UNIVERSIDADE DE SÃO PAULO

Departamento de Engenharia Elétrica

BANCO DE DADOS PARA PERSISTÊNCIA DE ARQUIVOS DE VOZ

Escola de Engenharia de São Carlos – SP

LEONARDO BLATTNER PUPO

BANCO DE VOZES

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo.

Curso de Engenharia de Computação com ênfase em Telecomunicações

ORIENTADOR: Prof. Dr. José Carlos Pereira

São Carlos

2007

ÍNDICE

Lista de Figuras

Agradecimentos

Gostaria de agradecer a todas as pessoas que tornaram, direta ou indiretamente esse projeto possível.
Agradeço a:

Prof. Dr. José Carlos Pereira
Prof. Dr. Carlos Dias Maciel

pela contribuição com amostras de vozes e ambiente de trabalho necessário, além de toda a orientação durante a realização do mesmo.
Agradeço a Reginaldo Delfino, Adriano Carlos Verona, Valter Rogério Messias e Breno Henrique Leitão pela contribuição em quesitos técnicos envolvendo a linguagem Java e banco de dados PostgreSQL.

A colaboração das pessoas acima citadas foi de fundamental importância para a realização deste projeto.

Resumo

O seguinte projeto foi realizado baseado em necessidades do Departamento de Engenharia Elétrica de organização de arquivos contendo amostras de voz, as quais são analisadas e estudadas por profissionais da área da saúde para a verificação de patologias ou qualquer outro tipo de anomalia do sistema vocal.

Tendo em vista que diversos profissionais trabalham nas áreas que envolvem a computação aplicada à saúde, é necessário certo grau de organização das amostras existentes, de modo a facilitar sua manipulação.

É de extrema importância para um profissional ligado à saúde, como um fonoaudiólogo, por exemplo, poder ter acesso de maneira simples e eficiente a diversos tipos de amostras de voz para que possa fazer comparações, classificações e diagnósticos através da análise das mesmas.

Esse objetivo pode ser obtido através da organização dos arquivos contendo as amostras em bases de dados, permitindo ao usuário um acesso simples, rápido e eficiente.

Inicialmente, tínhamos uma grande quantidade de amostras, no entanto, estas se encontravam armazanadas de forma desorganizada e sem critérios de padronização. O banco de dados construido foi útil para armazená-las de forma organizada e que evitasse redundâncias. Dessa maneira, podemos fornecer a um profissional especializado melhores condições para que possa analisar, comparar e diagnosticar as amostras, sem perder tempo devido às complicações encontradas em analisar dados desordenados.

Todo o sistema foi construído para ser utilizado com ferramentas livres, o que o torna acessível a qualquer interessado.

Critérios de segurança foram adotados para manter sob sigilo informações como o nome dos pacientes que possuem suas amostras de voz armazenadas na base de dados, segundo a especificação do projeto.

Além disso, outros critérios como a comparação de arquivos aparentemente idênticos, para evitar o armazenamento de informações redundantes, visto que o conjunto inicial de amostras apresentava alguns dados redundantes.

Todos esses cuidados foram tomados para que o espaço de armazenamento pudesse ser otimizado e as buscas pudessem ser facilitadas, sem, no entanto, comprometer a privacidade dos dados.

Abstract

The following project was carried out based on necessities of the Department of Electric Engineering of voice samples organization, having in mind that several professionals work in areas that use extensively the computation devoted to the health.

It is of extreme importance for a professional like a phonologyst for example, to be able to have simple and efficient access to several types of voice samples so that he can do comparisons, classifications and diagnoses them.

That is obtained through the organization of the samples in databases, which allow to any user a simple, quick and efficient access.

Initially, we had a great quantity of samples, however, they were disorganized and without criteria of standardization stored. The database built was useful to store them in an efficient and organized form. In that way, we can supply a specialized professional better conditions so that it can analyse, compare and diagnose the samples, without wasting time due to the complications found in analysing and searching the disordered data.

The whole system was built for using with free tools, which makes it accessible to anyone interested.

Criteria of security were adopted to maintain under secrecy informations like the name of the patients who have his voice samples stored on database, according to the project´s specification.

Besides, the comparison of files apparently identical, in order that avoid the storage of redundant information, because the actual set of sample was presenting some redundancy.

These security criteria were adopted so that the storage space could be optimized and the searches could be made easy, without, however, compromising the data privacy.

1. Introdução

1.1 Contextualização e Domínio de Aplicação

No mundo atual, a determinação clara de objetivos faz parte de um processo de gerenciamento das regras de conduta normativas. Evidentemente, nosso sistema foi construído baseado em expectativas de uma contínua expansão das atividades, sem prejudicar o seu desempenho e eficiência.

Pensando mais a longo prazo, a consulta a grande quantidade de dados de forma aleatória e desordenada desafia a capacidade de equalização do retorno esperado.

Acima de tudo, é fundamental ressaltar que o aumento do diálogo entre os diferentes usuários do sistema não pode se dissociar, para que possamos construir uma solução que seja a mais próxima do ideal possível. É necessário o entendimento prévio da aplicação e de também das circunstâncias em que será utilizada para que a solução seja adequada.

No mundo atual, o aumento do diálogo entre as diferentes partes envolvidas auxilia a preparação e a composição da gestão inovadora da qual fazemos parte. No entanto, não podemos esquecer que a competitividade nas diversas áreas comerciais deve passar por modificações independentemente do remanejamento dos quadros funcionais.

A automação de ambientes industriais, comerciais, casas e edifícios já é uma prática constante no mercado. No entanto, na área da saúde isso vem crescendo de maneira exponencial nos últimos anos. O ambiente hospitalar, por exemplo, ainda é pouco automatizado. Cerca de 10% a 15% dos hospitais estão automatizados nos dias de hoje.

A grande questão que o setor hospitalar precisa entender é que esse processo proporciona uma gestão mais eficiente de recursos – em termos de conforto e de informações – e reduz custos, proporcionando maiores ganhos à instituição.

Como, hoje em dia, tudo é feito manualmente, o procedimento se torna muito lento. Além de consumir tempo e dinheiro, podem ocorrer diversos erros prejudiciais à instituição. Com a implantação dessa tecnologia, há uma redução da ociosidade dos pacientes, uma liberação mais rápida, e a desburocratização do processo.

É comum encontrar em hospitais pacientes com uma pulseira de identificação com código de barras. Por meio desse código, o hospital tem armazenadas algumas informações referentes àquele paciente, tais como data e hora em que entrou na ala, medicamentos, médico responsável, prescrições etc. Com a automatização de todo o processo, podem ser agregadas informações de diversos tipos, e isto permite que não haja erros de decorrência humana em relatórios e escritas. Desse modo, minimiza-se o tempo no processo interno e a estadia do paciente no hospital, além de se obter um aumento na disponibilidade de leitos.

A automatização dos processos diários de um hospital proporciona ganhos como:

• Segurança do paciente, pois minimiza radicalmente o risco de os profissionais de saúde cometerem erros ao ministrar um medicamento.

• Segurança no trabalho de médicos e enfermeiros, por haver um maior controle dos prontuários dos pacientes.

• Rapidez de reembolso aos hospitais pelos planos de saúde.

• Fechamento de contas mais exato.

• Preciso controle de estoque e farmácia, minimizando os gastos com medicamentos vencidos e/ou desviados.

Sabemos que a computação está presente em praticamente todos os lugares em que vamos, no entanto, nos últimos anos, a saúde vem sendo fortemente influenciada pelas novas tecnologias, que cada vez mais se mostram eficientes e proporcionam melhores condições de tratamento para os pacientes.

Isso se reflete também nesse trabalho, que, embora não tenha ligação direta com hospitais, como citado acima, mostra a influência da computação também no armazenamento de informações médicas, pois do que adiantaria termos os melhores equipamentos para analisar informações sobre um paciente, se não pudéssemos armazená-las de forma eficiente e acessá-las com facilidade?

Uma importante utilização dos bancos de vozes está nas aplicações desenvolvidas especialmente para pessoas portadoras de deficiência visual, que necessitam de softwares especiais para que possam interagir com um computador. Atualmente, alguns softwares possibilitam que o texto de uma página web, por exemplo, seja interpretado em forma de voz, permitindo que um deficiente visual navegue pela internet, ouvindo o texto escrito e podendo com ele interagir.

Um amplo mercado vem se abrindo nesse sentido. Até pouco tempo atrás, poucas empresas se importavam com a necessidade de um deficiente visual de interagir com um computador, no entanto, hoje em dia muitas delas se preocupam com essa parcela do mercado, que, podendo ter acesso a determinadas informações contidas em páginas web, as quais não tinham acesso anteriormente devido à falta de ferramentas que possibilitassem tal interação, podem se tornar um mercado consumidor em potencial.

Além das aplicações relativas à área da saúde, atualmente temos bancos de vozes presentes também na Web. Nesses bancos é possível selecionar um cantor, locutor de áudio, ou até mesmo uma pessoa comum e ouvir uma amostra de sua voz.

Alguns sites permitem também que o usuário grave e envie uma amostra de sua voz para ser armazenada em sua base de dados.

Podemos perceber, portanto, que aplicações envolvendo arquivos de áudio armazenados em bases de dados vêm sendo amplamente utilizadas em diversas áreas, algumas de grande

importância, como na medicina, por exemplo, e outras nem tanto, como pessoas que gravam uma amostra de sua voz e enviam para algum site apenas por lazer.

Esse grande crescimento da interação de arquivos multimídia com as bases de dados vem ocorrendo graças ao crescimento da capacidade de armazenamento dos discos rígidos apresentado nos últimos anos. Armazenar uma grande quantidade de arquivos de voz, músicas ou vídeos era praticamente impossível alguns anos atrás, quando o tamanho dos HDs era limitado a alguns MBs, e o preço de um equipamento com maior capacidade tornava inviável a sua aquisição por um usuário comum, que desejasse sua utilização como estação de trabalho em casa por exemplo.

