UNIVERSIDADE DE SÃO PAULO

Escola de Engenharia de São Carlos – Departamento de Engenharia Elétrica

---

BANCO DE DADOS VOZES

Escola de Engenharia de São Carlos – SP

Fernando Demartine Cruvinel

BANCO DE DADOS VOZES

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

Curso de Engenharia de Computação

ORIENTADOR: **Prof. Dr. José Carlos Pereira**

São Carlos

2007

Agradecimentos

Agradeço a todas as pessoas as quais me ajudaram a chegar até aqui.

Com especial atenção à minha família, e meus professores, sem os seus esforços não teria conseguido.

Queria agradecer ao docente: Prof. Dr. José Carlos Pereira pela oportunidade dada para a realização deste projeto e por sua orientação, ao docente: Prof. Dr. Carlos Dias Maciel pela ajuda inestimável dada durante todo o desenvolvimento do banco, agradeço a meus amigos: Valter Rogério Messias, Breno Henrique Leitão e Henrique Barbosa pela contribuição em quesitos técnicos envolvendo a linguagem Java e banco de dados PostgreSQL.

ÍNDICE

LISTA DE FIGURAS

RESUMO

Este projeto tem como intuito a construção de um **banco de dados de vozes**, com o objetivo de organizar e facilitar o acesso ao grande acervo de amostras de vozes possuído pelo laboratório de instrumentação do departamento de engenharia elétrica da EESC, o qual é utilizado em diversos projetos.

Neste trabalho será descrito como foi planejado e montada essa base de dados, sendo apresentados os conceitos utilizados de maneira sucinta, e explicações para decisões tomadas durante o projeto.

O aplicativo desenvolvido para a facilitar a utilização do banco e as possíveis busca ao acervo foi implementado utilizando-se a linguagem de programação **JAVA**, devido a sua portabilidade e a facilidade que oferece para a interface com bancos de dados **JDBC**, o banco de dados escolhido foi o **PostgreSQL** e o sistema operacional **LINUX**; um dos requisitos do laboratório era a utilização de ferramentas sob a licença de software livre.

A necessidade do banco estava no fato de que a maior parte dos arquivos,em formato **WAVE**, se encontrava armazenadas em mídias óticas, ou em hard disks do laboratório, cada qual com o sua própria estruturação e padrão diferentes, dificultando muito o trabalho de encontrar amostras para outros projetos.

Os arquivos foram armazenados no banco utilizando-se do tipo de dado OID, possibilitando o armazenamento como objeto binário longo, portanto sem modificações.

Abstract

This Project has the intuit to build a **voice database**, with the objective to organize and to make easier the access to the big amount of voice samples held by the lab of electronic instrumentation.

During this work will be descript how was planned and built the database, the concepts needed for the projected will be briefly explained as well as the justifications for the project decisions.

The API was desenvolved to become easier the access and queries to the database, and was implemented in **JAVA** language, as a way to bring portability and because of the **JDBC** tool which offers a great interface with DBMS, the DBMS chosen was the **PostgreSQL** and the operational system **LINUX**; it is wise to remember that these choices were made to attend the lab request to use open source tools.

The database was needed because the majority of the files were stored in distinct optical discs and hard drives, in the file format **.WAV**, each one with its own structure and pattern, making hard to find samples to work in other projects.

The samples were stored into the database in the data type OID, which means that they were saved as binary large objects, thus the files were not modified.

1. Introdução:

1.1 Contextualização, Motivação e Domínio de Aplicação -

A necessidade de se organizar dados, criar padrões, sistematizar funções é de fundamental importância para o bom funcionamento de qualquer organização ou entidade que trabalhe e gere um grande número arquivos ou objetos.

Não é de se estranhar que exista um grande empenho do setor de TI em desenvolver tecnologias que facilitem este tipo de tarefa, trazendo soluções cadê vez mais inovadoras para o mercado.

Sem dúvidas nenhuma, uma das ferramenta que se tornou peça chave nesta área foram os chamados Sistemas Gerenciadores de Banco de Dados, os SGBDs, que vêm evoluindo constantemente ao longo dos anos.

