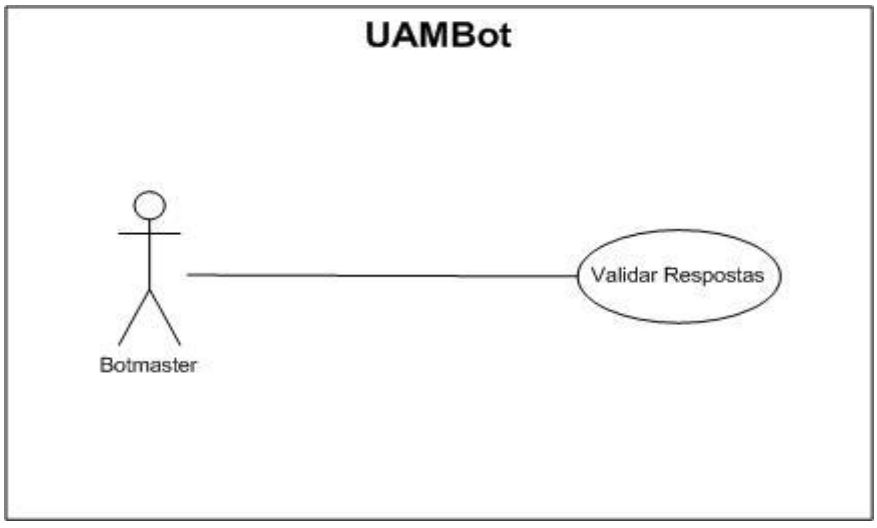


Figura 5: *Botmaster* e possíveis interações com o sistema. Fonte: (AUTORES, 2009).

O *botmaster* realizará a validação de todas as perguntas dos usuários respondidas por atendentes. O *botmaster* é o ator que manterá a qualidade da base de conhecimento do chatterbot analisando e validando todas as respostas que irão compor a base. Dessa forma, a

base de conhecimento do *bot* sempre será melhorada. A figura 6 mostra o fluxograma principal de funcionamento do UAMBot.

O usuário iniciará o atendimento e receberá uma frase de início que o induza a ser objetivo em sua pergunta. Por exemplo: “Bom dia, em que posso ajudar?”. O usuário realizará sua pergunta e receberá uma resposta do *chatterbot*.

Assim que o usuário realizar sua pergunta, o *chatterbot* buscará a resposta, como será detalhado no fluxograma de busca de resposta, e devolverá a resposta encontrada. Depois de devolver a resposta o *chatterbot* questionará o usuário sobre sua satisfação acerca da resposta apresentada. Caso a resposta seja positiva, o *chatterbot* questionará o usuário novamente, sobre sua possível necessidade de realizar outra pergunta. Caso a resposta seja negativa o atendimento estará encerrado. Caso o usuário desejar fazer outra pergunta o processo se reiniciará. Caso o usuário não esteja satisfeito com a resposta, o sistema realizará o fluxo de trabalho detalhado no fluxograma de busca de resposta alternativa. A figura 7 apresenta o fluxograma de busca de resposta detalhado.

Assim que o sistema receber uma pergunta, fará a análise inicial. Essa análise consiste em submeter a frase recebida a um filtro determinado pela lista de *stopwords*. A lista de stopwords proposta para esse protótipo encontra-se anexa ao trabalho como Anexo 2.

Depois de extraídas da frase as *stopwords*, restam apenas palavras consideradas palavras-chave. As palavras-chave serão a chave de busca para a resposta na base de conhecimento. A busca será realizada desrespeitando a ordem em que as palavras-chaves aparecem. Dessa forma, perguntas com formulação distinta, serão reconhecidas da mesma forma.

O *chatterbot* poderá encontrar em sua base de conhecimento uma ou mais respostas para a pergunta realizada. Na base de conhecimento as perguntas deverão ser armazenadas com um campo na tabela reservado para o número de utilizações positivas. Esse campo será um campo numérico que será incrementado sempre que o *chatterbot* utilizar essa resposta e ela for satisfatória ao usuário. Ou seja, sempre que depois de uma resposta dada, o usuário confirmar que está satisfeito, essa resposta recebe um incremento no campo de utilizações positivas.

O *chatterbot* retornará a resposta que tiver maior número de utilizações positivas.

O *chatterbot* poderá não encontrar uma resposta, nesse caso enviará a pergunta para a tabela de perguntas pendentes respondendo ao usuário, no caso de o atendimento estar sendo feito em horário comercial, a seguinte frase: “Um momento, estou buscando com os atendentes a resposta para a sua pergunta.”.

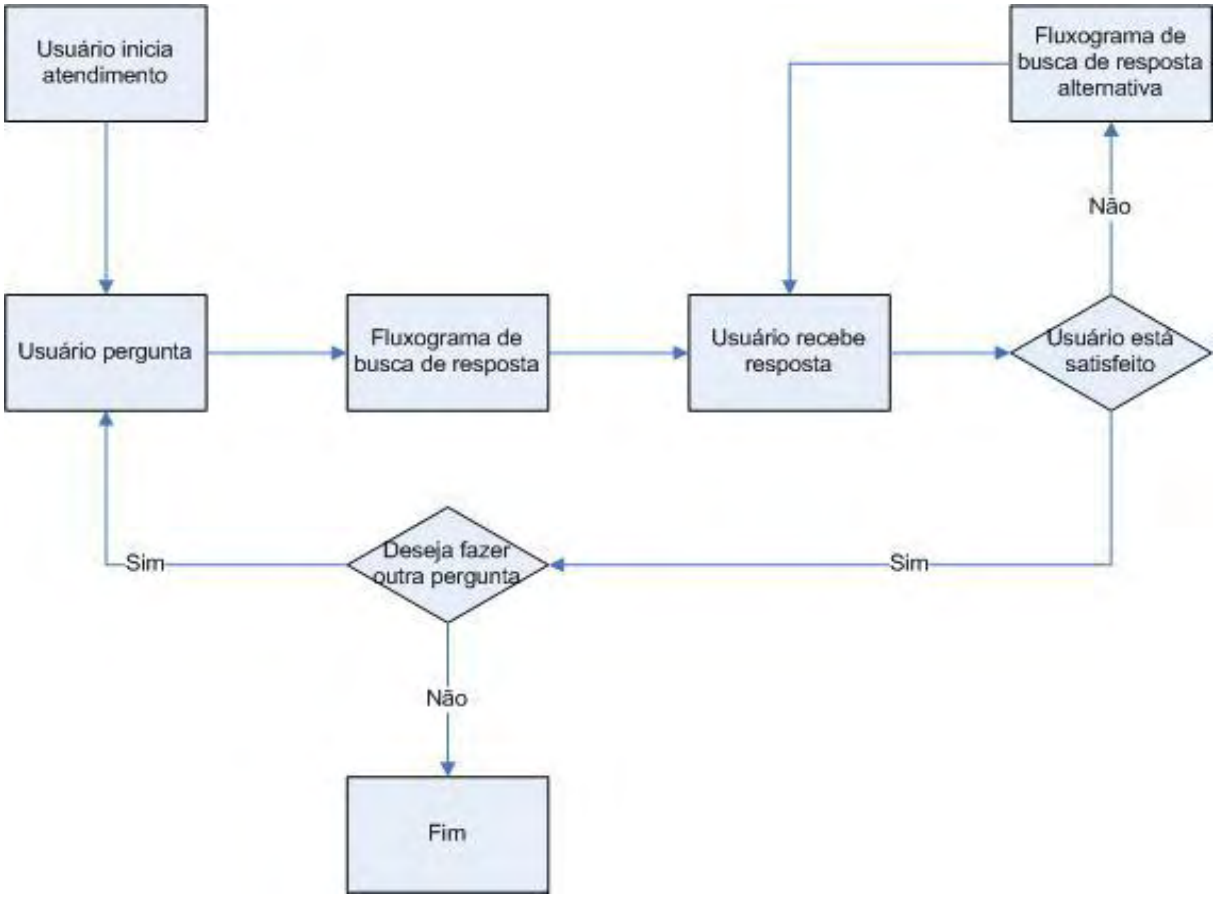


Figura 6: Fluxograma principal. Fonte: (AUTORES, 2009).

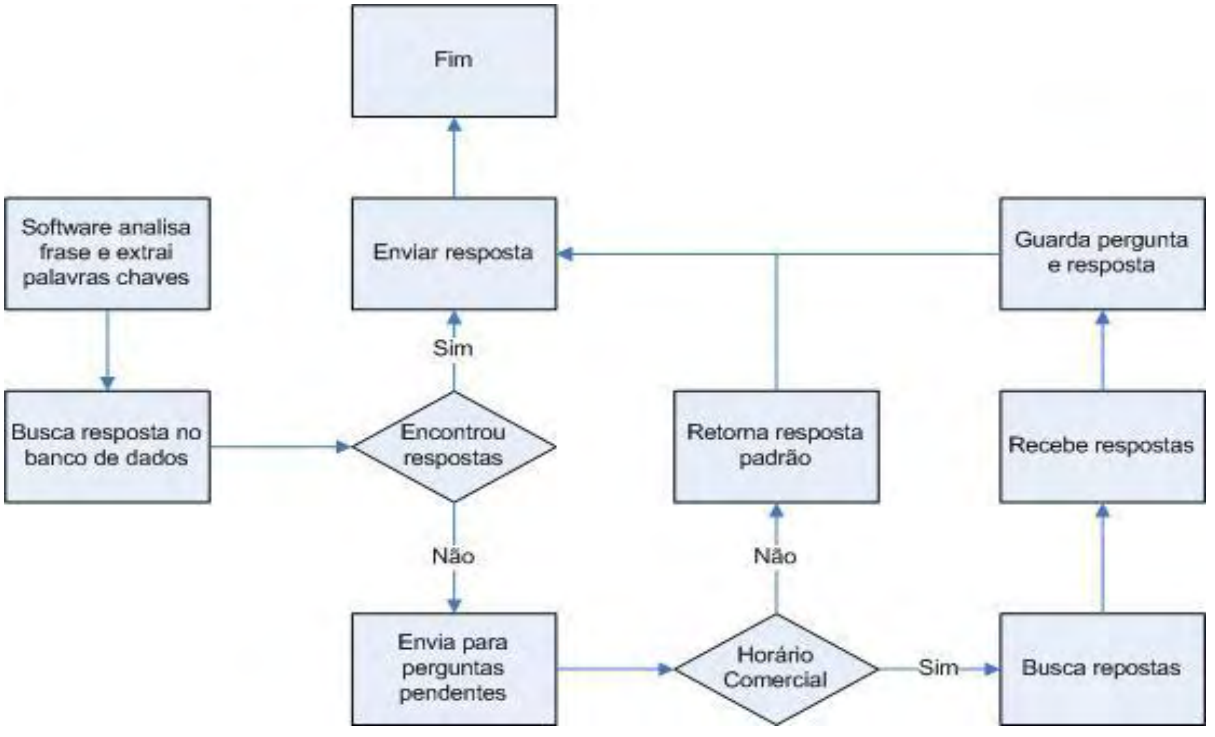


Figura 7: Fluxograma de busca de resposta. Fonte: (AUTORES, 2009).

Depois de enviada uma pergunta para a tabela de perguntas pendentes, o sistema esperará trinta segundos e buscará novamente a resposta para a pergunta. Caso encontre, retornará a resposta ao usuário. A pergunta e a resposta serão analisadas pelo *botmaster* posteriormente.

O sistema poderá não encontrar a resposta para uma pergunta pendente logo na primeira busca. Portanto realizará outras três buscas da mesma forma, respeitando um intervalo de trinta segundos. Sendo que, sempre que o sistema realizar nova busca, dará outra reposta ao usuário que está esperando como: “Mais um momento por favor.”

Depois de encontrada uma resposta guardará a pergunta e a resposta para que o *botmaster* possa analisar posteriormente e enviará a reposta ao usuário.

Caso o sistema não encontre a resposta para a pergunta mesmo depois de quatro tentativas, responderá ao usuário que entre em contato posteriormente.