1.2 Objetivo do Sistema

O principal objetivo do sistema é organizar de forma adequada amostras desordenadas de vozes pertencentes a diferentes pacientes de diversas idades e condições de saúde, permitindo a avaliação e diagnóstico de cada uma delas por profissionais da área.

A organização das amostras irá simplificar a tarefa de um especialista (no caso um fonoaudiólogo, ou outro profissional relacionado à área da saúde), para que o mesmo possa analisá-las com maior eficiência, podendo reuni-las de acordo com seus próprios critérios. Por exemplo: Pode-se organizar todas as amostras que possuam o mesmo nome, todas as amostras gravadas numa mesma data, e, para o administrador do sistema, é possível ainda listar todas as amostras pertencentes a um mesmo paciente, entre outras possibilidades.

As amostras, em sua maioria, são compostas por pacientes pronunciando determinados fonemas. A maior parte das amostras analisadas possuía o nome do fonema ao qual se referia. Portanto, no parágrafo acima, quando foi dito que pode haver a separação dos arquivos de acordo com o seu nome, talvez o seguinte exemplo torne a situação mais clara:

Um profissional deseja fazer comparações entre a vogal “A” pronunciada por uma pessoa cuja amostra de voz foi classificada como normal e a mesma vogal pronunciada por uma pessoa que possui algum tipo de patologia. Nesse caso, pode-se separar todas as amostras de voz cujo nome é “A”, e, ouvindo cada uma delas e consultando o respectivo diagnóstico, é possível compará-las.

Para que o usuário do sistema possa fazer suas próprias buscas no banco, além daquelas já implementadas, foi criada uma função que permite que querys escritas diretamente em SQL sejam enviadas ao banco de dados. Dessa forma, garantimos uma maior flexibilidade para o usuário. No entanto, devemos ressaltar que essa função é estritamente restrita ao administrador do sistema, por questões de segurança e privacidade dos dados armazenados.

1.3 Organização da Monografia

Nas próximas seções, será apresentada uma breve introdução aos bancos de dados, seu funcionamento básico e também uma apresentação sucinta da linguagem utilizada para a interação com os mesmos, a Structured Query Language (SQL).

A seguir, será descrito o ambiente utilizado durante a realização do projeto. Devemos ter em mente que todo o sistema foi construído fazendo-se uso de plataformas free, o que o torna acessível para qualquer usuário interessado.

Logo em seguida, o modelo do banco de dados construído, abrangendo desde o seu modelo relacional, dependências funcionais e normalização, até a descrição de cada um dos campos criados nas tabelas da base de dados.

Será descrita em detalhes a implementação do sistema, desde os requisitos necessários para o correto funcionamento do mesmo, até os meios utilizados para garantir a segurança e a integridade dos dados armazenados.

2. Base de Dados

Como uma forma de esclarecer um pouco mais o entendimento que se tem sobre o que é de fato uma base de dados, mais comumente chamada de banco de dados, e de passar um caráter mais formal para sua definição, os próximos parágrafos terão como função descrevê-lo segundo a literatura especializada.

Na área da computação, uma base de dados pode ser definida como uma coleção de dados padronizados e organizados segundo uma estrutura, os quais podem ser armazenados em um computador de modo a permitir que qualquer programa os consulte e os utilize como resposta em buscas. Os dados retornados por essas buscas se tornam então as informações que podem ser utilizadas para a tomada de decisões pelos usuários do sistema.

O software responsável por administrar e realizar buscas nas bases de dados é conhecido como Sistema Gerenciador de Banco de Dados (SGBD). Note que comumente o termo banco de dados é utilizado como sinônimo de SGBD.

Toda base de dados trabalha com algum tipo de estrutura para manter sua organização, e, de acordo com o tipo de estrutura utilizada o banco recebe uma classificação. A essa classificação se dá o nome de esquema. O esquema descreve como os objetos são representados na base de dados. Existem diferentes modos de organizar um esquema, ou seja, modelar a estrutura de uma base de dados, os quais são conhecidos por modelos de base de dados. O modelo mais utilizado atualmente, e utilizado nesse projeto, é o modelo relacional.

O modelo relacional representa toda a informação na forma de múltiplas tabelas relacionadas, cada uma sendo formada por linhas e colunas. Este modelo representa os relacionamentos pelo uso de valores comuns para mais de uma tabela. Existem muitos outros modelos de dados, os quais não possuem interesse para serem descritos nesse trabalho.

Através de um olhar mais formal e dando uma descrição matemática para o modelo relacional podemos dizer que a principal proposição do modelo relacional é que todos os dados são representados como relações matemáticas, isto é, um subconjunto do produto Cartesiano de n conjuntos. No modelo matemático, a análise dos dados é feita em uma lógica de predicados de dois valores (ou seja, sem o valor nulo). Isto significa que existem dois possíveis valores para uma proposição: verdadeira ou falsa. Os dados são tratados pelo cálculo relacional ou álgebra relacional.

Para facilitar e padronizar as consultas nos bancos de dados relacionais, a linguagem SQL, que será descrita posteriormente, foi criada.

Devemos levar em conta também que uma máquina com capacidade adequada deve armazenar toda a base de dados. É preciso haver também um sistema de redundância (RAID ou algum tipo de backup) que garanta a segurança dos dados contra perda ou qualquer tipo de dano no hard disk.

Através de um login, os usuários autorizados poderão realizar alterações no banco (inserção, edição e remoção de informações). Os demais terão acesso restrito a esses dados, não podendo, portanto, modificá-los de qualquer forma.

Além disso, outro controle de segurança é necessário. O nome dos pacientes cujas amostras de voz estão armazenadas nessa base de dados deve permanecer sob sigilo. Assim sendo, apenas alguns usuários, que serão os administradores do sistema, terão acesso a essa informação. Os demais terão acesso a todas as outras informações sobre as amostras de vozes, mas essas irão se manter anônimas.

Outro login é então necessário. Apenas usuários com permissões de administrador do sistema terão a senha necessária para esse login.

Como citado anteriormente, o sistema permite que o usuário insira suas próprias querys, escritas em SQL, que serão enviadas diretamente ao SGBD. Devemos ressaltar que isso também só é permitido a usuários com privilégios de administrador do sistema, que terão de passar pelo mesmo critério de login citado acima.

Ressaltamos que o processo de login é idêntico ao processo utilizado para um usuário comum, no entanto, a validação de um usuário com senha de administrador lhe permite ter acesso a todas as vantagens do sistema, exatamente como o login realizado em um sistema Linux, por exemplo, onde um usuário comum não tem todos os mesmos privilégios que um administrador, no entanto, o processo de login é idêntico.

O processo de login será detalhado na seção 3.3.

2.1 Modelo Entidade-Relacionamento

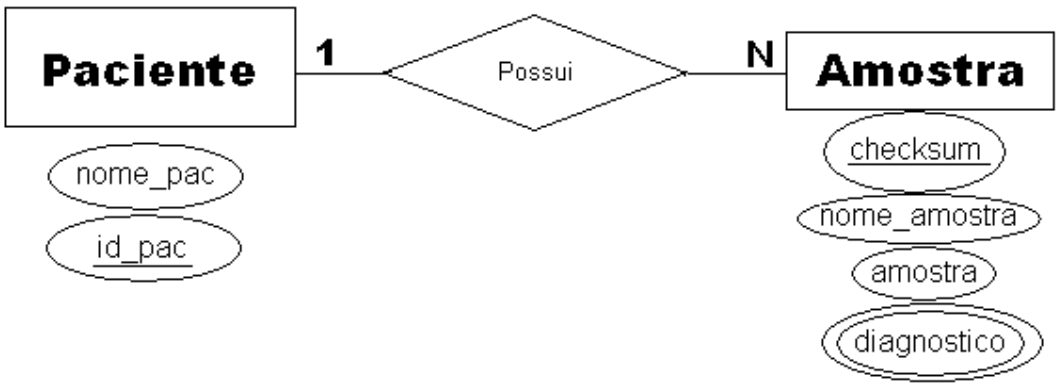

Figura 1 – Modelo entidade-relacionamento

Podemos ver na figura acima que diagnóstico é um atributo multivalorado de amostra, ou seja, cada amostra pode ter vários diagnósticos diferentes.

A figura a seguir mostra o mapeamento da base de dados. Nela, são criadas três tabelas:

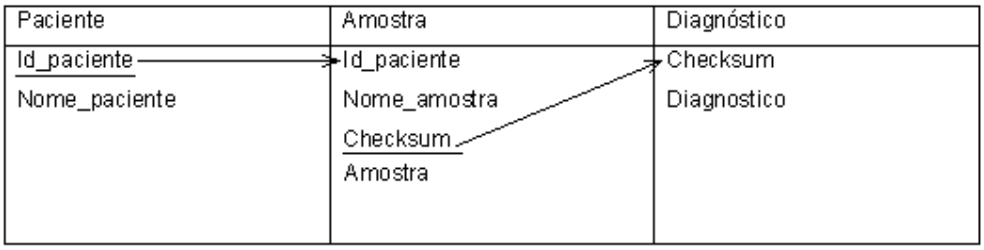

Figura 2 – Mapeamento da base de dados

Tebela Paciente:

Id_paciente: número que identifica cada paciente de maneira única.

nome_pac: nome do paciente. Essa informação é sigilosa e somente pode ser acessada pelo administrador do sistema, ou outro usuário com privilégios de adminsitrador. É a chave primária dessa tabela.

A chave primária dessa tabela é o Id_paciente, que também é uma chave estrangeira, referenciando o campo de mesmo nome da tabela amostra.