Mas precisamente podemos dizer que desde os anos 60 esses sistemas vem sendo desenvolvidos. Na época, procurava-se desenvolver modelos que utilizassem mais efetivamente os novos dispositivos de memória secundária de acesso direto, que substituíam os cartões perfurados e as fitas magnéticas, resultando nos modelos de banco de dados em rede, e um modelo mais simplificado, o modelo hierárquico.

Foi na década de 70 que foi proposto o modelo predominante até hoje, o modelo relacional, entretanto a implementação do modelo exigia pesquisas, e só na década de 80 começou a ganhar o mercado, se tornando líder a partir dos anos 90.

Atualmente já se encontra em desenvolvimento um novo modelo, o chamado banco de dados orientados a objetos, provavelmente o sucessor do relacional, que vem de encontro ao paradigma proposto pelas linguagens orientadas a objeto. Já existem modelos intermediários entre os dois, que preparam o cenário para uma possível transição.

A implementação de banco de dados no cenário atual é rotineira, se tratando de uma ferramenta utilizada pelos mais diferentes setores, desde uma simples empresa de contabilidade ou uma padaria, a qual precisa manter um cadastro de seus clientes de forma organizada, até uma grande indústria onde é necessário manter bem catalogado toda uma gama de informação de estoque, até uma lista de fornecedores, clientes, rotas de logísticas entre outras necessidades.

Isso mostra a versatilidade dessa ferramenta, não apenas no ponto de vista o qual aborda os mais diversos setores, mas também na liberdade e complexidade da sua implementação, podendo ser desenvolvido de acordo com as necessidades do usuário.

Por isso se tornou uma das soluções mais utilizadas para o problema de armazenamento, busca e organização de dados em um ambiente computacional, o qual se tornou ao longo do tempo praticamente inevitável a qualquer tipo de organização.

Um dos fatores mais interessantes de uma base de dados está no fato de que ela pode relacionar não apenas dados rotineiros como números, tabelas ou nomes, enfim tipos de dados relativamente simples, sendo representados por números inteiros e reais ou uma cadeia de caracteres, podendo também armazenar tipos de dados complexos, pelo menos para a realidade

computacional, como arquivos de imagens, áudio, planilhas eletrônicas, documentos de texto entre outros, resumindo qualquer tipo de objeto que possa ser representado como um objeto formado por um conjunto de bytes.

Sendo que o banco de dados ainda permite que estes tipos de dados mais complexos sejam relacionados normalmente com os outros. Sempre lembrando que a modelagem de um banco de dados procura relacionar as informações de uma maneira estruturada e intuitiva, levando em consideração que as consultas à base, as quais serão realizadas depois são de vital importância, pois o maior objetivo de uma base de dados é transformar dados em informações relevantes e úteis para os usuários.

1.2 Objetivo do Sistema

O laboratório de instrumentação do departamento de Engenharia Elétrica, portador de um grande número de amostras de vozes em formato eletrônico, mas infelizmente salvas de maneira desorganizada em mídias distintas, já há algum tempo veio com a idéia de transformar seu acervo em um banco de dados, como sendo uma evolução natural para o gerenciamento do acervo e facilitar a disponibilidade deste tipo de informação para seus diferente projetos de pesquisa. Sendo proposta pelo orientador deste trabalho como sendo um bom tema para o trabalho de conclusão do curso deste dois alunos.

O projeto vem então com a proposta de implementar uma base de dados de vozes e uma ferramenta gerenciadora do mesmo, que possibilite a manutenção da base e a realização de buscas, num ambiente LINUX através de ferramentas sob a licença de software livre.

O projeto foi desenvolvido utilizando a linguagem de programação JAVA, da ferramenta de interface com banco de dados JDBC, disponibilizada pela linguagem, da IDE Netbeans, e do SGBD PostgreSQL.

1.3 Organização da Monografia:

Nos próximos tópicos deste trabalho, serão apresentados e explanados os conceitos abordados no projeto, também será descrito o modelo conceitual do banco de dados, explicando sua estrutura e organização. Assim como uma visão geral da ferramenta desenvolvida para o acesso e manutenção da base de dados.