O sistema poderá estar realizando um atendimento em horário não comercial. Nesse caso, caso não encontre uma resposta na base de conhecimento, retornará uma resposta padrão: “Não achei a resposta para essa pergunta e no momento não há ninguém aqui que possa ajudar. Peço que entre em contato em horário comercial. Deseja fazer outra pergunta?”.

A figura 8 apresenta o fluxograma de busca de resposta alternativa detalhado.

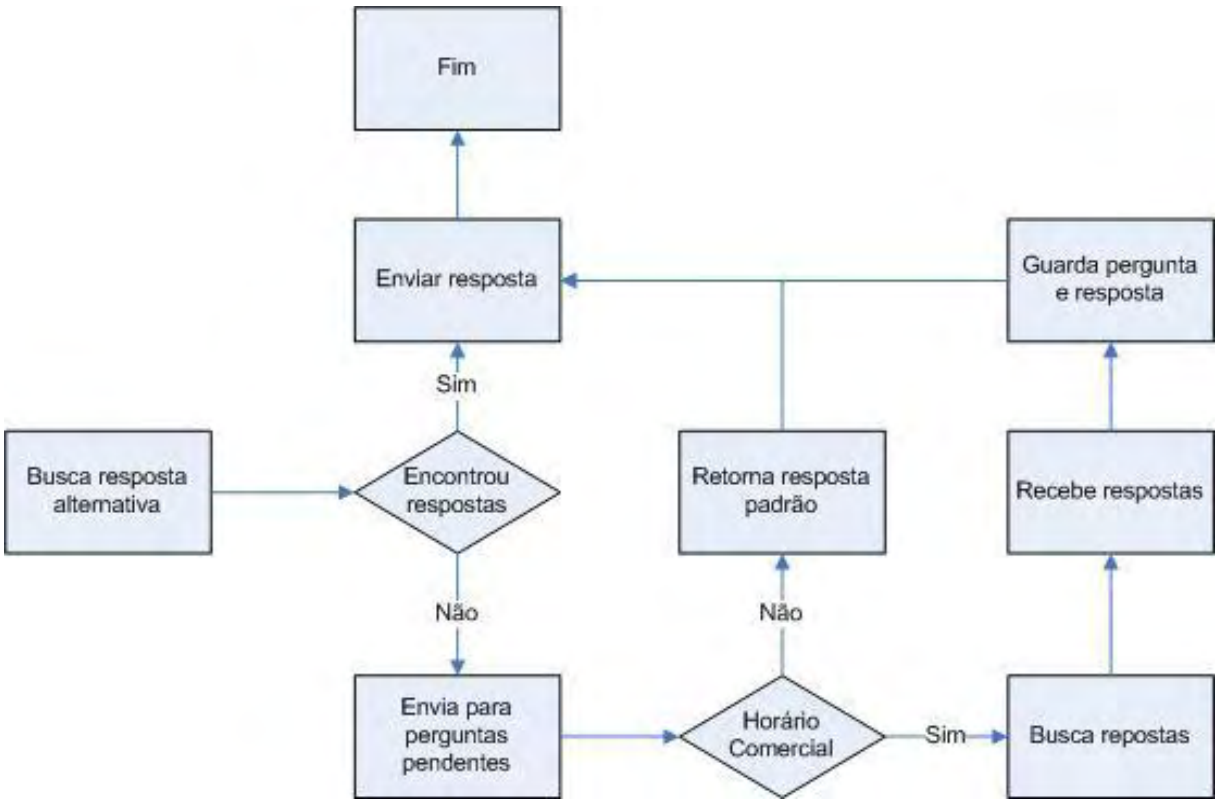


Figura 8: Fluxograma de busca de resposta alternativa. Fonte: (AUTORES, 2009).

O usuário poderá receber uma resposta e não ficar satisfeito. Nesse caso o *chatterbot* buscará uma resposta alternativa para o usuário.

O *chatterbot*, nesse momento, retornará a segunda resposta mais utilizada positivamente. E assim procederá até que se esgotem as respostas. Quando isso acontecer, o *chatterbot* enviará a pergunta para a tabela de perguntas pendentes.

Depois de enviada a pergunta para a tabela de perguntas pendentes, o sistema esperará trinta segundos e buscará novamente a resposta para a pergunta. Caso encontre, retornará a reposta ao usuário. A pergunta e a resposta serão analisadas posteriormente pelo *botmaster*.

O sistema poderá não encontrar a resposta para uma pergunta pendente logo na primeira busca. Portanto realizará outras três buscas da mesma forma, respeitando um intervalo de trinta segundos. Sendo que, sempre que o sistema realizar nova busca, dará outra reposta ao usuário que está esperando como: “Mais um momento por favor.”

Caso o sistema não encontre a resposta para a pergunta mesmo depois de quatro tentativas, responderá ao usuário que entre em contato posteriormente como “Não achei a resposta para essa pergunta e os atendentes estão ocupados. Peço que entre em contato mais tarde. Deseja fazer outra pergunta?”.

O sistema poderá estar realizando um atendimento em horário não comercial. Nesse caso, caso não encontre uma resposta na base de conhecimento, retornará uma resposta padrão: “Não achei a resposta para essa pergunta e no momento não há ninguém aqui que possa ajudar. Peço que entre em contato em horário comercial. Deseja fazer outra pergunta?”.

A figura 9 apresenta o fluxo de trabalho do atendente.

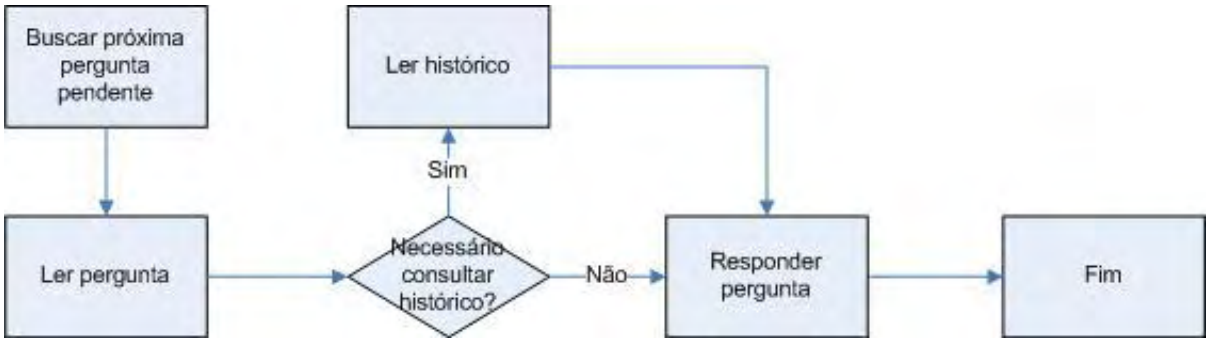


Figura 9: Fluxo de trabalho do atendente. Fonte: (AUTORES, 2009).

O atendente, ao se logar no sistema, buscará uma pergunta para ser respondida. O sistema retornará para o atendente a pergunta mais antiga ainda pendente. O atendente lerá a pergunta e caso não seja necessário consultar o histórico responderá a pergunta.

Se o atendente necessitar, poderá consultar o histórico da conversa entre o usuário e o *chatterbot*. Depois responderá a pergunta.

Ao terminar de responder uma pergunta o atendente buscará a próxima pergunta pendente para que possa proceder da mesma forma. A figura 10 apresenta o fluxograma do fluxo de trabalho do *botmaster*.

Figura 10: Fluxo de trabalho do *botmaster*. Fonte: (AUTORES, 2009)

O *botmaster*, ao se logar no sistema, buscará por uma próxima pergunta respondida por atendente. O sistema retornará a pergunta mais antiga respondida por um atendente e ainda não validada. O *botmaster* lerá a pergunta e a resposta e validará a pergunta para que entre na base de conhecimento do *chatterbot*. Esse processo visa dar maior confiabilidade à base de conhecimento do *chatterbot*.

Caso seja necessário, o *botmaster* poderá consultar o histórico de conversa entre o *chatterbot* e o usuário. Depois de ler o histórico poderá validar a pergunta para que integre a base de conhecimento.

Depois de validada uma pergunta pelo *botmaster*, o sistema integrará os dados à base de conhecimento e eliminará os dados da tabela de perguntas pendentes. Caso a pergunta seja invalidada pelo *botmaster*, o sistema eliminará a pergunta e a resposta da tabela de perguntas pendentes e não incluirá os dados na base e conhecimento.

5.3 BASE DE CONHECIMENTO

A base de conhecimento do *chatterbot* deverá ser implementada no mesmo banco de dados utilizado pelo site da Anhembi Morumbi. O banco de dados deverá ser um banco de dados robusto, com bom controle de acesso concorrente e boa velocidade de resposta. Indica-se o banco ORACLE, o SQL Server ou o Postgree para essa aplicação.

No entanto, para a implementação do protótipo, optou-se pelo uso do MYSQL, por ser um banco de dados leve e funcional, bem manejável e de fácil utilização. O MYSQL suporta bem os testes do protótipo, com uma base de conhecimento limitada e poucos acessos concorrentes.

O *chatterbot* terá uma base de conhecimento inicial. Para o protótipo a base inicial será bem reduzida, apenas para testes. Mas, para a proposta do UAMBot, a base de conhecimento inicial será mais robusta. Ela poderá ser retirada da sessão de perguntas mais freqüentes do *web site* da universidade. As informações propostas para a base de conhecimento inicial encontram-se anexas ao trabalho como Anexo A - Protocolos/Requerimentos/Solicitações.

A notação "ub_" foi escolhida para nomear as tabelas do banco de dados do *chatterbot*. Uma alusão ao "u" de UAM e ao "b" de bot. A figura 8 ilustra o diagrama de classes da base de conhecimento

A figura 11 apresenta as entidades que farão parte de toda a lógica de busca de respostas.

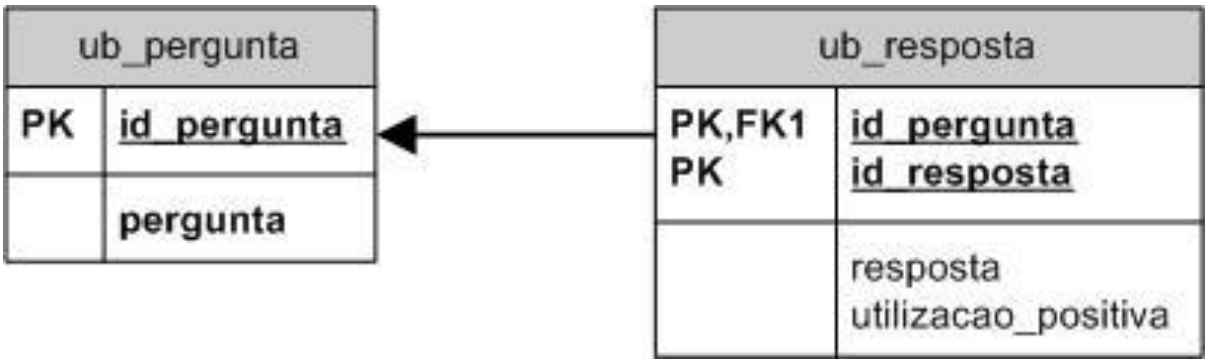


Figura 11: Entidades da base de conhecimento. Fonte: (AUTORES, 2009)

Para a tabela de perguntas existem os campos ID_PERGUNTA e PERGUNTA.

O campo ID_PERGUNTA será um campo numérico, de quatro posições, que será a chave da tabela de perguntas. Sempre que uma pergunta for inserida, terá como chave o primeiro número após o maior número encontrado na tabela no campo ID_PERGUNTA.

O campo PERGUNTA é o campo do tipo texto, de duzentas posições, que guardará as palavras chaves correspondentes a uma pergunta.

Para a tabela de respostas, existem os campos ID_PERGUNTA, ID_RESPOSTA, RESPOSTA e UTILIZACAO_POSITIVA.

O campo ID_PERGUNTA servirá para a ligação da tabela de respostas com a tabela de perguntas. Esse campo será uma chave estrangeira para o campo de mesmo nome na tabela de perguntas.