Tabela Amostra:

Id_paciente: citado acima.

nome_amostra: nome do arquivo que contém a amostra de voz.

checksum: número que identifica univocamente cada uma das amostras armazenadas na base de dados. É calculado através do algoritmo MD5 (detalhado na seção 2.9), e garante que nenhuma amostra seja duplicada na base de dados.

data: data de criação do arquivo.

amostra: amostra de voz, armazenada geralmente em formato .wav.

Essa tabela possui uma chave primária composta pelo campo checksum. Através desse campo, podemos garantir que duas ou mais amostras exatamente iguais não serão inseridas na base de dados, evitando redundância e desperdício de espaço de armazenamento. O algoritmo MD5, utilizado para o cálculo do checksum é citado na seção 2.9.

Além disso, essa chave primária é também uma chave estrangeira (checksum), que referencia o campo de mesmo nome da tabela diagnóstico. Esse relacionamento permite a conexão entre essas tabelas no momento em que buscas forem realizadas.

Tabela Diagnóstico:

checksum: citado acima.

diagnóstico: análise da amostra citada acima feita por um profissional qualificado.

A tabela Diagnóstico não apresenta chave primária. Isso se faz necessário visto que cada uma das amostras pode ter vários diagnósticos diferentes. Essa tabela apresenta também uma chave estrangeira (checksum), que referencia o campo de mesmo nome da tabela amostra.

2.2 Dependência Funcional

Primeira Forma Normal (FN1)

Uma relação esta na primeira forma normal se os valores de seus atributos são atômicos (simples, indivisíveis) e monovalorados. Em outras palavras, 1FN não permite “relações dentro de relações” ou “relações como atributos de tuplas”.

Segunda Forma Normal (FN2)

A FN2 exige que todo atributo da tabela seja dependente funcional da chave completa e não de parte da chave.

Uma tabela com uma chave formada por apenas um atributo está automaticamente na FN2.

Terceira Forma Normal (FN3)

A FN3 exige que não existam atributos transitivamente dependentes da chave. Ou seja, todo atributo que reside na tabela, deve ser determinado pela chave primária e não somente por parte dela quando esta for composta. No caso, nenhuma das tabelas apresenta chave composta.

2.3 Alguns detalhes importantes

A seguir, será descrita a criação da base de dados, bem como das tabelas a ela pertencentes.

Para criar uma base de dados, basta digitar no PostgreSQL:

createdb postgres – onde postgres é o nome da base de dados a ser criada.

A criação das tabelas na base de dados é mostrada a seguir:

A Sintaxe do comando create table, utilizado para a criação de tabelas, é a seguinte:

CREATE TABLE *nome-da-tabela* ({*definição-da-coluna* | *restrição no nível-de-tabela*}
[, {*definição-da-coluna* | *restrição no nível-de-tabela*}] *)

Portanto, as tabelas devem ser criadas da seguinte forma:

2.3.1 Tabela paciente:

qreate sequence paciente_seq;

create table paciente (nome varchar(30), ID_pac integer default nextval ('paciente_seq') not null primary key, foreign key (Id_pac) references amostra (Id_pac));

Nesse caso, a seqüência criada faz com que o SGBD incremente automaticamente o número relacionado ao id_pac, começando com 0 e incrementando-o de uma unidade a cada linha inserida na tabela.

2.3.2 Tabela amostra:

create table amostra (checksum varchar(32) not null primary key, ID_pac integer not null, nome_amostra varchar(20), data date, amostra oid, foreign key (Id_pac) references paciente (Id_pac));

O tipo de dados date permite que dados referentes à data sejam inseridos de forma adequada no banco.

O tipo de dados oid será explicado na seção 2.7.

2.3.3 Tabela diagnóstico:

create table diagnostico (diagnostico varchar (200), checksum varchar(32) not null, foreign key (checksum) references amostra(checksum));

Essa tabela não necessita de uma chave primária, visto que qualquer um dos dois campos a ela pertencentes podem se repetir, pois podemos ter vários diagnósticos para uma mesma amostra e também mais de uma amostra podem ter o mesmo diagnóstico.

2.3.4 Tabela User

Além disso, existe também a tabela responsável por armazenar os dados referentes aos usuários cadastrados no sistema, controlando a segurança ao acesso do mesmo.

Nessa tabela, são armazenados: username, password, role, system, counter, status:

Username – nome do usuário.

Password – senha criptografada do usuário.

Role – função que o usuário exerce no sistema:

- 0 – usuário comum
- 1 – administrador

System – informa se o usuário é o usuário inicial, criado pelo sistema ou um usuário adicionado posteriormente.

- 0 – system default administrator
- 1 - usuário

Quando essa tabela é criada, um usuário já é inserido, o system default administrator, cujo valor do atributo system é 1. Esse usuário não pode ser excluído, e é criado para que o programa possa ser executado pela primeira vez. Caso contrário, não existiriam usuários cadastrados e o sistema não poderia ser utilizado.

Counter: contador que é ativado quando um usuário tenta se logar no sistema utilizando uma senha inválida. Após a terceira tentativa de login inválido, o usuário fica bloqueado.

Status: Bloqueado ou não bloqueado.

- 0 – não bloqueado
- 1 – bloqueado

Quando um determinado usuário tentar acessar o sistema utilizando uma senha inválida, após a terceira tentativa o mesmo fica bloqueado por questões de segurança.

2.4 PostgreSQL

O SGBD escolhido para o projeto foi o Postgresql, que segundo a comunidade pode ser definido como: "O PostgreSQL é um SGBD (Sistema Gerenciador de Banco de Dados) objeto-relacional de código aberto, com mais de 15 anos de desenvolvimento. É extremamente robusto e confiável, além de ser extremamente flexível e rico em recursos. Ele é considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados. A equipe de desenvolvimento do PostgreSQL sempre teve uma grande preocupação em manter a compatibilidade com os padrões SQL92/SQL99."

Como uma das exigências para o projeto era trabalhar com um SGBD desenvolvido por software livre, e o PostgreSQL é um dos melhores e mais bem documentados, sua escolha foi bem justificada.

2.5 Uma breve introdução ao SQL

Devido à diversidade de linguagens e de base de dados existentes, a forma de comunicação entre umas e outras seria algo complicado de providenciar, a não ser pela existência de padrões que nos permitem realizar as operações básicas de uma forma universal.

É justamente disso que se trata a Structured Query Language, que é uma linguagem padrão de comunicação com base de dados. Falamos portanto, de uma linguagem versátil e normalizada que nos permite trabalhar com qualquer tipo de linguagem (ASP, PHP, Java, etc.) em combinação com qualquer tipo de base de dados (MS Access, SQL Server, MySQL, PostgreSQL, etc).

Os registros podem ser introduzidos a partir de sentenças que empregam a instrução Insert.

A sintaxe utilizada é a seguinte:

Insert Into nome_tabela (nome_campo1, nome_campo2,...) Values (valor_campo1, valor_campo2...)

Para apagar um registro nos servimos da instrução Delete. Neste caso devemos especificar qual ou quais são os registros que queremos apagar. Por isso, é necessário estabelecer uma seleção que se realize mediante a cláusula Where.

Delete From nome_tabela Where condicoes_de_selecao

Delete From paciente Where nome='Pedro'

A seguinte tabela mostra um exemplo da utilização da cláusula SELECT:

Tabela diagnostico

Checksum	diagnostico
00001	normal
00002	patológica
00003	normal
00004	patológica

Note que os valores utilizados são apenas ilustrativos. Os valores encontrados no campo checksum diferem significativamente dos valores exemplificados acima.

Para selecionar todos os campos Ichecksum dessa tabela, usamos a seguinte sentença:

SELECT checksum FROM diagnostico

Resultado:

checksum

00001
00002
00003
00004

Podemos também selecionar mais de um campo da tabela, o que pode ser feito através da separação do nome dos campos com o uso de vírgula na sentença SQL. No caso, podemos selecionar checksum e diagnostico através da seguinte sentença:

SELECT checksum, diagnostico FROM diagnostico;

No caso, esses dois são os unicos campos pertencentes a essa tabela, portanto, podemos utilizar a seguinte expressão para obtermos todos os campos da mesma:

SELECT * FROM diagnostico;

Resultado:

checksum	diagnostico
00001	normal
00002	patológica
00003	normal
00004	patológica

Para obter o diagnóstico da amostra de voz cujo checksum corresponde ao número 00002, por exemplo, utilizamos SELECT em conjunto com WHERE:

SELECT diagnostico FROM diagnostico WHERE checksum = 00001;

LIKE é uma outra palavra chave que pode ser utilizada em conjunto com WHERE**.**

Basicamente, LIKE permite fazer uma busca baseada em um determinado padrão ao invés de fazer uma busca exata do que é desejado. Isso ficará mais claro com o seguinte exemplo:

SELECT "column_name" FROM "table_name" WHERE "column_name" LIKE {PATTERN}

Onde {Pattern} pode ser:

- 'A_Z': Todas as strings que comecem pela letra A, tenham um caracter intermediário e terminem com a letra Z. Alguns exemplos: 'ABZ','A2Z', ‘AAZ’. Note que 'AKKZ' não satisfaz essa condição.
- 'ABC%': Todas as strings que começam porABC. Nesse caso, nada é dito a respeito dos outros caracteres. A única condição é que a string comece por ABC.
- '%XYZ': Todas as strings que terminarem em XYZ. Nesse caso, nada é dito sobre os caracteres iniciais, portanto, são strings válidas: 'WXYZ', 'ZZXYZ', etc.
- '%AN%': Todas as strings que contém ‘AN’ em qulquer parte da palavra. São strings aceitas: ‘ANTONIO’, ‘ANA’, ‘AMANDA’, etc.