O primeiro passo mostra o cenário no qual se encontravam armazenadas as amostras, dando uma dimensão do problema a ser solucionado, quais soluções serão pospostas, basicamente seria um estudo de caso.

Depois serão brevemente explanados alguns dos conceitos utilizados no projeto, com o objetivo de tentar prover um entendimento melhor do projeto a pessoas que não estão familiarizadas com a implementação de um banco de dados.

Então será detalhada a modelagem do banco de dados, desde o seu modelo entidade relacionamento, seu mapeamento em tabelas para o modelo relacional, e sua construção em linguagem sql no banco de dados.

Em seguida é dada uma visão geral do aplicativo que está sendo desenvolvido para gerenciar e realizar buscas no banco de dados.

Por fim é apresentada a conclusão do projeto e as referências bibliográficas do mesmo.

2. Uma breve análise do problema e possíveis soluções

Os dados existentes antes de serem inseridos na base dados se encontravam armazenadas de forma completamente desorganizada e desestruturada. Não era difícil de se encontrar dois arquivos com o mesmo nome, cuja amostra de voz pertencia ao mesmo paciente e criados na mesma data.

Também foi fornecido uma mídia óptica, um DVD, contendo amostras e outros arquivos de áudio que não eram amostras, como arquivos de músicas por exemplo, ou seja uma fonte com vários arquivos não elegíveis.

Como o intuito do projeto não envolveu a análise sonora dos arquivos, como uma possível análise do formato de onda, ou uma análise no domínio da freqüência, ou seja um estudo do áudio em si, a solução para esse problema tem de ser encontrada com o arquivo em si, e não o seu conteúdo.

Por isso foi concentrada as atenções para os atributos que poderiam gerar informações suficientes para diferenciar os arquivos, evitando assim a inserção de amostras ilegíveis e arquivos redundantes.

A primeira idéia foi à verificação dos nomes dos arquivos, mas isso levou ao problema de que os arquivos não estavam nomeados de uma forma padronizada, por exemplo, um mesmo paciente possuía seu nome escrito de várias formas diferentes. Amostras coletadas com procedimentos distintos também foram encontradas. Isso dificultou muito a possível implementação de um parser, que poderia separar os arquivos pelo seus nomes, pois este tipo de implementação levaria a distinção de um mesmo paciente como um sendo um outro a cada vez que seu nome fosse escrito de uma maneira distinta.

Outra idéia foi à verificação do tamanho do arquivo e de sua data de criação, mas sozinhos não fornecem um conjunto de informações boas o suficiente para evitar a inserção de arquivos redundantes, pois várias amostras podem apresentar uma data de criação igual e o tamanho das amostras tende a serem semelhantes.

Foi então que surgiu a idéia de se calcular um checksum para cada arquivo, uma chave que garante a sua unicidade. Existem vários algoritmos que calculam uma soma praticamente única para cada arquivo, baseado em sua cadeia de código binário.

Já para evitar que arquivos de formatos distintos, ou seja não elegíveis, possam ser inseridos na base, uma solução seria evitar os arquivos que não fossem do formato wave fossem selecionáveis pelo aplicativo; mas isso tiraria muito da flexibilidade do sistema, pois existem diversos formatos de dados para áudio que oferecem uma forma de armazenamento lossless, os quais poderiam ser de interesse para o projeto. Sendo uma área na qual ainda é necessário buscar uma solução mais adequada. A solução mais simples é considerar que o usuário tentará armazenar apenas amostras de vozes no banco.

Retornando para o problema anterior, para evitar que arquivos redundantes fossem inseridos na base, agora é calculado o checksum do arquivo através do algoritmo MD5 . Caso ocorram valores de checksum iguais, significa que os arquivos são idênticos. Caso contrário, trata-se de dois arquivos distintos, assim um deles é renomeado e ambos são inseridos. Isso também previne que amostras iguais, mas com nomes diferentes nos arquivos inicias não sejam inseridas no banco.

Ainda assim é preciso organizar os dados existentes para que fiquem de uma forma homogênea, permitindo que eles sejam inseridos na base de dados. Identificando quando possível, amostras distintas de um mesmo paciente, uma vez que nos arquivos originais era comum se encontrar diretórios com o nome de um paciente, contendo as amostras do mesmo. Por isso foi decidido identificar o paciente pelo nome que o diretório contendo suas amostras foi nomeado.