O campo ID_RESPOSTA, assim como o campo ID_PERGUNTA, será um campo numérico, de quatro posições, que servirá como chave da tabela de respostas. Essas tabelas permitem que uma pergunta tenha mais de uma resposta.

O campo RESPOSTA é um campo texto de quatrocentas posições, que guardará a resposta.

Por fim, o campo UTILIZAO_POSITIVA é um campo numérico, de seis posições, que conterá um número que identifique quantas vezes a resposta foi usada com sucesso. Ou seja, sempre que uma pergunta for aprovada pelo usuário, no questionamento realizado pelo *chatterbot* logo após responder ao usuário, o campo UTILIZACAO_POSITIVA será incrementado.

A figura 12 mostra o diagrama que apresenta outra parte da implementação do *chatterbot*. A parte da implementação que envolve os atores atendente e *botmaster*.

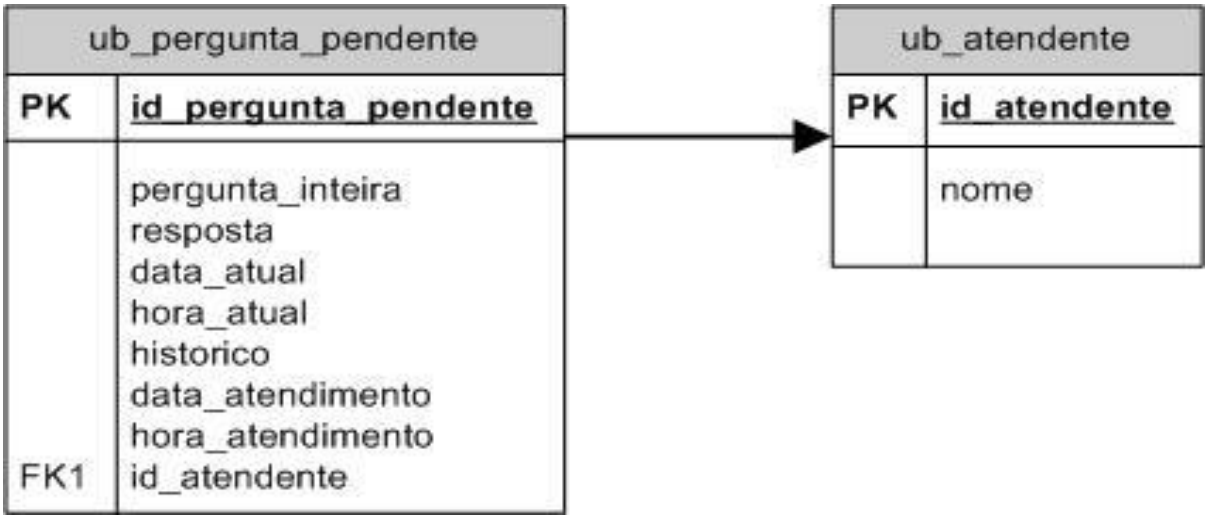


Figura 12: Entidades da evolução da base de conhecimento. Fonte: (AUTORES, 2009)

A tabela UB_PERGUNTA_PENDENTE é a tabela que armazenará todas as perguntas sem resposta na base de conhecimento. Nessa tabela temos os campos ID_PERGUNTA_PENDENTE, PERGUNTA_INTEIRA, RESPOSTA, DATA_ATUAL, HORA_ATUAL, HISTORICO, DATA_ATENDIMENTO, HORA_ATENDIMENTO e ID_ATENDENTE.

O campo ID_PERGUNTA_PENDENTE é um campo numérico, de quatro posições, que armazenará a chave da tabela. Cada nova pergunta pendente incluída ganhará como chave o número após o maior número encontrado na tabela no campo ID_PERGUNTA_PENDENTE.

O campo PERGUNTA_INTEIRA é um campo texto, de quatrocentas posições, que guardará a pergunta feita pelo aluno. A pergunta será armazenada inteira nesse campo, sem tratamento algum. O tratamento com o processo de reconhecimento de *stopwords* será realizado após o processo de resposta do atendente e de validação do *botmaster*. O motivo é que, para serem bem entendidas pelo atendente e pelo *botmaster*, as perguntas serão armazenadas como foram feitas.

O campo RESPOSTA é um campo texto de quatrocentas posições que armazenará a resposta dada pelo atendente. Caso o atendente ainda não tenha respondido a pergunta, esse campo estará vazio.

O campo DATA_ATUAL será um campo do tipo data que armazenará a data atual no momento da inclusão da pergunta na tabela. Esse campo é necessário já que a resposta às perguntas será feita na ordem em que elas foram feitas.

O campo HORA_ATUAL será um campo do tipo texto de oito posições que armazenará a hora exata em que a pergunta foi incluída na tabela. Esse campo, assim como o campo DATA_ATUAL é necessário devido à necessidade de classificação das perguntas pelo momento em que foram feitas, para que a reposta e a validação ocorram em um prazo aceitável ao usuário.

O campo HISTORICO será um campo texto de duas mil posições. Armazenará todo o histórico de conversa do usuário com o *chatterbot* até o momento em que uma pergunta foi feita e não tem resposta. Esse campo é importante para que atendente e *botmaster* possam entender o contexto da pergunta antes de respectivamente responder as perguntas ou validar as respostas.

O campo DATA_ATENDIMENTO será um campo do tipo data que armazenará a data em que a resposta foi dada pelo atendente.

O campo HORA_ATENDIMENTO será um campo texto de oito posições que armazenará a hora exata em que a pergunta foi respondida por um atendente. Esse campo, juntamente com o campo DATA_ATENDIMENTO serão importantes para futuros controles quanto ao tempo de atendimento aos usuários.

O campo ID_ATENDENTE será o campo que armazenará uma informação que identifique o atendente. É importante saber qual atendente realizou o atendimento de qual pergunta. Poderá ser o número de matrícula ou alguma outra informação que identifique o atendente. Dependerá da modelagem de banco de dados do *web site* da Anhembi Morumbi.

5.4 IMPLEMENTAÇÃO

A implementação das interfaces do *chatterbot* se dará em JSP. A implementação da lógica do *chatterbot* se dará em JAVA utilizando conceitos de MVC (Model View Control) *STRUTS*. Será utilizado o conceito de *multithread* para que uma grande quantidade de alunos possa ser atendida simultaneamente.

A escolha da linguagem JSP é fundamentada em suas vantagens quando a outras linguagens para o mesmo fim. O primeiro ponto é que a parte dinâmica do JSP, ou seja, a programação da lógica do software é realizada em JAVA. Por esse motivo, o JSP torna-se tão poderoso quanto o JAVA. Outro ponto é que o JSP tem uma separação natural entre a camada de apresentação e a camada de aplicação que ainda será reforçada pelo conceito de MVC utilizado no desenvolvimento do *chatterbot*.

A seguir, a lógica de implementação do fluxograma principal.

Depois de recebida uma pergunta, o primeiro passo será converter a pergunta para palavras maiúsculas. O segundo passo será analisar a pergunta e extrair as palavras chaves. Foi escolhida para a proposta uma lógica com base em *stopwords.* A frase será analisada e cada palavra que compor a lista de *stopwords* será excluída. Essas palavras não têm importância para a determinação do sentido da pergunta. A frase será divida em cada espaço, sendo que cada palavra ficará em uma posição de um vetor. Feito isso, cada palavra será consultada individualmente na lista de *stopwords*. Caso esteja na lista será excluída. As palavras restantes serão consideradas palavras chaves. Segue trecho de código em Java do protótipo exemplificando:

```
palavraChaveString = perguntar.getTxPergunta().replace("?", "").replace(",", "").split(" ");
```

Podem ser vistos alguns pontos de tratamento para pontuação da frase.

Já com as palavras chaves, o sistema buscará no banco de dados alguma pergunta que tenha essas palavras chaves em sua composição. O sistema buscará na tabela UB_PERGUNTA quais registros tem no campo PERGUNTA as palavras chaves da frase

formulada pelo usuário. Exemplo: Uma pergunta como “Como faço para fazer um pagamento atrasado?”, as palavras chaves comparadas serão “COMO FAÇO FAZER PAGAMENTO ATRASADO”. O sistema buscará na tabela, perguntas que foram indexadas anteriormente com essas palavras. A ordem em que as palavras chaves aparecem não será respeitada, já que uma pergunta pode ser feita de formas canônicas. O trecho de código seguinte ilustra o ponto que o protótipo busca as palavras chaves retirando as *stopwords* e busca no base de conhecimento a resposta:

```
for (int i = 0; i < palavraChaveString.length; i++) {

   if (stopWordsDao.test(palavraChaveString[i].toUpperCase()) == true) {

      palavraChaveString[i] = "";

   }

}
respostas = perguntaDao.read(palavraChaveString);
```

Com o retorno do banco de dados, o sistema usará o identificador da pergunta, o campo ID_PERGUNTA, para buscar na tabela UB_RESPOSTA as respostas correspondentes a essa pergunta. Com o retorno, essas respostas deverão ser ordenadas pelo conteúdo do campo UTILIZACAO_POSITIVA que marcará quantas vezes a resposta foi utilizada e aceita pelo usuário.

Depois de ordenadas, as respostas deverão ser retornadas em ordem decrescente. O próximo trecho de código do protótipo ilustra esse ponto:

```
pergunta = (Pergunta) respostas.get(0);
```

Quando a resposta for entregue ao aluno, e ele responder que está satisfeito com a resposta o sistema atualizará o campo UTILIZACAO_POSITIVA no banco de dados, incrementando o campo. O trecho de código a seguir ilustra esse ponto:

```
historico = historico + "\nUAMBot: " + "Você está satisfeito com essa resposta?";
```

Será apresentada a lógica para a alimentação da base de conhecimento.

Quando o aluno não está satisfeito com uma resposta, o *chatterbot* retornará respostas alternativas, que estarão armazenadas no vetor de retorno da busca no banco de dados. O vetor é importante para que haja um controle das respostas já dadas e para que possa atualizar a base de conhecimento incrementando o campo UTILIZACAO_POSITIVA. Caso as respostas se esgotem, e o usuário não esteja satisfeito, ou até mesmo se logo na primeira busca não seja encontrada nenhuma resposta, a seguinte lógica será implementada:

O *chatterbot* enviará a pergunta para a tabela de perguntas pendentes. O sistema alimentará a tabela UB_PERGUNTA_PENDENTE com os campos ID_PERGUNTA_PENDENTE, PERGUNTA_INTEIRA, DATA_ATUAL, HORA_ATUAL e HISTORICO. Para o campo ID_PERGUNTA_PENDENTE o sistema realizará uma busca na tabela pelo maior número que esteja nesse campo. O ID_PERGUNTA_PENDENTE da pergunta em questão será esse número acrescido de um. O campo PERGUNTA_INTEIRA receberá o valor da variável de tela que guarda a pergunta feita pelo usuário. Nesse momento, a pergunta não será tratada para que possa ser apresentada ao atendente conforme foi feita. O campo DATA_ATUAL receberá a data atual no momento que esse procedimento está sendo feito. O campo HORA_ATUAL receberá a hora atual em que esse procedimento está sendo feito. O campo HISTORICO receberá a variável de tela que guarda o histórico da conversa entre o chatterbot e o aluno. Os outros campos da tabela deverão ser preenchidos com espaço, já que esses campos serão preenchidos na hora do atendimento.

Quando o atendente se loga no sistema, solicita por meio de um clique em um botão, que apenas se tornará ativo quando existirem perguntas pendentes, uma nova pergunta para responder. Quando o atendente clica no botão, o sistema buscará na tabela UB_PERGUNTA_PENDENTE a pergunta que esteja com o campo RESPOSTA igual a espaço e que tenha a menor data e hora dentre as outras. Essa pergunta será carregada na tela do atendente para ser lida e respondida. Caso o atendente necessite, poderá clicar no botão ver histórico que carregará o conteúdo do campo HISTORICO na tela para que o atendente possa entender o contexto da pergunta. O histórico é carregado apenas se o atendente precisar, isso ocorrerá para que a tela de trabalho do atendente seja simples.