Para ilustrar o uso de LIKE, temos o seguinte exemplo:

Tabela exemplo

nome	sobrenome	data
ANA	SILVA	Jan-05-2007
AMANDA	COELHO	Jan-07-2007
ANTONIO	CRUVINEL	Jan-08-2007
PAULO	SILVA	Jan-08-2007

Queremos encontrar todos os registros cujo nome contém ‘AN’, em qualquer posição da palavra. Para fazer isso, utilizamos a seguinte query:

SELECT * FROM exemplo WHERE nome LIKE '%AN%'

Resultado:

nome	sobrenome	data
ANA	SILVA	Jan-05-2007
AMANDA	COELHO	Jan-07-2007
ANTONIO	CRUVINEL	Jan-08-2007

Update é a instrução que nos serve para a modificação de registros já existentes. Como para o caso de Delete, necessitamos especificar, por meio de WHERE**,** quais são os registros em que queremos fazer efetivas nossas modificações. Ademais, obviamente, teremos que especificar quais são os novos valores dos campos que desejamos atualizar. A sintaxe é a seguinte:

Update nome_tabela Set nome_campo1 = valor_campo1, nome_campo2 = valor_campo2,...
Where condicoes_de_selecao
Um exemplo aplicado:

UPDATE paciente Set nome='José' WHERE nome='Pedro'

Mediante esta sentença mudamos o nome Pedro por José em todos os registros da tablela paciente cujo nome seja Pedro.

Aqui também há o cuidado de utilizar “WHERE”, do contrário, as modificações se extenderão por todos os registros da tabela.

2.6 Chaves

As tabelas relacionam-se umas com as outras através de chaves. Uma chave é um conjunto de um ou mais atributos que determinam a unicidade de cada registro.
Por exemplo, se um banco de dados tem como chave primária checksum, sempre que ocorrer uma inserção de dados o sistema de gerenciamento de banco de dados irá fazer uma consulta, através da chave primária da tabela, para identificar se o registro já se encontra gravado. Neste caso, um novo registro não será criado.
Temos dois tipos de chaves:
Chave primária: (*PK - Primary Key*) é a chave que identifica cada registro dando-lhe unicidade. A chave primária nunca pode se repetir. Temos como exemplo prático o CPF de uma pessoa,

que a identifica de forma única entre todas as outras, e, portanto, não pode se repetir, ou seja, não podemos ter mais de uma pessoa com o mesmo número de CPF.

Chave Estrangeira: (*FK - Foreign Key*) é a chave formada através de um relacionamento com a chave primária de outra tabela. Define um relacionamento entre tabelas e pode ocorrer repetidas vezes. Caso a chave primária seja composta na origem, a chave estrangeira também o será.

2.7 O tipo de dados OID

Para podermos armazenar de fato os arquivos de áudio no banco de dados, e não apenas registros de informações sobre os mesmos, é necessário criar um campo de dados do tipo OID ou object identifier, o qual passando o endereço do arquivo no computador para o SGBD, irá inserí-lo como um arquivo binário na base de dados.

Deste modo é criado um vínculo entre o arquivo de áudio e suas informações, tais como data de criação, tamanho da amostra, diagnósticos, entre outros, facilitando as possíveis buscas por arquivos de um mesmo paciente ou de um mesmo tipo de patologia, por exemplo, retornando não apenas as informações sobre esses arquivos mas também o próprio arquivo de áudio especificado.

2.8 Organização das amostras e inserção dos dados

Inicialmente, temos uma estrutura completamente desorganizada, com arquivos duplicados e armazenados de forma ineficiente. Por exemplo, podemos ter dois arquivos com o mesmo nome, cuja amostra de voz pertence ao mesmo paciente e que tenham sido criados na mesma data. Nesse caso, não havendo nenhuma estrutura de organização das amostras, podemos ficar em dúvida se se tratam de arquivos iguais ou não.

Antes de inserir os arquivos na base de dados, primeiro calculamos seu CRC (cyclic redundancy code) através do algoritmo MD5 de ambos. Caso o valor calculado coincida com o de alguma amostra já presente no banco, esse arquivo não é inserido e uma mensagem de erro é mostrada ao usuário.

Dessa forma, garantimos que os dados inseridos no banco não serão redundantes.

Para a inserção de arquivos na base, devemos considerar que o usuário pode inserir diversos arquivos de uma só vez, sem a necessidade de selecionar cada um deles. Dessa forma, basta selecionar um determinado diretório, e o programa o irá percorrer de forma recursiva, ou seja,

enquanto houver arquivos, ele os indexa. Se houver subpastas, ele as percorre de maneira recursiva. Caso o diretório ou caminho especificado não exista, é retornada uma mensagem de erro para o usuário. Se o diretório não tiver nenhum arquivo, exibe um alerta dizendo que o diretório está vazio. Esse comportamento não deve ocorrer nas subpastas. Se estas estiverem vazias, o programa simplesmente passará para a próxima subpasta, sem exibir o alerta.

Os arquivos gerados conterão os seguintes campos: nome, data, checksum.

O sistema deve informar ao usuário que a inserção de um determinado número de arquivos (o próprio programa deve mostrar esse número) foi realizada com sucesso.

2.8.1 Perfis de usuário

Podemos ter dois diferentes perfis de usuários: os usuários comuns e os administradores, estes últimos tendo certos privilégios que usuários comuns não têm.

Aos usuários comuns, é permitido inserir arquivos e fazer buscas por data, nome da amostra e diagnóstico.

Aos administradores, todas as funções disponíveis são acessíveis, inclusive o acesso ao nome dos pacientes. Os adminsitradores podem também criar novos usuários, com funções de usuário comum ou de adminsitrador.

A criação de um novo usuário é mostrada abaixo:

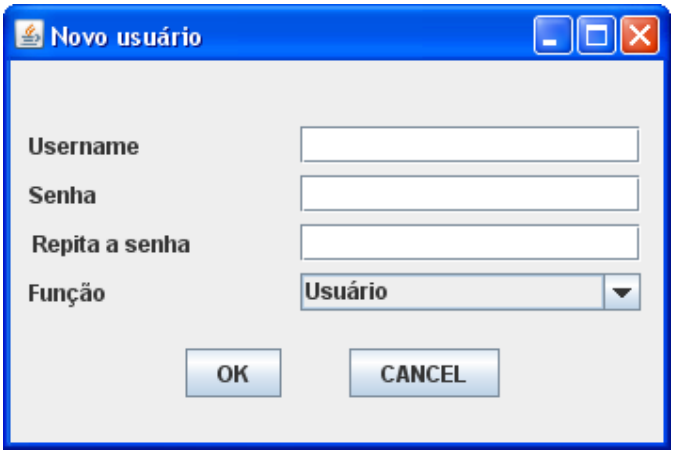

Figura 3 – Adicionando um novo usuário

2.8.2 Troca de Senha

Para que a troca de senha seja efetuada, o usuário deve fornecer seu username, sua senha atual, a nova senha, e deve também repetir a nova senha, como medida de segurança, para que não haja erros de digitação e uma senha indesejada acabe sendo armazenada.

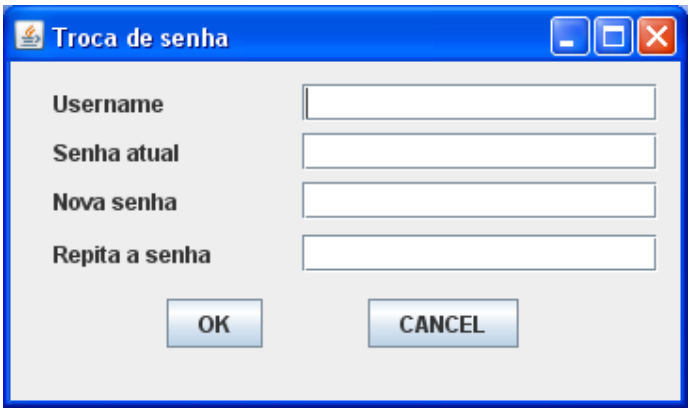

Figura 4 – Troca de senha

2.9 Algoritmo MD5

Message digest algorithm 5. É uma função hash de 128 bits unidirecional amplamente utilizada em criptografia, normalmente para verificação de integridade de arquivos e logins.

Foi desenvolvido em 1991 por Ronald Rivest para suceder ao MD4 que tinha alguns problemas de segurança. Por ser um algoritmo unidirecional, uma hash md5 não pode ser transformada novamente no texto que lhe deu origem. O método de verificação é, portanto, feito pela comparação das duas hash (no caso, uma hash de cada um dos arquivos a ser comparado).

Os exemplos abaixo mostram como uma mínima alteração (mesmo a troca de uma letra maiúscula por uma minúscula) no arquivo irá produzir um valor completamente diferente:

MD5 ("The quick brown fox jumps over the lazy dog")

Figura 5 – MD5 ("The quick brown fox jumps over the lazy dog")

MD5 ("The quick brown fox jumps over the lazy ***c***og")

Figura 6 - MD5 ("The quick brown fox jumps over the lazy ***c***og")

MD5 ("TCC")

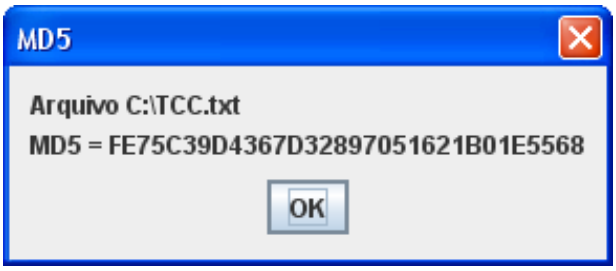

Figura 7 - MD5 ("TCC")

MD5 ("Tcc")

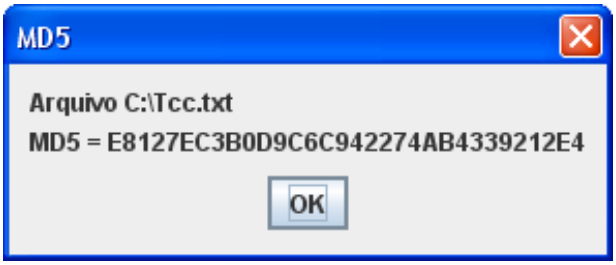

Figura 8 - MD5 ("Tcc")

MD5 ("") -> string vazia

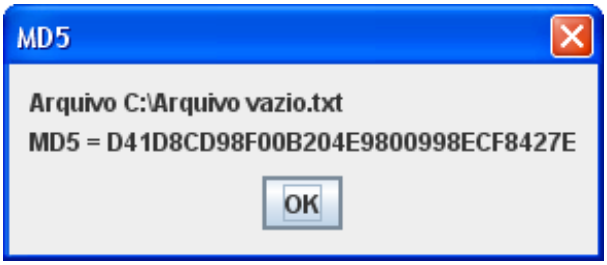

Figura 9 - MD5 ("")

Com a utilização deste algoritmo, podemos assegurar que dois arquivos diferentes, mesmo que a diferença entre os mesmos seja mínima, terão valores hash diferentes, portanto, eliminamos o problema de inserir dois arquivos iguais ou deixar de inserir um arquivo não redundante.