O programa então deverá verificar antes de realizar a inserção, se a base de dados já possui alguma entrada com as mesmas características do arquivo a ser inserido. Se isso acontecer, a inserção não acontece.

Para a inserção de arquivos na base, deve-se considerar que o usuário poderia querer inserir diversos arquivos de uma só vez, sem a necessidade de selecionar cada um deles. Desta forma, basta selecionar um determinado diretório, e o programa o iria percorre-lô de forma recursiva, enquanto houvessem arquivos, ele os indexaria. Se houvessem sub pastas, ele as percorria de maneira recursiva também.

O sistema deveria informar ao usuário que a inserção de um certo número de arquivos foi realizada com sucesso.Caso a operação seja abortada antes de ser concluída, deveria ser efetuado um rollback no banco de dados para manter a sua consistência. Caso contrário, a operação é confirmada e os dados seriam armazenados com sucesso.

3. Revisão de Conceitos

3.1 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[1]. Os dados retornados por essas buscas se tornam então as informações que podem ser utilizadas para a tomada de decisões pelos usuários do sistema.

[1] David M. Kroenke, Database Processing: Fundamentals, Design, and Implementation (1997), Prentice-Hall, Inc., pages 130-144

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 sinômino 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.

3.2 - SQL

SQL (Structured Query Language) é uma linguagem de computador usada para criar, obter, alterar e deletar dados de um sistema gerenciador de banco de dados relacionais. Foi padronizada pela ANSI e também pela ISO[2].

O SQL foi desenvolvido com um propósito específico, consultar dados em uma base de dados relacional, por isso se trata de uma linguagem declarativa, e não uma linguagem imperativa como o C.

Podemos citar algumas palavras chaves da linguagem e suas classificações:

CONSULTAS - podem ser realizadas através do comando SELECT, por exemplo:

```
SELECT * FROM books WHERE price > 100.00 ORDER BY title
```

MANIPULAÇÃO DE DADOS - Data Manipulation Language (DML), que fornece os comandos de INSERT, UPDATE, MERGE e DELETE, responsáveis por modificar os dados presente na base de dados, exemplo:

[2] http://www.sqlsummit.com/People/JMelton.htm

```
INSERT INTO my_table (field1, field2, field3) VALUES ('test', 'N', NULL);

UPDATE my_table SET field1 = 'updated value' WHERE field2 = 'N';
```

CONTROLE DE TRANSAÇÕES - Quando disponível, usado para controlar as alterações feitas na base, fornece os comandos BEGIN WORK(marca o início de uma transação), COMMIT (confirma que todas as modificações feitas na base serão permanentes) e ROLLBACK(faz com que a base de dados retorne ao estado do último COMMIT ou ROLLBACK, descartando qualquer mudança feita após isso) exemplo:

```
BEGIN WORK;

UPDATE inventory SET quantity = quantity - 3 WHERE item = 'pants';

COMMIT;
```

Definição de dados - Data Definition Language (DDL) permite ao usuário definir novas tabelas e associar elementos, os comandos principais são: CREATE,ALTER,RENAME,TRUNCATE e DROP, exemplo:

```
CREATE TABLE my_table (

  my_field1   INT,

  my_field2   VARCHAR (50),

  my_field3   DATE        NOT NULL,

  PRIMARY KEY (my_field1, my_field2)

);
```

CONTROLE DE DADOS - Data Control Language (DCL). Cuida da parte de autorização dos dados, definindo qual usuário tem poder para acessar e ou manipular os dados da base. Os dois comandos principais são GRANT(garante privilégio) e REVOKE(retira privilégio), exemplo:

```
GRANT SELECT, UPDATE ON my_table TO some_user, another_user.
```

3.2.1 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 *ID_amostra*, 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.

3.3 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.

3.4 JDBC

Java Database Connectivity ou JDBC é uma API para a linguagem de programação JAVA, a qual define como um cliente pode acessar uma base de dados. Ela fornece métodos para a realização de consultas e alterações de dados na base. É voltada para banco de dados relacionais[3].