O atendente responderá a pergunta e clicará no botão que envia a resposta ao banco de dados. A tabela UB_PERGUNTA_PENDENTE será carregada com os campos faltantes RESPOSTA, DATA_ATENDIMENTO, HORA_ATENDIMENTO e ID_ATENDENTE. No campo resposta será carregado o conteúdo do campo de tela do atendente reservado à resposta. No campo DATA_ATENDIMENTO será carregada a data atual em que aconteceu o

atendimento. No campo HORA_ATUAL será carregada a hora exata do momento do atendimento feito pelo atendente. No campo ID_ATENDENTE será carregada uma informação do atendente que o identifique como único, que estará na variável de sessão do login do atendente.

O *chatterbot* terá guardado o identificador da pergunta pendente inserida e fará uma busca por esse campo de trinta em trinta segundos por quatro vezes verificando se o campo resposta está preenchido. Quando encontrar retornará ao aluno o conteúdo do campo RESPOSTA que estará preenchido com a resposta dada pelo atendente.

Depois de retornar ao aluno, o *chatterbot* o questionará se está satisfeito com a resposta. Caso o aluno esteja satisfeito o sistema perguntará se ele deseja fazer outra pergunta. Caso ele não esteja satisfeito, o *chatterbot* pedirá que o aluno repita a pergunta e enviará novamente para a tabela de perguntas pendentes.

Quando o *botmaster* se loga no sistema, solicita por meio de um clique em um botão, que apenas se tornará ativo quando existirem perguntas respondidas por atendentes, uma próxima pergunta para validar. Quando o *botmaster* clica no botão, o sistema buscará na tabela UB_PERGUNTA_PENDENTE a pergunta que não esteja com o campo RESPOSTA igual a espaço e que tenha a menor data e hora dentre as outras. Essa pergunta será carregada na tela do botmaster para ser lida e validada. Caso o *botmaster* necessite poderá clicar no botão ver histórico que carregará o conteúdo do campo HISTORICO na tela para que o botmaster possa entender o contexto da pergunta. O histórico é carregado apenas se o botmaster precisar, isso ocorrerá para que a tela de trabalho do *botmaster* seja simples.

O botmaster então validará ou invalidará uma pergunta. A resposta será carregada na tela do botmaster em um campo que permita alteração. O *botmaster* poderá alterar o que pense ser necessário e validar a resposta. Caso isso seja feito, o sistema tratará a pergunta do usuário para que possa integrar a base de conhecimento. Converterá a pergunta para palavras maiúsculas e depois analisará a pergunta e extrairá as palavras chaves. A frase será analisada e cada palavra que compor a lista de *stopwords* será excluída. A frase formada com as palavras que restarem, as palavras chaves ou índices, será o conteúdo do campo PERGUNTA da tabela UB_PERGUNTA.

O sistema alimentará as tabelas UB_PERGUNTA e UB_RESPOSTA.

Na tabela UB_PERGUNTA, primeiramente o sistema buscará se já existe uma indexação igual a que vai ser gerada. Se não houver, o sistema buscará o maior conteúdo no campo ID_PERGUNTA. Esse número acrescido de um será o ID_PERGUNTA inserido nesse processo. O campo PERGUNTA será o resultado da variável do tratamento anterior.

Na tabela UB_RESPOSTA o sistema carregará o campo ID_PERGUNTA com o mesmo conteúdo inserido no campo ID_PERGUNTA da tabela UB_PERGUNTA. O campo ID_RESPOSTA será carregado com o valor “1” já que é a primeira resposta para esse identificador de pergunta. O campo RESPOSTA será carregado com o conteúdo da variável de tela do *botmaster* reservada para a resposta, que poderá ter sido alterada por ele. O campo UTILIZACAO_POSITIVA será alimentado com o valor “1”.

Caso o sistema encontre na tabela UB_PERGUNTA outra pergunta com a mesma indexação buscará o ID_PERGUNTA dessa pergunta e atualizará apenas a tabela UB_REPOSTA. O conteúdo do campo ID_REPOSTA será o mesmo do campo ID_RESPOSTA da tabela UB_PERGUNTA encontrado. Buscará o maior número no campo ID_REPOSTA para esse ID_PERGUNTA, acrescerá um e utilizará no campo ID_RESPOSTA em questão. O campo RESPOSTA será carregado com o conteúdo da variável de tela do *botmaster* reservada para a resposta, que poderá ter sido alterada por ele. O campo UTILIZACAO_POSITIVA será alimentado com o valor “1”.

Exemplo de conversa com o protótipo do UAMBot numa situação onde a base de conhecimento contém a resposta.

UAMBot: Bom dia, em que posso ajudar?
Usuário: Bom dia, como faço para retirar a carteirinha da universidade?
UAMBot: Os alunos calouros (ingressantes no semestre vigente) não precisam solicitar a carteirinha da Universidade. Elas são entregues todo início de semestre pelo Banco Real. O local, data e horário de entrega são fixados pelo banco e divulgados em cartazes espalhados pelos campi.
UAMBot: Você está satisfeito com essa resposta?
Usuário: Sim, obrigado.
UAMBot: A Anhembi Morumbi agradece seu contato.

Exemplo de conversa com o protótipo do UAMBot numa situação onde a base de conhecimento não contém a resposta mas tem atendentes online.

UAMBot: Bom dia, em que posso ajudar?
Usuário: Bom dia, como faço para retirar a carteirinha da universidade?
UAMBot: Um momento, estou buscando com os atendentes a resposta para a sua pergunta.

UAMBot: Mais um momento por favor.
UAMBot: Os alunos calouros (ingressantes no semestre vigente) não precisam solicitar a carteirinha da Universidade. Elas são entregues todo início de semestre pelo Banco Real. O local, data e horário de entrega são fixados pelo banco e divulgados em cartazes espalhados pelos campi.
UAMBot: Você está satisfeito com essa resposta?
Usuário: Sim, obrigado.
UAMBot: A Anhembi Morumbi agradece seu contato.

Exemplo de conversa com o protótipo do UAMBot numa situação onde a base de conhecimento não contém a resposta e não há atendentes online.

UAMBot: Bom dia, em que posso ajudar?
Usuário: Bom dia, como faço para retirar a carteirinha da universidade?
UAMBot: Não achei a resposta para essa pergunta e no momento não há ninguém aqui que possa te ajudar. Peço que entre em contato em horário comercial. Deseja fazer outra pergunta?
Usuário: Não, obrigado.
UAMBot: A Anhembi Morumbi agradece seu contato.

O código desenvolvido para testes no protótipo em java, tanto para as partes lógicas quanto para as interfaces e partes de acesso a banco de dados estão anexas a esse trabalho como Apêndice B.

6 CONCLUSÃO

Espera-se que este trabalho tenha demonstrado os meios existentes para a implementação de um *chatterbot* especialista para a Universidade Anhembi Morumbi.

A partir de um levantamento das dúvidas mais freqüentes dos alunos que acessam o site da Universidade Anhembi Morumbi, uma base de conhecimento inicial foi criada.

O usuário sabe que está dialogando com um robô, mas devido ao conhecimento da base e aos algoritmos de mineração de texto usados, a qualidade no padrão de conversação é elevada, inibindo qualquer desconforto por parte do aluno de estar conversando com um robô.

Dentre as dificuldades encontradas, uma delas é a forma como o robô pensa e coleta informações. Outra dificuldade encontrada é a utilização de uma linguagem natural na gramática nativa. A maioria das bases de dados existentes trabalha com o idioma inglês.

Espera-se que o UAMBot obtenha sucesso na área de aplicação que é o site da Universidade Anhembi Morumbi e em outras áreas de aplicação principalmente na área comercial.

Espera-se também que o *chatterbot* seja estudado e melhorado para atender as necessidades dos usuários mais exigentes.

Como proposta para trabalhos futuros essa monografia propõe a criação de um *chatterbot* especialista para atendimento a pessoas que queriam estudar na universidade. Um *chatterbot* que tenha como base de conhecimento os procedimentos para entrar na universidade, os cursos oferecidos, a grade curricular de cada curso e informações gerais acerca da universidade.

Outra proposta para trabalho futuro é de criar bases de conhecimento para o UAMBot em outros idiomas, já que a universidade é uma universidade internacional e recebe todos os anos muitos alunos de outros países. Isso será importante para que os alunos sejam atendidos em suas línguas nativas. Uma base de conhecimento em inglês e outra base de conhecimento em espanhol seriam suficientes.

REFERÊNCIAS BIBLIOGRAFICAS

ARANHA, Christian; PASSOS, Emmanuel. A Tecnologia de Mineração de Textos. RESI-Revista Elerônica de Sistemas de Informação, Nº2, 2006. Disponível em: <http://revistas.facecla.com.br/index.php/reinfo/article/viewFile/171/66>. Acesso em: 01 Out. 2009.

CHAVES, Marcirio Silveira. *Um Estudo e Apreciação sobre Algoritmos de Stemming para a Língua Portuguesa*. Porto Alegre – RS. Agosto 2003. Disponível em: < http://xldb.di.fc.ul.pt/~mchaves/public/stemming.pdf>. Acesso em: 01 Out. 2009

COLLISON, Robert Lewis. *Índices e indexação: guia para indexação de livros e coleções de livros.* São Paulo: Polígono, 1972. 225p.

BARRETO, J.M. *Inteligência Artificial No limiar do Século XXI Abordagem Híbrida Simbólica, Conexionista e Evolucionária*. 3. ed. Florianópolis: Editora rrr, 2001.

CANUTO, Anne Magály de Paula. Sistemas baseados em conhecimento e Sistemas especialistas, Universidade Federal do Rio Grande do Norte, Natal, 2008. Disponível em: <www.dimap.ufrn.br/~anne/Aula%20SE.ppt>. Acesso em: 7 de Out. 2009

_____, _____. 2. Sistemas baseados em conhecimento e Sistemas especialistas, Universidade Federal do Rio Grande do Norte, Natal, 2008. Disponível em: <www.dimap.ufrn.br/~anne/Aula%20SE.ppt>. Acesso em: 7 de Out. 2009.

DEPARTAMENTO DE INFORMÁTICA DA UNIVERSIDADE ESTADUAL DE MARINGÁ. *Sistemas Especialistas.* 2004. Disponível em: <http://www.din.uem.br/ia/especialistas/basese.html>.Acessado em 01 Novembro 2009.

DE LUCCA, J. L.; NUNES, M.G.V. *Lematização versus Stemming. Série de Relatórios Técnicos do NILC* – ICM-USP, 2002. 16 p.

ELIZA. Disponível em: <http://www-ai.ijs.si/eliza-cgi-bin/eliza_script>. Acesso em: 22 de Mar. 2009.

FAVERO, Alexandre José. Tutorial sobre sistemas especialistas, Universidade Estadual de Maringá, Maringá – PR, Brasil. Disponível em: <http://www.din.uem.br/ia/especialistas/introdu.html> . Acesso em: 7 de Out. 2009.

_____, _____. 2. Tutorial sobre sistemas especialistas, Universidade Estadual de Maringá, Maringá – PR, Brasil. Disponível em: <http://www.din.uem.br/ia/especialistas/basese.html>. Acesso em: 7 de Out. 2009.

_____, _____. 3. Sistemas Especialistas. Universidade Estadual de Maringá, 2009. Disponível em: <http://www.din.uem.br/ia/especialistas/>. Acesso em 22 mar. 2008.

FEIGENBAUM, Edward A; Barr, Avron. *The Handbook of Intelligence - Vol I*: 1981

GALVÃO, A.; NEVES, A.; BARROS, F. *Persona-AIML: Uma Arquitetura para Desenvolver Chatterbots com Personalidade*. XXIII Congresso da Sociedade Brasileira de Computação – CSBC – 2003, Campinas, SP, 2003.

INSITE. Disponível em: <http://www.insite.com.br/>. Acesso: em: 7 de Out. 2009.

_____. 2. Disponível em: <http://www.inbot.com.br/novo/cases.php>. Acesso: em 7 de Out. 2009.

JENNINGS N. R; WOOLDRIDGE, M. *Applications of Intelligent Agents*, Queen Mary & Westfield College University of London. - 1996

LACHI, R. L. Chapa: *Um Agente de Interface para Ferramentas de Bate-papo em Ambientes de Ensino a Distância na Web.* Campinas, SP, 2003.