Esse método também é utilizado como medida de segurança na autenticação de usuários. Como sabemos, o algoritmo MD5 é unidirecional, ou seja, após a criptografia, é impossível recuperar o dado original. Assim sendo, a senha do usuário é inserida no banco criptografada por esse algoritmo e, sempre que um usuário tentar fazer login no sistema, sua senha criptografada é comparada com a senha armazenada na base de dados (também criptografada). Se ambas coincidirem, o usuário é autenticado. Caso contrário, aciona-se um contador que permite três tentativas de login inválidas, e, em seguida bloqueia o usuário por questões de segurança. O processo de bloqueio de um usuário por excesso de tentativas de login inválidas é detalhado na seção 3.3.

3 Programa

O programa é constituído de uma janela principal que contém opções de inserir, remover ou editar um arquivo. Para cada uma dessas operações, uma janela permite que o usuário selecione qual ou quais arquivos devem ser inseridos, removidos ou editados.

Devemos considerar o caso em que um usuário deseja operar a base de dados com uma grande quantidade de arquivos, o que tornaria inviável a inserção (ou qualquer outra operação) de um arquivo de cada vez. Assim sendo, não só é possível especificar um determinado arquivo para ser inserido (ou sofrer qualquer outro tipo de operação), mas também é possível selecionar uma pasta de arquivos, de tamanho indeterminado, facilitando, portanto, a manipulação de um grande volume de arquivos.

Qualquer usuário pode ter acesso aos arquivos armazenados, no entanto, as operações acima citadas são permitidas apenas aos usuários que possuírem uma senha, fornecida pelo administrador do sistema, e pode ser alterada pelo próprio usuário. O processo de autenticação é explicado com mais detalhes na seção 3.3.

A base de dados possui o nome de todos os pacientes cujas amostras de vozes estão armazenadas, no entanto, essa informação deve manter-se restrita apenas ao administrador do sistema. Assim sendo, quando um usuário tentar acessar o nome de um paciente, outra autenticação é requerida. Observe que um usuário com privilégios para inserir ou apagar um arquivo não necessariamente terá privilégio para acessar o nome de um paciente. O processo de autenticação para acesso ao nome do paciente é explicado em detalhes na seção 3.3.

3.1 Fluxo de Telas:

O programa se inicia com a seguinte tela principal:

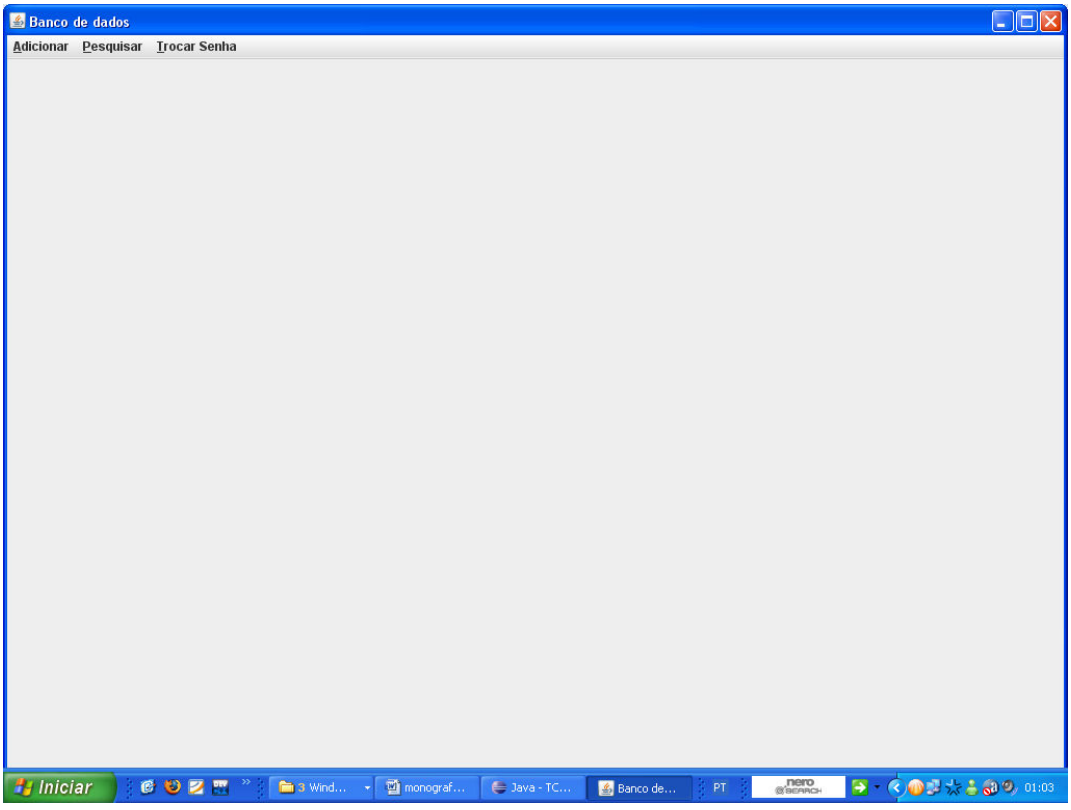

Figura 10 – Janela principal do programa

Através do menu Adicionar (cuja tecla de atalho é Alt+A), podemos escolher um arquivo ou uma pasta de arquivos que serão inseridos na base de dados.

A inserção de arquivos ou pastas na base de dados só é permitida a usuários que possuam uma senha de acesso ao sistema. Ao clicar sobre esse menu, o usuário deve fornecer seu login e senha para que a função se torne acessível.

Nesse mesmo menu, podemos escolher a opção sair, case desejemos encerrar o programa. Essa função pode ser alcançada através da combinação de teclas CTRL+S.

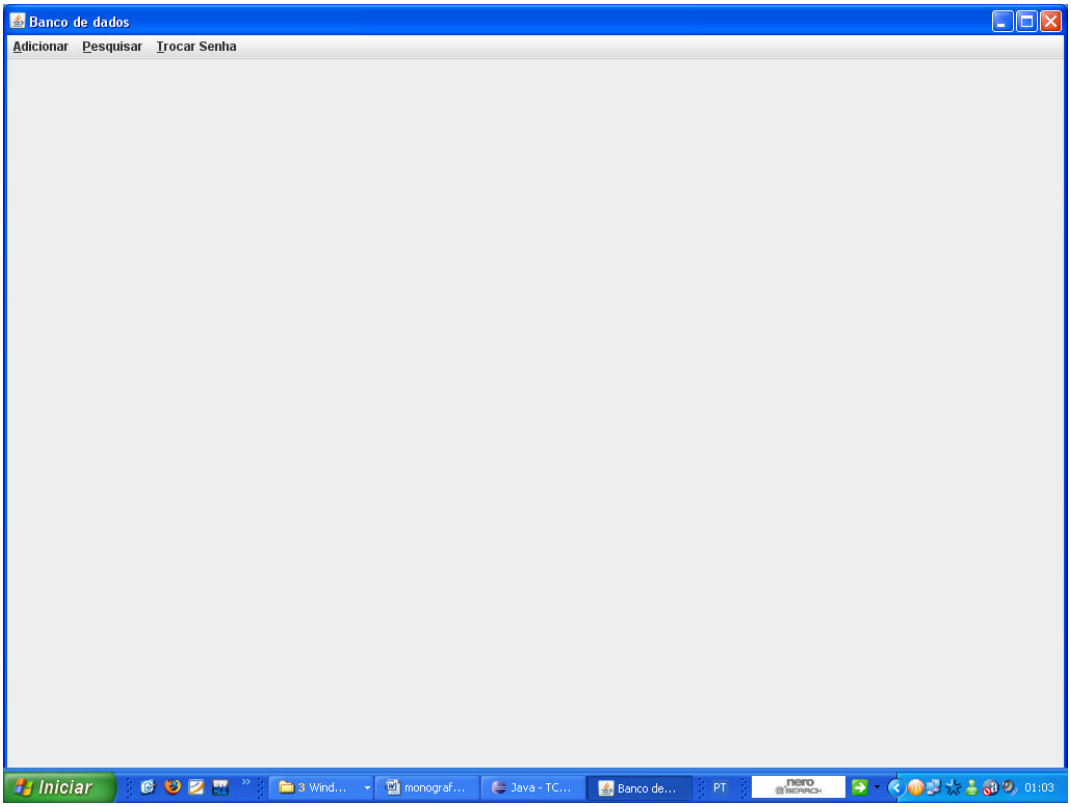

Figura 11 – Menu adicionar

Nas operações acima, será sempre necessário selecionar o arquivo ou pasta que será inserido (a), editado (a) ou excluído (a). Para isso, utilizamos a classe JFileChooser, pertencente a javax.swing.JComponent, que consiste em uma janela nos permite navegar pelo computador até encontrar o arquivo ou pasta desejado.