A API permite múltiplas implementações coexistirem e serem utilizadas pela mesma aplicação. Fornecendo um mecanismo para o carregamento dinâmico dos pacotes JAVA corretos e os registrando com o JDBC Driver Manager, responsável por criar as conexões JDBC.

As conexões permitem a criação e execução de sentenças. Estas sentenças podem ser sentenças de atualização como por exemplo SQL CREATE, INSERT, UPDATE e DELETE ou podem ser consultas utilizando o SELECT. Procedimentos guardados podem ser invocados através de sentenças. AS sentenças podem ser do seguinte tipo:

Statement: a sentença é enviada ao servidor da base a cada vez.

PreparedStatement: uma sentença é guardada, e então o caminho de execução é pré-determinado no servidor da base, deixando com que ela seja executada várias vezes de uma maneira eficiente.

CallableStatement: usado para execução de procedimentos guardados na base de dados.

[3] http://java.sun.com/javase/6/docs/technotes/guides/jdbc

Sentenças do tipo INSERT, UPDATE e DELETE retornam uma contagem de atualização que indica quantas linhas foram modificadas na base de dados. Essas sentenças não retornam qualquer outro tipo de informação. Consultas retornam uma linha JDBC result set. A linha result set é utilizada para caminhar sobre o resultado da consulta. Colunas individuais em uma linha são resgatadas pelo nome ou pelo número da coluna. A linha result set possui metadados que descrevem os nomes das colunas e seus tipos.

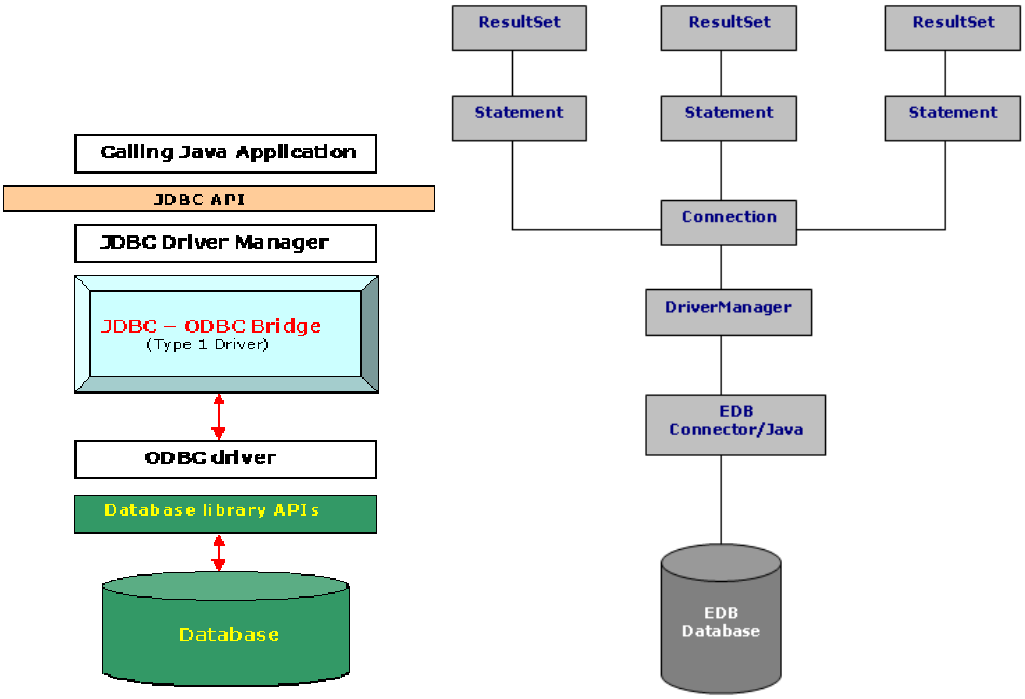

Figura 1 – A interface JDBC

3.5 WAVE

WAV ou WAVE, abreviação para Waveform audio format, formato padrão para se armazenar aúdio em computadores pessoais desenvolvido pela Microsoft e IBM[4]. É uma variação do formato RIFF, que armazena fluxo de bits em blocos. Estes formatos são muito parecidos com os formatos IFF e AIFF usados em computadores Macintosh. Tanto arquivos WAV como AIFF são compatíveis nos sistemas operacionais Windows e Macintosh. O formato leva em consideração algumas diferenças dos processadores Intel, como a ordenação de bytes inteiros little-endian.