LAUREANO, E. A. G. C. ConsultBot - *Um Chatterbot Consultor para Ambientes Virtuais de Estudo na Internet*. Recife, PE, 1999.

LEVINE, R. I.; DRANG, D. E.; EDELSON, B. *Inteligência Artificial e Sistemas Especialistas*. São Paulo: Mcgraw-hill, 1988. LOEBNER, Hugh. *Reflections on the Loebner Competition.* DARTMOUTH 2000.

NEVES, A. M. M. iAIML: Um Mecanismo para o Tratamento de Intenção em Chatterbots. 2005. Tese (Doutorado) - Centro de Informática, Universidade Federal de Pernambuco, 2005. Disponível em: <http://www.sbc.org.br/bibliotecadigital/download.php?paper=355>. Acesso em: 29 maio 2009.

RABELLO, Roberto dos Santos. *Inteligência Artificial - Quebrando paradigmas*. 24 Fev. 2005. Disponível em: <http://www.universia.com.br/materia/materia.jsp?materia=6310>. Acesso em: 28 Set. 2009.

RICH, Elaine. *Inteligência Artificial*. Editora McGRAW HILL, 1988.

STERNBERG, Robert. As capacidades Intelectuais Humanas: Uma abordagem de Processamento da Informação. Porto Alegre: Artes Médicas, 1992.

TEIXEIRA, S.; MENEZES, C. S. CHATTERBOT: uma ferramenta para motivar estudantes de cursos a distância. Revista Aprender Virtual, Marília, SP, 2003. Disponível em: <http://www.multicast.com.br/sergio/amcorabot-revista-aprender.pdf>. Acesso em: 29 maio 2009.

_____, _____. 2. *Facilitando o uso de Ambientes Virtuais através de Agentes de Conversação*. XIV Simpósio Brasileiro de Informática na Educação - SBIE - 2003, Rio de Janeiro, RJ, Brasil, 2003. Disponível em: <http://www.nce.ufrj.br/sbie2003/publicacoes/paper48.pdf>. Acesso em: 29 maio 2009.

TEIXEIRA, S.; RAMIRO, T. B.; OLIVEIRA, E.; MENEZES, C. S. *Chatterbots em ambientes de aprendizagem – uma proposta para a construção de bases de conhecimento*. XXV Congresso da Sociedade Brasileira de Computação – CSBC – 2005, São Leopoldo, RS, Brasil, 2005. Disponível em: <http://www.multicast.com.br/sergio/tuxbot-artigo-sbc2005.pdf>. Acesso em: 29 maio 2009.

TEIXEIRA, Sérgio. Chatterbots – Uma Proposta Para A Construção de Bases de Conhecimento, Vitória, 2005.

WALLACE, R S. ALICE Silver Edition. Disponível em: <http://www.alicebot.org>. Acesso em: 29 maio 2009.

_____, _____. 2. *Don't read me - A.L.I.C.E. and AIML documentation*. Disponível em: <http://www.alicebot.org/articles/wallace/dont.html>. Acesso em: 29 maio 2009.

_____, _____. 3. *The Elements of AIML Style*. Disponível em: <http://www.alicebot.org>. Acesso em: 29 maio 2009.

_____, _____. 4. *Be your own botmaster* . Disponível em: <http://www.alicebot.org>. Acesso em: 29 maio 2009.

_____, _____. 5. *AIML Overview*. 2009. Disponível em: <http://www.pandorabots.com/pandora/pics/wallaceaimltutorial.html>. Acesso em 01 Out. de 2009.

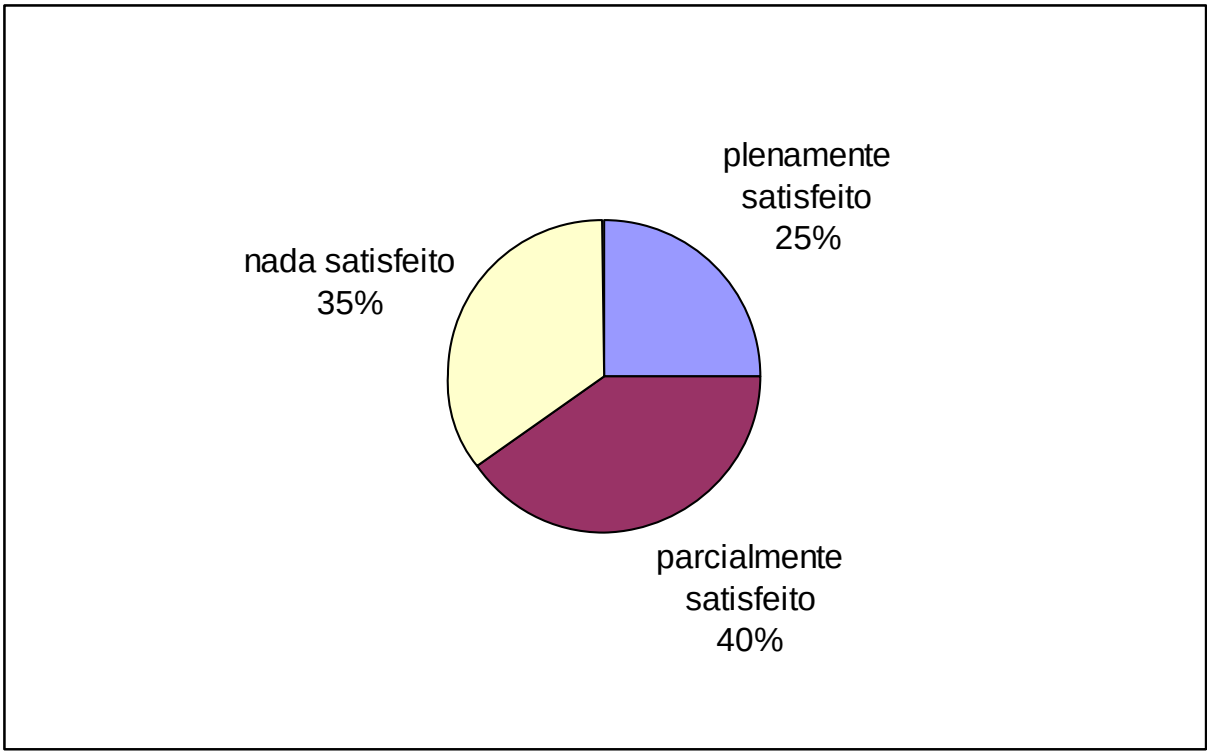
_____, _____. 6. *The Anatomy of A.L.I.C.E.* Publicado em: 24 Outubro 2002. Disponível em: <http://www.alicebot.org/anatomy.html>. Acesso em 20 Out. 2009.

WEIZENBAUM, J. Eliza. *A Computer Program For the Study of Natural Language Communication Between Man and Machine*. Communications of the ACM Volume 9, Number 1, 1966.

APÊNDICE A - PESQUISA DE SATISFAÇÃO

Pesquisa realizada com vinte alunos do curso de sistemas de informação com ênfase em banco de dados no período de primeiro de dezembro de dois mil e nove a sete de dezembro de dois mil e nove.

Os alunos foram questionados quanto a sua satisfação em relação ao atendimento prestado pela universidade como: plenamente satisfeitos, parcialmente satisfeitos ou nada satisfeitos.



APÊNDICE B - CÓDIGO DO PROTÓTIPO

Código desenvolvido para testes do protótipo da proposta. Será apresentado por arquivos, primeiramente os arquivos de interface e posteriormente os arquivos java.

index.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<form action="Usuario.do">
   <input type="submit" value="USUARIO" />
</form>
<form action="Botmaster.do">
   <input type="submit" value="BOTMASTER" />
</form>
<form action="Atendente.do">
   <input type="submit" value="ATENDENTE" />
</form>
```

usuario.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic" %>

<html:html locale="true">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title><bean:message key="welcome.title"/></title>
      <html:base/>
```

```
</head>
<body style="background-color: white">

    <logic:notPresent name="org.apache.struts.action.MESSAGE" scope="application">
        <div  style="color: red">
            ERROR:  Application resources not loaded -- check servlet container
            logs for error messages.
        </div>
    </logic:notPresent>

    <table width="150" border="1">
        <tr>
            <td><div align="center"><img src="UAMBot.jpg" width="185"
height="243"></div></td>
        </tr>
        <tr>
            <td>

                <p>
                    <textarea name="txHistorico" cols="150" rows="15" disabled="true">
            <%
    String historico = (String) (session.getAttribute("historico"));

    if (historico == null) {
        historico = "\nBom dia, em que posso ajudar?";
    }

    out.print(historico);
    %>
                </textarea></p>
            </td>
        </tr>
        <tr>
            <td><div align="center">
```

```
                <form name="formPerguntar" action="perguntar.do">
                  <p align="left">                     </p>
                  <p align="center">
                    <input name="txPergunta" type="text" value="digite sua pergunta"
size="150" maxlength="150" />
                  </p>
                  <p>
                    <input type="submit" value="Enviar" name="btnEnviar" />
                  </p>
                </form>
              </div></td>
            </tr>
        </table>
        <p> </p>
    </body>
</html:html>
```

Action.PerguntarAction.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Action;

import Dao.PerguntaDao;
import Dao.PerguntaPendenteDao;
import Dao.StopWordsDao;
import Form.PerguntarForm;
import Model.Pergunta;
import Model.PerguntaPendente;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
```

```
import java.util.Date;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

/**
 *
 * @author Rodrigo
 */
public class PerguntarAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response) throws Exception {

        String palavraChaveString[] = null;

        String historico = "";

        PerguntaDao perguntaDao = new PerguntaDao();

        StopWordsDao stopWordsDao = new StopWordsDao();

        Pergunta pergunta = new Pergunta();

        PerguntaPendente perguntaPendente = new PerguntaPendente();

        PerguntaPendenteDao perguntaPendenteDao = new PerguntaPendenteDao();

        ArrayList respostas;
```

```
int lastId;

DateFormat data = new SimpleDateFormat("yyyy-MM-dd");
DateFormat hora = new SimpleDateFormat("HH:mm:ss");

Date dataSistema = new Date();

HttpSession session = request.getSession();

PerguntarForm perguntar = (PerguntarForm) form;

String questionamento = "";

questionamento = (String) session.getAttribute("questionamento");

historico = (String) session.getAttribute("historico");

if (historico == null) {
   historico = "\nBom dia, em que posso ajudar?";
}

historico = historico + "\nAluno : " + perguntar.getTxPergunta();

if (questionamento == null) {

   palavraChaveString = perguntar.getTxPergunta().replace("?", "").replace(",",
"").split(" ");

   for (int i = 0; i < palavraChaveString.length; i++) {

      if (stopWordsDao.test(palavraChaveString[i].toUpperCase()) == true) {

         palavraChaveString[i] = "";
```

```
            }

        }

        respostas = perguntaDao.read(palavraChaveString);

        if (respostas == null) {

            int horaAtendimento = 0;

            lastId = perguntaPendenteDao.lastId();

            perguntaPendente.setId_pergunta_pendente(lastId);
            perguntaPendente.setPergunta_inteira(perguntar.getTxPergunta());
            perguntaPendente.setResposta(" ");
            perguntaPendente.setData_atual(data.format(dataSistema));
            perguntaPendente.setHora_atual(hora.format(dataSistema));
            perguntaPendente.setHistorico(historico);

            perguntaPendenteDao.create(perguntaPendente);

            horaAtendimento = Integer.parseInt(perguntaPendente.getHora_atual().substring(0,
2));

            if (horaAtendimento < 18) {

                pergunta.setResposta("Um momento, estou buscando com os atendentes a
resposta para a sua pergunta.");

                historico = historico + "\nUAMBot: " + pergunta.getResposta();

                pergunta.setResposta(perguntaPendenteDao.readResposta(lastId));

                historico = historico + "\nUAMBot: " + pergunta.getResposta();
```

```
                historico = historico + "\nUAMBot: " + "Você está satisfeito com essa
resposta?";

                session.setAttribute("questionamento", "satisfacao");