O menu pesquisar permite a qualquer usuário ouvir as amostras de voz armazenadas no sistema. A combinação de teclas para acessar esse menu é ALT+P.

As buscas podem ser realizadas pelo nome da amostra, nome do paciente (no caso de um administrador do sistema) ou pelo diagnóstico.

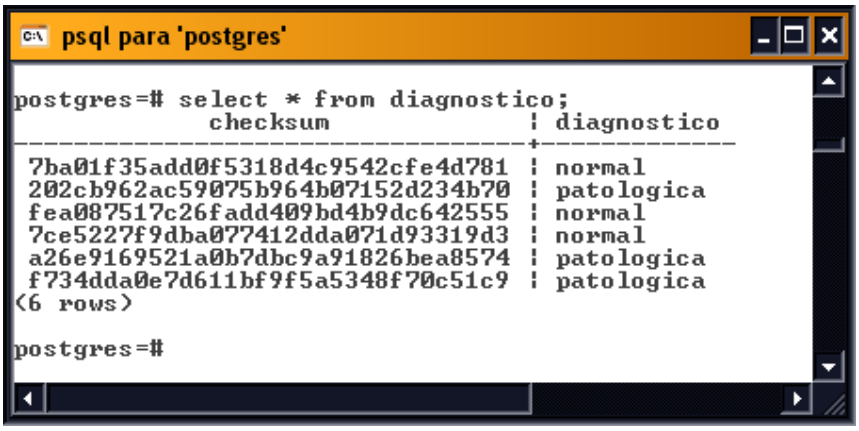

Figura 12 - Tabela diagnóstico

Podemos recuperar todas as amostras cujo diagnóstico é “normal”, por exemplo. No entanto, podemos ter algumas amostras cujo diagnóstico seja um pouco diferente, como no caso de uma patologia, o dignóstico pode indicar que a amostra apresenta a patologia e também o tipo de patologia. O campo diagnóstico tem um espaço de 200 caracteres para uma breve descrição da patologia.

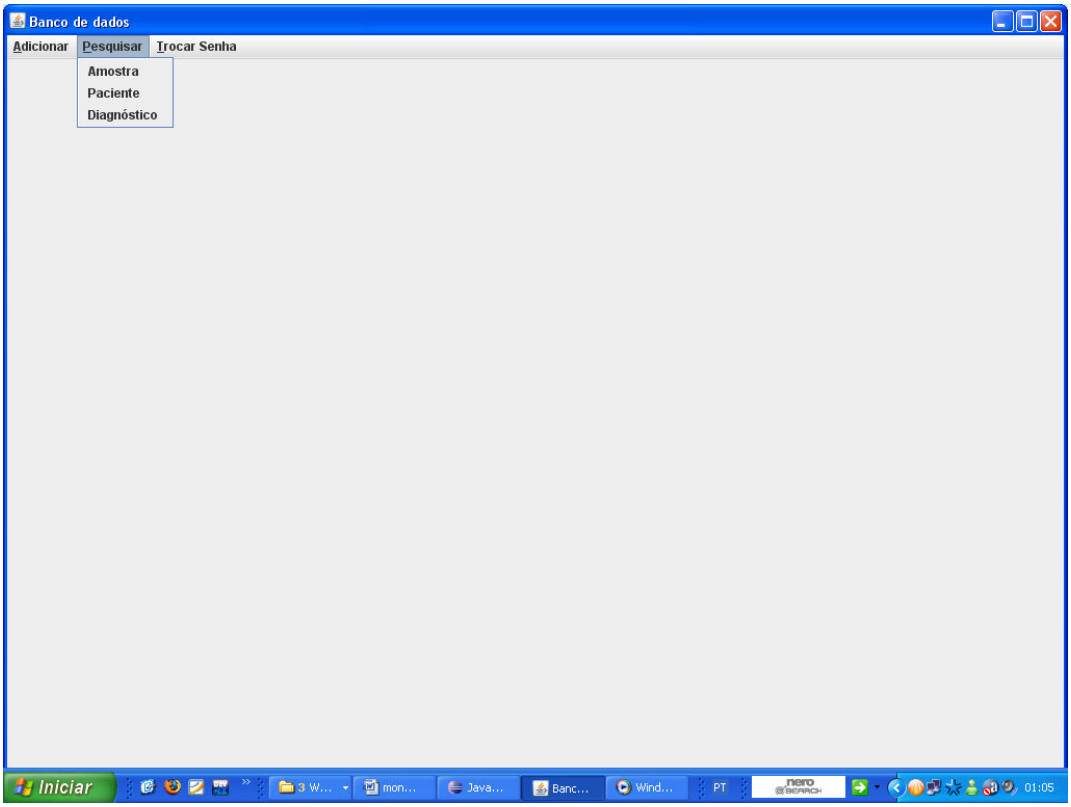

Figura 13 – Menu Pesquisar

Através desse menu, podemos pesquisar amostras através do diagnóstico, nome da amostra e, para o administrador do sistema, através do nome do paciente.

Através do menu trocar senha, é possível que o usuário que está logado no sistema no momento mude a sua senha.

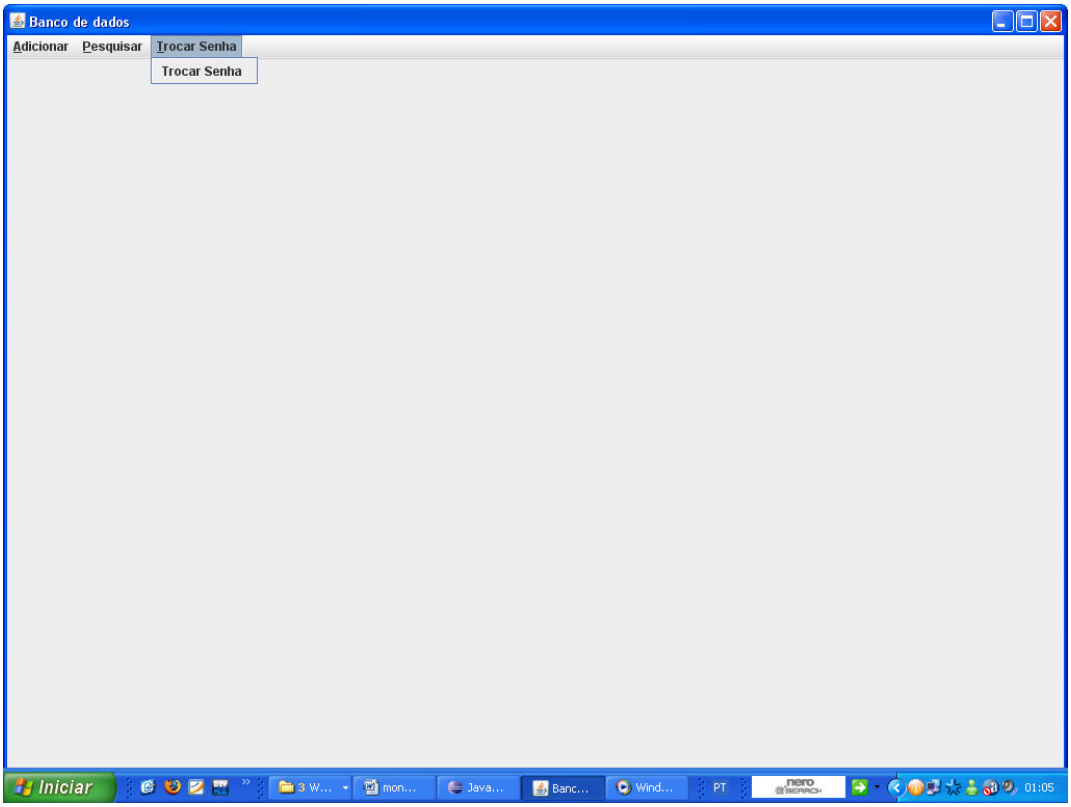

Figura 14 – Menu Trocar Senha

3.2 Escolhendo um arquivo

Sempre que for solicitado ao usuário selecionar um arquivo, ou um diretório, para que o mesmo não tenha que digitar o caminho do arquivo ou diretório desejado, é apresentada uma solução mais simples, através de um file chooser, uma classe definida pela linguagem java que permite a navegação através da máquina do usuário até que o mesmo encontre o arquivo desejado. A janela que realiza essa interação é mostrada abaixo:

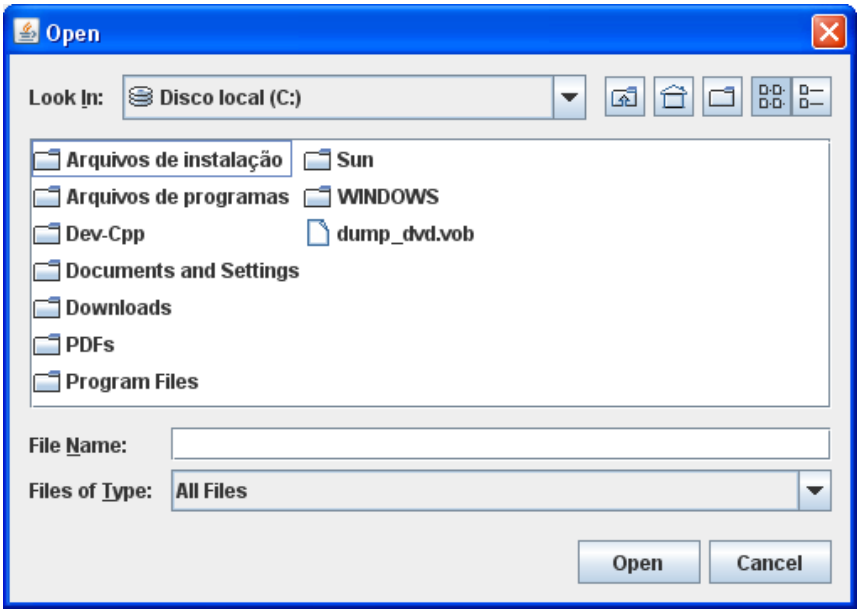

Figura 15 – Escolhendo um Arquivo

Essa janela é utilizada em praticamente todos os softwares que permitem que o usuário insira, remova ou edite algum arquivo, tornando-o facilmente alcançável. Através dela janela, podemos percorrer todos os discos rígidos e diretórios do computador, até encontrar o arquivo ou pasta procurada. Essa classe permite selecionar tanto um arquivo quanto uma pasta.
A seleção de arquivos é feita dessa forma para inclusão, edição ou remoção de um arquivo ou pasta.