O formato RIFF atua como um "embrulho" para vários codecs de compressão de áudio, se trata do principal formato usado em ambiente Windows para áudio cru.

O formato wave normalmente contem áudio sem nenhum tipo de compressão, apenas sendo modulado por PCM (Pulse Code Modulation) a qual é ilustrada nas figuras abaixo. Áudio

[4] Ostrovsky, L. A. and Potapov, A. S. (1999). *Modulated Waves, Theory and Applications*. Baltimore: The Johns Hopkins University Press.

PCM é o formato padrão para CDs, com uma taxa de amostragem de 44.100 hz e 16 bits por amostra. PCM se trata de um método que não aplica nenhum tipo de perda (lossless) para o arquivo, mantendo todas as amostras de uma trilha de áudio, usuários profissionais podem utilizar WAV para qualidade máxima de áudio. O formato também oferece facilidade para edição e manipulação com uma certa facilidade através de softwares.

A limitação do formato fica por conta dos arquivos terem de ser menores que 4 gigabytes, porque ele utiliza inteiros de 32 bits não sinalizados para gravar o tamanho do arquivo no cabeçalho. Na qualidade de CD isto significa aproximadamente 6 horas e meia de áudio, entretanto especialistas podem precisar de arquivos maiores. Por isso foi criado o fomato W64 para o utilização com o software de edição Sound Forge, que permite arquivos muito maiores.

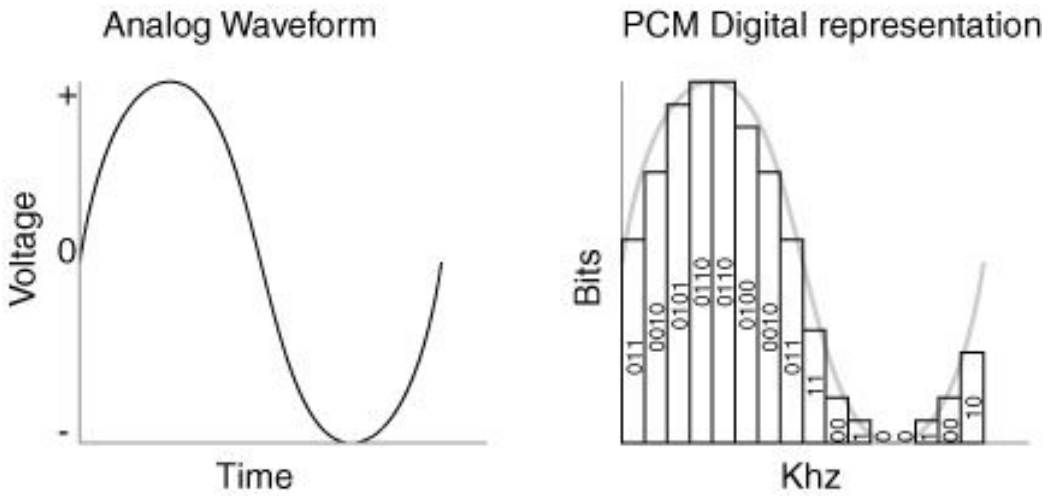

Figura 2 – A modulação PCM, a representação de um sinal e sua posterior representação em PCM.

3.6 Algoritmo MD5

Message digest algorithm 5. É uma função hash de 128 bits unidirecional amplamente utilizada em criptografia, usada normalmente para verificação de integridade de arquivos e logins, também é bastante utilizada para se verificar a consistência de grandes arquivos baixados na internet. O servidor fornece o checksum do arquivo original , para ser comparado com o arquivo baixado pelo cliente, caso sejam encontrados valores distintos significa que o arquivo do cliente se encontra corrompido[5].

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).

---

[5] Berson, Thomas A. (1992). "Differential Cryptanalysis Mod 2^{32} with Applications to MD5". *EUROCRYPT*: 71–80. ISBN 3-540-56413-6.