            } else {

                pergunta.setResposta("Não achei a resposta para essa pergunta e no momento não
há ninguém aqui que possa te ajudar. Peço que entre em contato em horário comercial. Deseja
fazer outra pergunta?");

                historico = historico + "\nUAMBot: " + pergunta.getResposta();

                session.setAttribute("questionamento", "outra");

            }

        } else {

            session.setAttribute("respostas", respostas);
            pergunta = (Pergunta) respostas.get(0);

            historico = historico + "\nUAMBot: " + pergunta.getResposta();

            historico = historico + "\nUAMBot: " + "Você está satisfeito com essa resposta?";

            session.setAttribute("questionamento", "satisfacao");

        }

        session.setAttribute("historico", historico);

    } else {
```

```
        if (questionamento.equalsIgnoreCase("satisfacao")) {

            if (perguntar.getTxPergunta().contains("sim")) {

                historico = historico + "\nUAMBot: " + "Deseja fazer outra pergunta?";
                session.setAttribute("questionamento", "outra");

                session.setAttribute("historico", historico);

                session.removeAttribute("respostas");

            } else {

                respostas = (ArrayList) session.getAttribute("respostas");

                pergunta = (Pergunta) respostas.get(1);

                historico = historico + "\nUAMBot: " + pergunta.getResposta();

                historico = historico + "\nUAMBot: " + "Você está satisfeito com essa
resposta?";

                session.setAttribute("historico", historico);

                session.setAttribute("questionamento", "satisfacao");

            }

        }

        if (questionamento.equalsIgnoreCase("outra")) {

            if (perguntar.getTxPergunta().contains("sim")) {
```

```
                historico = historico + "\nUAMBot: " + "Digite sua pergunta.";
                session.removeAttribute("questionamento");

                session.setAttribute("historico", historico);


            } else {

                historico = historico + "\nUAMBot: " + "A Anhembi Morumbi agradece seu
contato.";
                session.removeAttribute("questionamento");

                session.setAttribute("historico", historico);

            }

        }
    }
    return mapping.findForward("resposta");

  }
}
```

Dao.Dao.java

```
/*
 * Dao.java
 *
 * Created on 15 de Abril de 2008, 21:37
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */
```

```
package Dao;

import java.util.ArrayList;

/**
 *
 * @author aluno506-02
 */
public interface  Dao {

    public boolean create(Object o);

    public boolean delete(Object o);

    public boolean update(Object o);

    public ArrayList read(String perguntaFeita[]);

}
```

Dao.DataSource.java

```
/*
 * DataSource.java
 *
 * Created on 15 de Abril de 2008, 21:39
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package Dao;
```

```
import java.sql.*;

/**
 *
 * @author aluno506-02
 */
public class DataSource {

    private Connection conn = null;

    private Driver driver = null;

    private String url = "";

    private String username = "";

    private String password = "";

    public DataSource() {

        try {

            this.driver = new com.mysql.jdbc.Driver();

        } catch (SQLException ex) {

            ex.printStackTrace();

        }

        this.url = "jdbc:mysql://localhost:3306/UAMBOT";

        this.username = "root";
```

```
    this.password = "";
}
public void connect() {
    try {
        DriverManager.registerDriver( this.driver );
    } catch( Exception e ) {
        System.out.println("Falha ao carregar o driver!!!");
        e.printStackTrace();
    }
    try {
        this.conn = DriverManager.getConnection( this.url, this.username, this.password );
    } catch( Exception e ) {
        System.out.println("Falha ao obter conexao!!!");
        e.printStackTrace();
    }
}
public void disconnect() {
```

```
        if ( conn != null ) {
            try {
                this.conn.close();
                this.conn = null;
                DriverManager.deregisterDriver(driver);
            } catch( Exception e ) {
                System.out.println("Falha ao deconectar!!!");
                e.printStackTrace();
            }
        }
    }
    public Connection getConnection() {
        return ( this.conn );
    }
}
```

Dao.PerguntaDao.java

```
/*
 * To change this template, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
package Dao;

import Model.Pergunta;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

/**
 *
 * @author Rodrigo
 */
public class PerguntaDao implements Dao {

    private Connection conn;
    private static final String SQL_READ = "SELECT * FROM ub_pergunta WHERE
pergunta like ? and pergunta like ? and pergunta like ? and pergunta like ? and pergunta like ?
and pergunta like ? and pergunta like ? and pergunta like ? and pergunta like ? and pergunta
like ? and pergunta like ? and pergunta like ? and pergunta like ? and pergunta like ? and
pergunta like ? and pergunta like ? and pergunta like ? and pergunta like ? and pergunta like ?
and pergunta like ? order by utilizacao desc";

    public PerguntaDao() {
        DataSource ds;
        ds = new DataSource();
        ds.connect();
        this.conn = ds.getConnection();
    }

    public boolean create(Object o) {
        throw new UnsupportedOperationException("Not supported yet.");
```

```
}

public boolean delete(Object o) {
   throw new UnsupportedOperationException("Not supported yet.");
}

public boolean update(Object o) {
   throw new UnsupportedOperationException("Not supported yet.");
}

public ArrayList read(String perguntaFeita[]) {

   ArrayList respostas = new ArrayList();

   try {

      PreparedStatement stm = conn.prepareStatement(PerguntaDao.SQL_READ);

      for (int i = 0; i < perguntaFeita.length; i++) {

         stm.setString(i+1, "%" + perguntaFeita[i] + "%");

      }

      for (int i = perguntaFeita.length; i <= 20; i++) {

         stm.setString(i, "%");

      }

      ResultSet rs = stm.executeQuery();

      while (rs.next()) {
```

```
            Pergunta pergunta = new Pergunta();

            pergunta.setId(rs.getInt(1));
            pergunta.setPergunta(rs.getString(2));
            pergunta.setResposta(rs.getString(3));
            pergunta.setUtilizacao(rs.getInt(4));

            respostas.add(pergunta);

         }
      } catch (SQLException e) {
         e.printStackTrace(System.err);
         return null;
      }

      if (respostas.toArray().length == 0) {

         return null;

      }

      return respostas;

   }
}
```

Dao.PerguntaPendenteDao.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Dao;
```

```
import Model.PerguntaPendente;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

/**
 *
 * @author Rodrigo
 */
public class PerguntaPendenteDao implements Dao {

    private Connection conn;
    private static final String SQL_CREATE = "INSERT INTO
ub_pergunta_pendente(id_pergunta_pendente, pergunta_inteira, resposta, data_atual,
hora_atual, historico, data_atendimento, hora_atendimento, id_atendente)
values(?,?,?,?,?,?,?,?,?)";
    private static final String SQL_LAST_ID = "SELECT MAX(id_pergunta_pendente)
FROM ub_pergunta_pendente";
    private static final String SQL_READ_RESPOSTA = "SELECT resposta FROM
ub_pergunta_pendente WHERE id_pergunta_pendente = ?";

    public PerguntaPendenteDao() {
        DataSource ds;
        ds = new DataSource();
        ds.connect();
        this.conn = ds.getConnection();
    }

    public boolean create(Object o) {

        PerguntaPendente perguntaPendente = (PerguntaPendente) o;
```

```
        try {
            PreparedStatement stm =
conn.prepareStatement(PerguntaPendenteDao.SQL_CREATE);

            stm.setInt(1, perguntaPendente.getId_pergunta_pendente());
            stm.setString(2, perguntaPendente.getPergunta_inteira());
            stm.setString(3, perguntaPendente.getResposta());
            stm.setString(4, perguntaPendente.getData_atual());
            stm.setString(5, perguntaPendente.getHora_atual());
            stm.setString(6, perguntaPendente.getHistorico());
            stm.setString(7, perguntaPendente.getData_atendimento());
            stm.setString(8, perguntaPendente.getHora_atendimento());
            stm.setString(9, perguntaPendente.getId_atendente());

            stm.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace(System.err);
            return false;
        }

        return true;

    }

    public boolean delete(Object o) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    public boolean update(Object o) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    public ArrayList read(String[] perguntaFeita) {
```

```
        throw new UnsupportedOperationException("Not supported yet.");
    }

    public int lastId() {

        int lastId = 0;

        try {

            PreparedStatement stm =
conn.prepareStatement(PerguntaPendenteDao.SQL_LAST_ID);

            ResultSet rs = stm.executeQuery();

            if (rs.next()) {

                lastId = rs.getInt(1);

                lastId++;

                return lastId;

            }
        } catch (SQLException e) {
            e.printStackTrace(System.err);
            return 0;
        }

        return 0;
    }

    public String readResposta(int idPerguntaPendente) {

        String resposta = null;
```

```
        try {

            PreparedStatement stm =
conn.prepareStatement(PerguntaPendenteDao.SQL_READ_RESPOSTA);

            stm.setInt(1, idPerguntaPendente);

            ResultSet rs = stm.executeQuery();

            if (rs.next()) {

                resposta = rs.getString(1);

            }
        } catch (SQLException e) {
            e.printStackTrace(System.err);
            return null;
        }

        if (resposta.equalsIgnoreCase(" ")) {

            resposta = null;

        }

        return resposta;

    }
}
```

Dao.StopWordsDao.java

```
/*
```

```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

/**
 *
 * @author Rodrigo
 */
public class StopWordsDao implements Dao {

    private Connection conn;
    private static final String SQL_TEST = "SELECT * FROM ub_stop_word WHERE word
= ?";

    public StopWordsDao() {
        DataSource ds;
        ds = new DataSource();
        ds.connect();
        this.conn = ds.getConnection();
    }

    public boolean create(Object o) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    public boolean delete(Object o) {
        throw new UnsupportedOperationException("Not supported yet.");
```

```
}

public boolean update(Object o) {
   throw new UnsupportedOperationException("Not supported yet.");
}

public ArrayList read(String[] perguntaFeita) {
   throw new UnsupportedOperationException("Not supported yet.");
}

public boolean test(String word) {

   try {

      PreparedStatement stm = conn.prepareStatement(StopWordsDao.SQL_TEST);

      stm.setString(1, word);

      ResultSet rs = stm.executeQuery();

            if (rs.next()) {

               return true;

      }

   } catch (SQLException e) {
      e.printStackTrace(System.err);
      return false;
   }

   return false;

}
```

```
}
```

Form.PerguntarForm.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package Form;

import org.apache.struts.action.ActionForm;

/**
 *
 * @author Rodrigo
 */
public class PerguntarForm extends ActionForm {

    private String txPergunta;

    /**
     * @return the txPergunta
     */
    public String getTxPergunta() {
        return txPergunta;
    }

    /**
     * @param txPergunta the txPergunta to set
     */
    public void setTxPergunta(String txPergunta) {
        this.txPergunta = txPergunta;
    }
```

```
}
```

Model.Pergunta.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Model;

/**
 *
 * @author Rodrigo
 */
public class Pergunta {

    private int id;
    private String pergunta;
    private String resposta;
    private int utilizacao;

    /**
     * @return the id
     */
    public int getId() {
        return id;
    }

    /**
     * @param id the id to set
     */
    public void setId(int id) {
```

```
    this.id = id;
}

/**
 * @return the pergunta
 */
public String getPergunta() {
    return pergunta;
}

/**
 * @param pergunta the pergunta to set
 */
public void setPergunta(String pergunta) {
    this.pergunta = pergunta;
}

/**
 * @return the resposta
 */
public String getResposta() {
    return resposta;
}

/**
 * @param resposta the resposta to set
 */
public void setResposta(String resposta) {
    this.resposta = resposta;
}

/**
 * @return the utilizacao
 */
```

```
    public int getUtilizacao() {
        return utilizacao;
    }

    /**
     * @param utilizacao the utilizacao to set
     */
    public void setUtilizacao(int utilizacao) {
        this.utilizacao = utilizacao;
    }
}
```

Model.PerguntaPendente.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Model;

import java.util.Date;

/**
 *
 * @author Rodrigo
 */
public class PerguntaPendente {

    private int id_pergunta_pendente;
    private String pergunta_inteira;
    private String resposta;
    private String data_atual;
    private String hora_atual;
    private String historico;
```