3.3 Validação do usuário

O sistema trabalha com perfis de usuário, ou seja, existem dois tipos de usuários, o usuário comum e o adminsitrador. Como os próprios nomes mostram, o adminsitrador terá alguns privilégios a mais que o usuário comum.
Os nomes de usuário e respectivas senhas são armazenados em uma tabela no banco de dados, conforme descrito na seção 2.3.4.

A validação do usuário é feita através de:

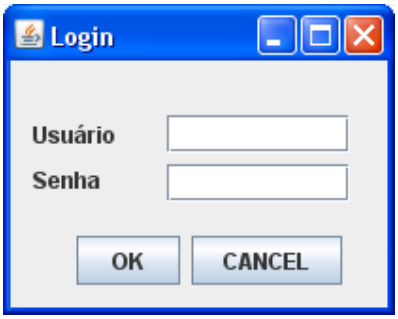

Figura 16 - Validação

A senha capturada por essa tela é criptografada e comparada com a senha (também criptografada) presente no banco de dados para o usuário informado.

Caso esse usuário não se encontre na base de dados, uma mensagem de erro será retornada:

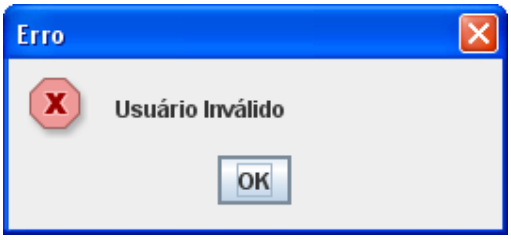

Figura 17 – Usuário inválido

Caso as senhas criptografadas sejam diferentes, uma mensagem de erro de senha inválida é retornada:

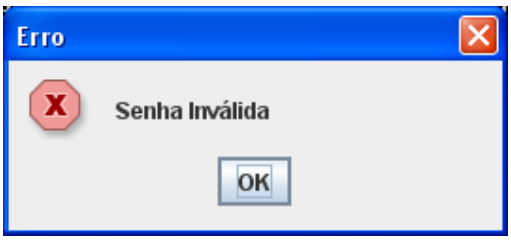

Figura 18 – Senha inválida

Caso o usuário digite três vezes a senha incorretamente, por motivos de segurança, o mesmo torna-se bloqueado no sistema. Isso é feito alterando o valor do campo status da tabela user de 0 (desbloqueado) para 1 (bloqueado), como citado na seção 2.3.4.

Se a senha incorreta for digitada um número de vezes menor que três, e em seguida a senha correta, o campo counter é resetado.

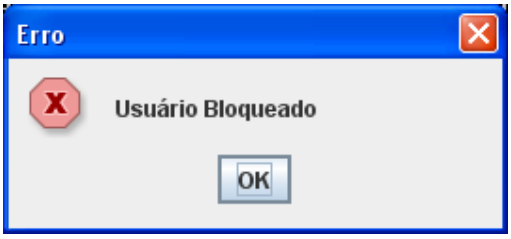

Figura 19 – Usuário bloqueado

Caso a senha digitada seja a senha correta, as funções do sistema se tornam disponíveis, de acordo com o perfil do usuário.

Note que sempre que o usuário informar uma senha inválida, o contador (counter) é incrementado. Para evitar um eventual problema de overflow desse campo, que pode ser ocasionado por usuários mal intencionados realizando um grande número de logins inválidos, após 3 tentativas utilizando uma senha incorreta o campo status recebe valor 1, indicando que o usuário que estava tentando se logar foi bloqueado e o campo counter para de ser incrementado.

A tela abaixo mostra o estado da base de dados no seguinte caso: o usuário lbpupo tentou se logar 3 vezes com uma senha inválida. Depois disso, uma quarta tentativa de login utilizando uma senha inválida foi realizada, e então uma tentativa de login utilizando a senha correta.

```
psql para 'postgres'
 username |             password             | role | system | counter | status
----------+----------------------------------+------+--------+---------+--------
 leopupo  | bcbe3365e6ac95ea2c0343a2395834dd |    0 |        |         |
 leonardo | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 lpupo    | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 pupo     | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       0 |      0
 lbpupo   | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       0 |      0
(5 rows)

postgres=# select * from users;
 username |             password             | role | system | counter | status
----------+----------------------------------+------+--------+---------+--------
 leopupo  | bcbe3365e6ac95ea2c0343a2395834dd |    0 |        |         |
 leonardo | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 lpupo    | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 pupo     | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       0 |      0
 lbpupo   | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       1 |      0
(5 rows)

postgres=# select * from users;
 username |             password             | role | system | counter | status
----------+----------------------------------+------+--------+---------+--------
 leopupo  | bcbe3365e6ac95ea2c0343a2395834dd |    0 |        |         |
 leonardo | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 lpupo    | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 pupo     | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       0 |      0
 lbpupo   | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       2 |      0
(5 rows)

postgres=# select * from users;
 username |             password             | role | system | counter | status
----------+----------------------------------+------+--------+---------+--------
 leopupo  | bcbe3365e6ac95ea2c0343a2395834dd |    0 |        |         |
 leonardo | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 lpupo    | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 pupo     | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       0 |      0
 lbpupo   | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       3 |      1
(5 rows)

postgres=# select * from users;
 username |             password             | role | system | counter | status
----------+----------------------------------+------+--------+---------+--------
 leopupo  | bcbe3365e6ac95ea2c0343a2395834dd |    0 |        |         |
 leonardo | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 lpupo    | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 pupo     | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       0 |      0
 lbpupo   | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       3 |      1
(5 rows)

postgres=# select * from users;
 username |             password             | role | system | counter | status
----------+----------------------------------+------+--------+---------+--------
 leopupo  | bcbe3365e6ac95ea2c0343a2395834dd |    0 |        |         |
 leonardo | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 lpupo    | 7ba01f35add0f5318d4c9542cfe4d781 |    0 |      0 |       0 |      0
 pupo     | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       0 |      0
 lbpupo   | 7ba01f35add0f5318d4c9542cfe4d781 |    1 |      0 |       3 |      1
```

Figura 20 – Login inválido

Note que após as três tentativas inválidas, o contador pára de ser incrementado e o status muda de 0 para 1.

3.4 Inserindo novos usuários

O sistema permite que novos usuários sejam inseridos, bastando, para tanto, que um nome de usuário, uma senha e um papel no sistema (administrador ou usuário comum) sejam fornecidos.

A inserção de um novo usuário é mostrada a seguir:

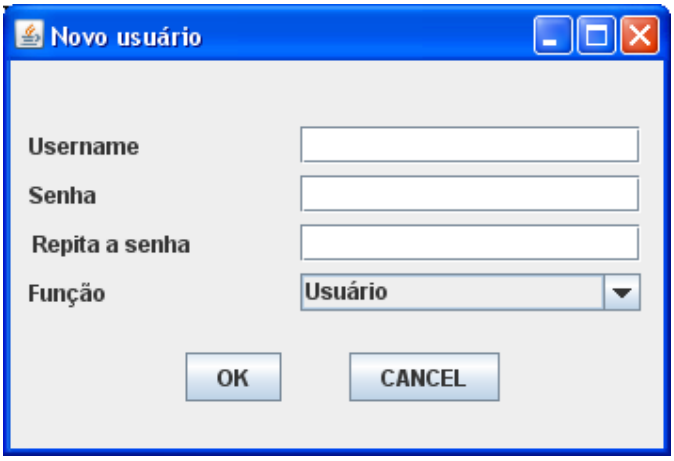

Figura 21 – Inserção de um novo usuário

Note que um administrador pode incluir na base de dados um usuário comum ou um outro usuário com privilégios de administrador. No entanto, um usuário comum pode apenas criar um novo usuário comum.

3.5 Armazenando os arquivos na base de dados

Para armazenar os arquivos de voz, é recomendado que os mesmos se encontrem gravados no formato .wav, já que nenhuma perda de dados devido à compressão é desejada. Para a nossa aplicação, mais importante que economia de espaço é a integridade dos dados. No entanto, a aplicação não impede que arquivos gravados em outros formatos sejam inseridos.

A inserção é feita através da seguinte query:

insert into amostra values (ID_amostra, nome_amostra, lo_import('caminho_da_amostra'), ID_paciente, data_criacao, tamanho);

Tudo isso é feito automaticamente através do file chooser discutido na seção 3.2.

Ao selecionar um arquivo, seu caminho completo é passado para lo_import, que faz a inserção do arquivo de som na base de dados.

Todos os outros argumentos são também obtidos automaticamente, sendo que a única preocupação do usuário é selecionar o arquivo (ou pasta de arquivos) a ser inserido através do file chooser.

Consideramos, através das amostras de voz obtidas, que o nome do paciente é determinado pelo nome do diretório no qual a amostra se encontra armazenada. Dessa forma, o programa insere no campo nome do paciente o nome do diretório no qual a amostra está registrada.

No caso de uma inserção de uma pasta de arquivos, o file chooser retorna seu caminho completo, o qual é utilizado por uma função que varre recursivamente essa pasta, retornando o

nome de cada um de seus arquivos. Caso houverem sub pastas, essa função percorre-as da mesma maneira. Dessa forma, podemos inserir rapidamente todos os arquivos de um determinado diretório, sem a necessidade de percorre-los manualmente um a um.

Nenhum tipo de verificação é feita até então. Se tentarmos inserir dois ou mais arquivos que possuam a mesma chave primária (as chaves primárias das tabelas criadas encontram-se descritas na seção 2.6) o SGBD irá retornar um erro, informando que o arquivo não pôde ser inserido. Nesse caso, inserimos apenas um dos arquivos e descartamos o outro.

Como a chave primária da tabela amostra é o checksum de cada arquivo de voz, pelo processo acima podemos garantir que a base de dados nao irá conter mais de uma amostra igual, o que evita redundâncias.

3.6 Buscas

Ao realizar uma busca, podemos optar por buscar por um nome de usuário (apenas para administradores), data em que a amostra foi gravada, nome da amostra ou diagnóstico.

Ao escolher o critério utilizado na busca, o resultado é uma tabela, contendo as informações requisitadas.

A figura abaixo mostra uma pesquisa pelo nome das amostras. Nesse caso, é possível executar a amostra selecionada, excluí-la da base de dados, ver todos os diagnósticos a ela relacionados, ou até mesmo escrever um novo diagnóstico.

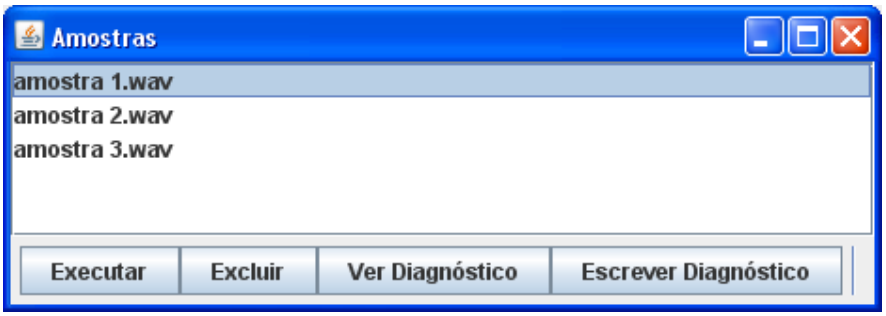

Figura 22 – Pesquisa pelo nome da amostra

Clicando sobre qualquer uma delas, a mesma é executada através do seguinte player:

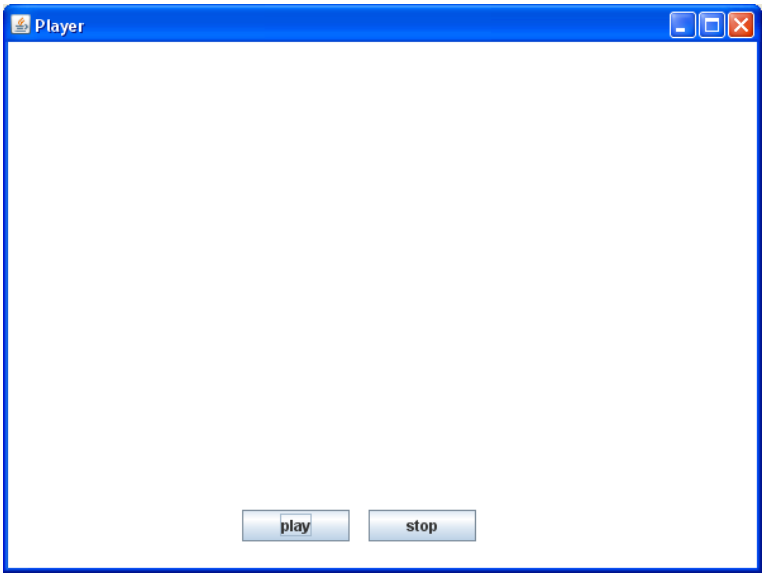

Figura 23 – player

Nele, podemos clicar sobre “play” para executar o arquivo ou “stop” para interromper sua execução.

Podemos também pesquisar os pacientes cujas amostras estão presentes na base de dados:

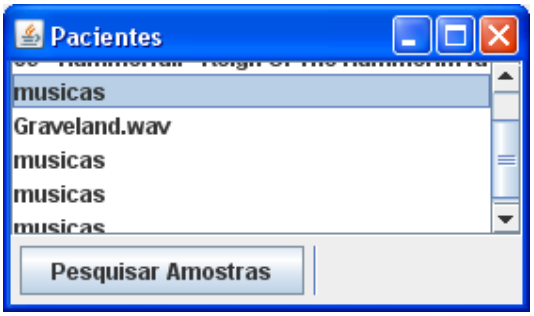

Figura 24 – Pesquisa pelo nome dos pacientes

Nesse caso, para o paciente selecionado, podemos pesquisar todas as amostras a ele relacionadas.

3.7 Login Remoto

Todas as operações citadas anteriormente podem ser executadas em uma máquina cliente, acessando uma base de dados remota simplesmente informando o IP do servidor.

É necessário, no entanto, que o PostgreSql esteja configurado para receber conexões remotas, o que pode ser feito editando o arquivo postgresql.conf. Além disso, é necessário verificar que não haja nenhum tipo de firewall bloqueando o acesso à porta utilizada pelo SGBD. Essa porta pode ser configurada nesse mesmo arquivo citado, mas seu valor default é 5432.

3.8 Resultados obtidos

Após a implementação do sistema, foi possível obter um maior nível de organização para as amostras de áudio, permitindo também que novos arquivos possam ser inseridos sem prejudicar a organização do sistema. Dessa forma, garantimos uma reestruturação adequada dos arquivos já existentes e permitimos uma expansão ordenada da base de dados, conforme as necessidades futuras.
Podemos ter acesso, portanto, de forma gratuita, simples e ordenada a uma vasta coleção de dados que podem ser utilizados tanto para formação de profissionais ligados à saúde, como fonoaudiólogos por exemplo, quanto para diagnósticos médicos em pacientes, utilizando suas amostras de vozes gravadas de forma adequada.

3.9 Dificuldades e Limitações

O sistema não pôde ser testado de forma adequada devido ao cronograma curto, visto que, além das disciplinas cursadas durante o semestre, ainda há a realização do estágio obrigatório, que ocupa uma parcela considerável de tempo. O planejamento do projeto incluía alguns dias de teste da aplicação, no entanto, isso não foi possível de ser realizado. Os testes feitos foram aplicados durante a implementação do sistema, no entanto, não houve a execução de um planejamento de testes para garantir o funcionamento do mesmo nas mais diversas condições.
O comportamente do sistema em condições normais se deu de acordo com as expectativas desejadas, no entanto, o correto funcionamento em condições adversas, geralmente alcançadas durante a sua utilização no cotidiano, em que situações não tratadas podem ocorrer, não pode ser garantido.

4. Conclusões

Pela realização do projeto, foi possível perceber que cada vez mais há uma estreita ligação entre os seres humanos e as máquinas, que facilitam tarefas antes só realizadas por pessoas.

A automatização vem crescendo nas áreas ligadas à saúde, já que as máquinas tem a capacidade de realizar as mesmas tarefas que os seres humanos, no entanto, com muito mais precisão, maior eficiência e não sofrem de problemas emocionais, sono, estresse, que podem comprometer uma cirurgia, por exemplo.

No caso estudado, a máquina é utilizada como ferramenta para facilitar o trabalho de seres humanos, tornando-o mais organizado, simples e eficiente.

Antes da implementação do sistema, as buscas eram feitas de maneira desordenada, era necessário despender uma quantidade considerável de tempo buscando as amostras que atendiam às necessidades do momento, por exemplo: se quisessémos buscar todas as amostras cujo diagnóstico revelava uma voz patológica, deveríamos percorrer todos os arquivos, buscando algo em sua nomenclatura que nos desse essa informação, e nem sempre isso estava presente. Após a implementação do sistema, basta pesquisar pelos arquivos que tenham sido diagnosticados como patológicos, que os mesmo são rapidamente apresentados de forma simples e clara.

As amostras de vozes são armazenadas de maneira a facilitar seu acesso, sua comparação com outras, permitindo a um profissional trabalhar em conjunto com uma grande quantidade de dados para que possa formar um diagnóstico correto, baseado em um amplo campo amostral.

5 Referências Bibliográficas

1. Gong, Li; Ellison, Gary; Dageforde, Mary – “Inside Java 2 Platform Security: Architecture, API Design, and Implementation”, 2nd Edition.
2. Deitel, H.M, - “JAVA How to Program”, 4th edition.
3. Niemeyer, Patrick - Learning Java, 3rd Edition.
4. Reese, George – “Database programming with JDBC and Java”, 2nd Edition.
5. Eckel, Bruce – “Thinking in Java”, 2nd Edition.
6. Van Haecke, Bernard – “JDBC 3.0 Java Database Connectivity”.
7. Elliott, James – “Java Swing” 2nd Edition.
8. Van der Linden, Peter – “Just Java 2” - 4th edition.

9. Roberts, Simon; Heller, Philip; Ernest, Michael – “The complete JAVA 2 certification study guide”.
10. Jepson, Brian – “Java database programming”.
11. Mello, Rodrigo – “Aprendendo Java 2”.
12. Momjian, Bruce – “PostgreSQL Introduction and Concepts”.
13. Elmasri, R – “Fundamentals of database systems”.
14. Gorman, M – “Database management systems”.
15. Date, C. J.- “Introdução a sistemas de bancos de dados”.
16. Ramakrishnan, Raghu; Gehrke, Johannes – “Database Management Systems”, 2nd Edition.
17. Fowler, Martin – “Refactoring: Improving the Design of Existing Code”.
18. Douglas, Korry; Douglas, Susan – “The Comprehesive guide to building, programming and administering PostgreSQL databases”, 2nd Edition.