```
private String data_atendimento;
private String hora_atendimento;
private String id_atendente;

/**
 * @return the id_pergunta_pendente
 */
public int getId_pergunta_pendente() {
    return id_pergunta_pendente;
}

/**
 * @param id_pergunta_pendente the id_pergunta_pendente to set
 */
public void setId_pergunta_pendente(int id_pergunta_pendente) {
    this.id_pergunta_pendente = id_pergunta_pendente;
}

/**
 * @return the pergunta_inteira
 */
public String getPergunta_inteira() {
    return pergunta_inteira;
}

/**
 * @param pergunta_inteira the pergunta_inteira to set
 */
public void setPergunta_inteira(String pergunta_inteira) {
    this.pergunta_inteira = pergunta_inteira;
}

/**
 * @return the resposta
```

```
 */
public String getResposta() {
    return resposta;
}

/**
 * @param resposta the resposta to set
 */
public void setResposta(String resposta) {
    this.resposta = resposta;
}

/**
 * @return the data_atual
 */
public String getData_atual() {
    return data_atual;
}

/**
 * @param data_atual the data_atual to set
 */
public void setData_atual(String data_atual) {
    this.data_atual = data_atual;
}

/**
 * @return the hora_atual
 */
public String getHora_atual() {
    return hora_atual;
}

/**
```

```
 * @param hora_atual the hora_atual to set
 */
public void setHora_atual(String hora_atual) {
    this.hora_atual = hora_atual;
}

/**
 * @return the historico
 */
public String getHistorico() {
    return historico;
}

/**
 * @param historico the historico to set
 */
public void setHistorico(String historico) {
    this.historico = historico;
}

/**
 * @return the data_atendimento
 */
public String getData_atendimento() {
    return data_atendimento;
}

/**
 * @param data_atendimento the data_atendimento to set
 */
public void setData_atendimento(String data_atendimento) {
    this.data_atendimento = data_atendimento;
}
```

```
    /**
     * @return the hora_atendimento
     */
    public String getHora_atendimento() {
        return hora_atendimento;
    }

    /**
     * @param hora_atendimento the hora_atendimento to set
     */
    public void setHora_atendimento(String hora_atendimento) {
        this.hora_atendimento = hora_atendimento;
    }

    /**
     * @return the id_atendente
     */
    public String getId_atendente() {
        return id_atendente;
    }

    /**
     * @param id_atendente the id_atendente to set
     */
    public void setId_atendente(String id_atendente) {
        this.id_atendente = id_atendente;
    }
}
```

ANEXO A - PROTOCOLOS / REQUERIMENTOS / SOLICITAÇÕES

Como e onde fazer solicitações à Universidade?

Solicitações referentes à vida acadêmica, em alguns casos, devem ser feitas presencialmente na Central de Atendimento ao Aluno; em outros, é possível utilizar a Central de Serviços On-Line na página de Serviços para Alunos, no portal. O aluno deverá preencher o requerimento, pagar a taxa (quando houver) e retornar dentro do prazo estipulado.

QUAIS OS PRAZOS DE EXPEDIÇÃO DOS DOCUMENTOS?

Cada requerimento tem seu prazo para expedição. Veja tabela.

DOCUMENTOS	**PRAZO DE EXPEDIÇÃO**
Atestado de Matrícula	3 dias úteis
Atestados com Especificação	5 dias úteis
Histórico Escolar	15 dias úteis
Certificado de Conclusão do Curso	15 dias úteis
Programas por Período Letivo	15 dias úteis
Diploma	6 meses
Pontuação do Vestibular	1 dia útil
Critério de Avaliação	1 dia útil

Como fico sabendo se meu requerimento ficou pronto?

No portal da Universidade na internet, você deve acessar a página de Serviços para Alunos e em seguida o link Andamento de Requerimentos. A Universidade não entra em contato com o aluno para avisar que a solicitação está pronta.

SENHA WEB

Como faço para obter senha da web?

A sua senha e login estão impressos na via de matrícula (rodapé esquerdo da folha), que é entregue no ato de sua matrícula.

E quem perdeu a via de matrícula ou esqueceu a senha, como deve proceder? Esses alunos devem acessar o portal da Universidade na Internet, página de Serviços para Alunos, link Senha de acesso à Unidade Web. O aluno deve digitar seu número de matrícula e

sua senha será enviada automaticamente para o e-mail que está cadastrado em seus dados pessoais. Caso não saiba seu número de matrícula, compareça a uma das Centrais de Atendimento ao Aluno.

ATUALIZAÇÃO CADASTRAL

Como faço para informar sobre mudanças nos dados de meu cadastro?
Sempre que houver mudança em seus dados pessoais, você deve preencher a solicitação de alteração na Unidade Web. Acesse o portal da Universidade e, na página de Serviços para Alunos, clique em Alteração Cadastral.

CRITÉRIO DE APROVAÇÃO

Quais são os critérios de aprovação para os alunos?
Para obter aprovação, o aluno precisa obter média igual ou superior a seis (6,0) e ter 75% de freqüência no semestre.

Qual é a quantidade máxima de faltas que eu posso ter em um semestre?
Depende da carga horária da disciplina. Veja tabela abaixo:

Informações sobre Carga Horária das Disciplinas de Graduação

Carga Horária da Disciplina (h/a)	Quantidade de Horas Semanais (h)	Máximo de Faltas no Semestre
17	1	5
34	2	10
40	2	10
51	3	13
68	4	18
80	4	20

85	5	22
102	6	26
136	8	34
204	12	51

NOTAS E FALTAS

Como consulto minhas notas e faltas?
Para verificar notas e faltas, basta acessar o portal e, na página de Serviços para Alunos, clicar em Notas, Faltas e Revisões usando sua senha pessoal. Caso não concorde com as informações lançadas, você pode solicitar um requerimento de revisão de notas ou de faltas pelo próprio site.

RENOVAÇÃO DE MATRÍCULA

Como eu faço para efetuar a minha renovação de matrícula?
É necessário estar sem pendências financeiras e nem de documentos. Você deve efetuar o pagamento do boleto de janeiro ou de julho até o prazo estipulado, para que sua renovação de matrícula seja automática. No final de cada semestre, a Universidade envia cartas aos alunos com esses procedimentos.

E se eu estiver em débito, o que faço?
Se estiver em débito com a Instituição, seja de documentos ou pagamentos, você deve regularizar a sua situação até a data da renovação de matrículas, apresentando o documento faltante ou o boleto de janeiro ou de julho devidamente pago na data estipulada.
E se eu não renovar a matrícula, o que acontece?
A não renovação da matrícula nas datas estipuladas implica em abandono de curso e desvinculação do aluno da Universidade.

DEPENDÊNCIA / ADAPTAÇÃO

O que é Dependência?

Dependência é a disciplina cursada pelo aluno que não atingiu média igual ou superior a seis (6,0) e/ou freqüência mínima de 75% para aprovação.

O que é Adaptação?

Adaptação é a disciplina que o aluno necessita cursar para se adaptar ao currículo, seja por transferência interna, externa, reingresso ou reabertura de matrícula.

Como faço para cumprir minha Dependência/Adaptação?

Para cumprir a Dependência/Adaptação, você deve efetuar matrícula pelo site na referida disciplina. Estando sujeito ao regime de Dependência/Adaptação, o aluno pagará, além da mensalidade, uma taxa suplementar por disciplina (de acordo com a carga horária), salvo se esta estiver inclusa nas lacunas de disciplinas dispensadas. A Dependência/Adaptação será cursada no horário oferecido pela Universidade.

Quais são os tipos de Dependência/Adaptação que posso fazer?

Normal: cursada em horários estabelecidos pelos cursos;

Especial: cursada em horário alternativo, com número mínimo de quinze alunos. Se não atingir tal número, o grupo formado deverá pagar o valor correspondente a quinze alunos;

Intensiva: cursada em janeiro ou julho, formada de acordo com a manifestação de interesse dos alunos (número mínimo de quinze alunos);

Assistida: disciplina que já não faz mais parte da grade curricular ou não é oferecida em horários regulares de qualquer outro curso. É orientada por um professor com provas e trabalhos;

Orientada: cursada ON-LINE.

Qual o valor pago pela DP?

Isso varia de acordo com a carga horária da disciplina. Verifique no Edital de Taxas e Emolumentos o valor da carga horária da disciplina.

Como é feito o pagamento da DP?

Os boletos serão enviados para a residência do aluno. O valor cobrado será dividido em cinco parcelas com vencimento no dia 20 de cada mês. Caso não receba o boleto até três dias antes do vencimento, retire-o presencialmente em uma das Centrais de Atendimento ou no Setor de Negociação.

CANCELAMENTO E TRANCAMENTO DE MATRÍCULA

O que é trancamento de matrícula?

O trancamento da matrícula corresponde à suspensão total das atividades acadêmicas por um determinado prazo. Neste intervalo de tempo, o aluno mantém vínculo com a Universidade e pode requerer seu retorno aos estudos sem prestar novo processo seletivo. Esse retorno é condicionado à oferta de turmas e existência de vagas no respectivo curso / campus / horário pretendidos, mantendo assim seu vínculo com a Universidade.

Após expiração do prazo de trancamento, o aluno deverá prestar novo Processo Seletivo, adequando-se assim à nova grade curricular.

O que é cancelamento de matrícula?

O cancelamento de matrícula corresponde à formalização do desligamento do aluno com a Universidade, estando assim sem nenhum vínculo com a mesma. Caso o aluno decida voltar a estudar na Universidade, deverá prestar novo Processo Seletivo.

Por quanto tempo a matrícula pode ficar trancada?

O período máximo de trancamento de matrícula é de 2 (dois) anos (4 semestres) para os cursos de graduação e de 1 (um) ano (2 semestres) para os cursos de curta duração.

Como devo proceder para pedir o trancamento/cancelamento da matrícula?

O pedido deve ser requerido presencialmente em uma das Centrais de Atendimento ao Aluno, antes do término do semestre (em período estipulado pelo Calendário Acadêmico).

REABERTURA DE MATRÍCULA

Como devo solicitar minha reabertura de matrícula?

Se você estiver regularmente com a matrícula trancada, deve solicitar a reabertura pelo portal da Universidade, na página de Serviços para Aluno, no link Análise Curricular – Reabertura de Matrícula/Retornar ao curso semestral.

Quando posso pedir reabertura de matrícula?

Ver calendário acadêmico no portal da Universidade.

TRANSFERÊNCIAS DE TURNO E CAMPUS

Quero mudar de campus/ou de turno, como faço?
Solicite a transferência pelo site da Universidade, link Serviços para Alunos – Central de Atendimento On-Line – Transferência de turno/campus. O deferimento do pedido está condicionado à oferta de vagas, devendo assim o aluno permanecer em sua turma atual até a transferência.

DE CURSOS / HABILITAÇÃO / FORMAÇÃO ESPECÍFICA

Quero me transferir de Curso/habilitação ou formação específica. O que faço?
Solicite a transferência pelo site da Universidade, link Serviços para Alunos – Análise Curricular - Transferir de Curso/Habilitação/Formação Específica. Vale lembrar que existe um prazo específico para tal solicitação (ver calendário acadêmico).

DA ANHEMBI MORUMBI PARA OUTRA INSTITUIÇÃO

Quem pode pedir transferência da Anhembi Morumbi?
Todo aluno que estiver regularmente matriculado na Universidade no período em que solicitar transferência. É considerado vinculado o aluno devidamente matriculado no semestre vigente ou com matrícula trancada.

Quando posso pedir minha transferência?
A Universidade aceita pedidos de transferência em qualquer época do ano.

Como devo proceder para pedir transferência da Anhembi Morumbi?
Solicite a transferência em formulário próprio, presencialmente, na Central de Atendimento ao Aluno, anexando o atestado de vaga da outra instituição. Para essa solicitação, você deverá pagar uma taxa (ver edital de taxas e emolumentos).

DE OUTRA INSTITUIÇÃO PARA A ANHEMBI MORUMBI

Como solicito minha transferência?

Você faz a solicitação pelo site da Universidade – link Serviços para Alunos – Análise Curricular – Transferir para a Anhembi Morumbi - e depois comparece a uma das Centrais de Atendimento para a entrega dos documentos.

Quando posso pedir transferência para a Anhembi Morumbi?
Verifique o período de transferência no calendário acadêmico.

Que documentos são necessários para a transferência?
O candidato deve apresentar o Conteúdo Programático das disciplinas cursadas e o Histórico Escolar. Os documentos devem ser originais ou cópias autenticadas.

Quanto tempo devo esperar a confirmação da minha transferência?
Todo o processo é realizado via web. Os documentos entregues serão submetidos à análise da coordenação do curso. A análise é enviada via e-mail ao aluno. Estando de acordo, ele dará início ao processo de transferência. A Universidade Anhembi Morumbi emitirá um atestado de vaga para a instituição de origem do candidato. Posteriormente, o candidato deverá solicitar formalmente em sua universidade de origem a transferência para a Anhembi Morumbi. Quando a escola de origem enviar a documentação de transferência do candidato, este será comunicado pela nossa secretaria e deverá imediatamente fazer sua matrícula.

DISPENSA DISCPLINA

Em que caso eu posso pedir dispensa de disciplina?
Você pode pedir dispensa da disciplina quando tiver em mãos uma análise curricular feita pelo coordenador, com a indicação desta dispensa; ou quando já cursou a disciplina na própria Anhembi Morumbi ou em outra instituição e quer pedir a dispensa de uma disciplina que está para ser cursada. Nestes casos, você deve fazer a solicitação por meio do site da Universidade.

Quando posso pedir dispensa de disciplina?
Ver calendário acadêmico.

No caso de disciplinas cursadas em outras escolas, como devo proceder?

Você deve solicitar a dispensa da disciplina pelo site e entregar imediatamente, em uma das Centrais de Atendimento ao Aluno, o Histórico Escolar e Conteúdo Programático da outra universidade (originais ou cópias autenticadas).
No caso de disciplinas cursadas na Anhembi Morumbi:
Requerimento indicando a disciplina, o ano e a série em que foi cumprida.

COMPENSAÇÃO DE FALTAS

Em que caso minhas faltas podem ser compensadas?
Quando o afastamento da instituição ocorrer por motivo de licença médica e o aluno estiver enquadrado nos casos permitidos pela Lei 1.044/69: doenças infecto-contagiosas, gravidez (licença gestante) e acidentes com traumatismos, quando o afastamento for superior a 5 dias.

O que devo fazer para pedir compensação de faltas?
Você deve apresentar, no prazo máximo de 10 dias após constatação do afastamento, o Laudo Médico, com especificação da doença ou CID (Código Internacional da Doença) e do período do afastamento (mínimo de 5 dias e máximo de 3 meses). A Central de Atendimento ao Professor se encarregará de providenciar os temas dos exercícios domiciliares e de avisar o aluno quando os mesmos estiverem disponíveis.

Em que caso minhas faltas podem ser abonadas?
Somente em dois casos: Prestação de Serviço Militar - o oficial ou aspirante a oficial da reserva convocado para o serviço ativo; e participação em congressos científicos e competições desportivas ou artísticas, de âmbito nacional ou internacional, com autorização expressa do Ministério da Educação.

O que eu devo fazer para pedir abono de faltas?
Você deve apresentar, no prazo máximo de 5 dias após a constatação do afastamento, documento que comprove o motivo no setor do Protocolo.

Quando as faltas serão retiradas?
As faltas serão retiradas no final do bimestre.

ANÁLISES CURRICULARES

Quem deve pedir Análise Curricular?
Quem se desligou da Anhembi Morumbi e quer saber como está sua situação atual;
Quem deseja pedir dispensa disciplina;
Quem tem dúvidas quanto às disciplinas pendentes;
Quem deseja parar o curricular e cursar somente dps ou vice-versa;
Quem trancou a matrícula e deseja voltar a estudar (solicitar reabertura).

Como solicito uma Análise Curricular?
Pelo site da Universidade – Link Serviços para Aluno – Análise Curricular.

REVISÃO DE NOTAS E FALTAS

Quando eu posso requerer revisão?
Você pode requerer revisão sempre que discordar das notas/faltas lançadas no sistema, ou ainda, quando a nota estiver ausente.

Como eu faço essa solicitação?
As notas e faltas serão revistas mediante solicitação via Web, em prazo estipulado pelo Calendário Acadêmico, de acordo com anotações das faltas no diário de classe.

Em quanto tempo tenho a resposta do requerimento?
Em 20 dias úteis, caso não coincida com período de férias ou recesso dos professores.

PROVAS DE SEGUNDA CHAMADA

O que é a prova de segunda chamada?
É uma prova que repõe a avaliação da N2 que não foi realizada.

Posso realizar duas provas de segunda chamada da mesma disciplina?
Não. É permitida apenas a realização de uma prova de segunda chamada, de uma determinada disciplina, em um mesmo semestre.

O que devo fazer para requerer prova de segunda chamada?
Na semana estipulada pelo calendário acadêmico, compareça à sala de aula no mesmo dia e horário da disciplina que irá fazer a prova. Assine a lista de presença e realize a prova com o professor. Estas provas são pagas (ver edital de taxas e emolumentos) e os boletos serão enviados para a sua residência. Caso isso não ocorra, solicite segunda via do boleto nas Centrais de Atendimento ao Aluno.

COLAÇÃO DE GRAU

O que é colação de grau?
É o último ato acadêmico do aluno. É oficial e obrigatória, sendo realizada em sessão solene e pública para os cursos de graduação, em dia e horário previamente fixados pela Universidade.

O que eu preciso fazer para participar da colação?
Você precisa ter concluído o curso de graduação, sem nenhuma pendência acadêmica, como: disciplinas, Trabalho de Conclusão de Curso ou estágio.

E se eu tiver alguma dependência, estágio ou disciplina por concluir?
Após a conclusão das pendências, você deve requerer a colação de grau especial pelo site da Universidade – Serviços para Alunos - Central de Atendimento On-Line. Aguarde a resposta da Secretaria de Registros Acadêmicos, que entrará em contato para passar informações a respeito da data, hora e local da colação especial ou sobre suas pendências.

Os cursos seqüenciais possuem colação de grau?
Não. O que existe é somente uma cerimônia formal como marco da conclusão do curso.

BOLSAS DE ESTUDO

Como faço para conseguir uma bolsa de estudo?
Você obtém toda e qualquer informação sobre bolsas de estudos no Setor de Bolsas (R. Libero Badaró, 487 – Campus Vale do Anhangabaú) ou pelo portal da Universidade - link Serviços para Aluno – Central de Atendimento on-line – Bolsas de Estudos.

BOLETOS DAS MENSALIDADES

Quando é o vencimento do boleto da mensalidade?
O boleto da mensalidade vence no 15º dia útil de cada mês.

Posso alterar a data de vencimento dos boletos?
Não. A Universidade não altera a data de vencimento.

Eu recebo o boleto em casa?
Sim. Os boletos são enviados para o endereço que está cadastrado no sistema. Caso você não o receba em até 3 dias antes do vencimento, retire a segunda via do boleto no site da Universidade – link Serviços para Alunos – Central de Atendimento On-Line – Boletos, ou presencialmente em uma das Centrais de Atendimento.

E os boletos vencidos, como faço para retirar?
Nos casos de boletos vencidos, retire presencialmente em umas das Centrais de Atendimento, no Setor de Negociação (Campus Vale do Anhangabaú), ou ainda solicitar pelo Contact Center (0800 015 9020). Existe a possibilidade de alteração da data de vencimento para que o aluno consiga retirar a segunda via pelo site, mas à mensalidade serão acrescidos multa e juros.

CARTEIRINHAS DA UNIVERSIDADE

Como obtenho minha carteirinha da Universidade?
Os alunos calouros (ingressantes no semestre vigente) não precisam solicitar a carteirinha da Universidade. Elas são entregues todo início de semestre pelo Banco Real. O local, data e horário de entrega são fixados pelo banco e divulgados em cartazes espalhados pelos campi.

Sou aluno veterano e ainda não tenho minha carteirinha, o que devo fazer?
Você deve comparecer a uma das Centrais de Atendimento ao Aluno para solicitar a carteirinha. Será verificado, junto à empresa que as emite, se realmente nunca foi impresso carteirinha com aquele RA de solicitação. Caso nunca tenha sido impresso, será emitida uma via. Se já houver registro de impressão, a solicitação será indeferida e o aluno deverá solicitar segunda via efetuando o pagamento da taxa.

Como solicito a segunda via da carteirinha da Universidade?

Você deve comparecer a uma das Centrais de Atendimento ao Aluno e solicitar a segunda via da carteirinha, efetuando o pagamento da taxa (ver edital de taxas e emolumentos). A nova carteirinha demora cerca de 40 dias para ficar pronta. A segunda via da carteirinha deverá ser solicitada pelo site da Universidade – Link Serviços para Alunos – Central de Atendimento On-line, com pagamento de taxa.

INTEGRALIZAÇÃO DOS CURSOS

Existe um limite de tempo para terminar o curso?

Todo curso tem um período mínimo e máximo (estipulado no currículo) para que o aluno possa concluí-lo. O não cumprimento desse prazo acarreta na integralização do curso, e o aluno é obrigado a prestar novamente o Processo Seletivo para dar continuidade ao mesmo. Veja tabela:

Cursos	Prazo Mínimo(Anos)	Prazo Máximo(Anos)
Administração	04	07
Arquitetura e Urbanismo	05	09
Ciência Da Computação	04(Bacharelado) 05(Licenciatura)	07 (Bacharelado) 07 (Licenciatura)
Ciências Econômicas	05	09
Comunicação Social	04	07
Dança e Movimento	03 03 e meio (a partir de 2001)	05
Design Digital/Embalagens	04	07
Design De Moda	04	07
Direito	05	09
Enfermagem	04	06
Engenharia Civil	06(até 1996) 05(a partir de 1997)	09 (até 1996) 09 (a partir de 1997)
Engenharia De Produção	05	09

Engenharia De Agrimensura e Geomática	05	09
Engenharia De Telecomunicações	05	09
Farmácia	04	07
Fisioterapia	04	07
Hotelaria	04	07
Lazer, Recreação E Eventos	04	07
Letras	03 e meio (a partir de 2001) Bacharelado e Licenciatura	07(até 1996) 07(Bacharelado e Licenciatura, a partir de 1997)
Marketing	04	07
Matemática	03 e meio (a partir de 2001)Bacharelado e Licenciatura	07 (até 1996) 07 (Bacharelado e Licenciatura, a partir de 1997)
Medicina Veterinária	05	09
Naturologia	04	07
Negócios Da Moda	04	07
Nutrição	04	06
Pedagogia	04(até 1996) 03(a partir de 1997)	0 (até 1996 e Licenciatura Em T.E.) 07 (a partir de 1997)
Química	03 e meio (a partir de 2000) Bacharelado e Licenciatura	07 (até 1996) 07 (Bacharelado e Licenciatura, a partir de 1997)
Quiropraxia	04 e meio	07

Secretariado Executivo Bilíngüe	03	05
Superior de Aviação Civil	03	05
Teatro	03 e meio (a partir de 2001)	05
Turismo	04	07
Cursos Seqüenciais	02	03

ANEXO B - LISTA DE STOP WORDS

a
à
agora
ainda
além
alguns
ano
anos
antes
ao
aos
apenas
após
aqui
área
as
às
assim
até
bem
bom
cada
caso
cerca
com
como
da
dar
das
de
depois
desde
do
dos
durante
e
é
ela
ele
eles
em
enquanto
então
entre
era
essa
esse
esta
está
este
eu
fato
faz
fazer
fez
ficou
foi
foram
grande
há
isso
já
mas
me
mesma
mesmo
muito
na
nada
nas
nem
neste
no
nos
nós
num
numa
o
onde
os
ou
para
pela
pelo
pelos
por
pouco
que
são
se
seja
sem
semana
ser
seu
seus
só
sobre
sua
suas
também
tão
tem
ter
teve
tinha
um
uma
vai
vão
vem
vez
vezes