Os exemplos abaixo ilustrados pelas figuras, 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 3 – MD5("The quick brown fox jumps over the lazy dog")

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

Figura 4 - MD5("The quick brown fox jumps over the lazy ***c***og")

MD5("TCC")

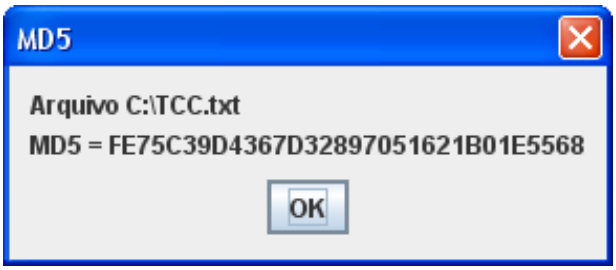

Figura 5 - MD5("TCC")

MD5("Tcc")

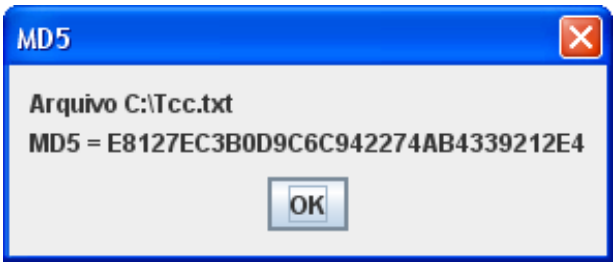

Figura 6 - MD5("Tcc")

MD5("") -> string vazia

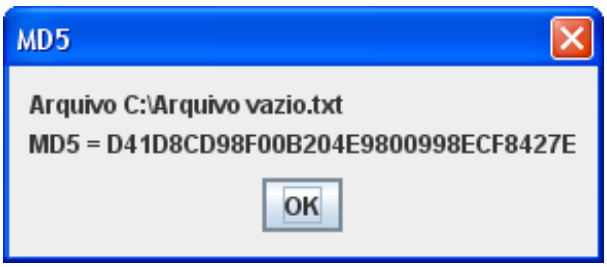

Figura 7 - MD5("")

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.

4. Modelagem e implementação da base de dados

4.1 Modelo Entidade-Relacionamento

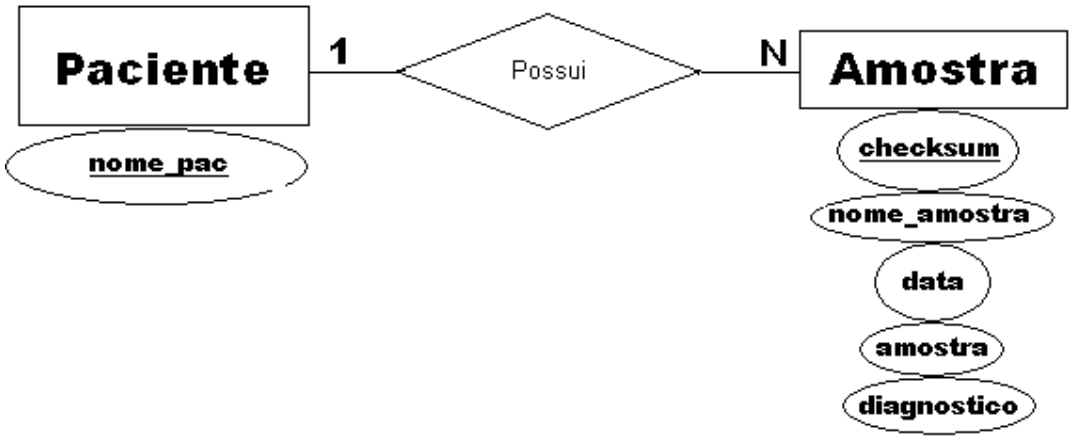

Figura 8 - Modelo entidade-relacionamento

Entidade Paciente:

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.

Entidade Amostra:

checksum: é que número que identifica univocamente cada uma das amostras armazenadas na base de dados, gerado através do algoritmo md5.

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

data: data de criação do arquivo.

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

diagnostico: análise da amostra citada acima feita por um profissional qualificado, se trata de um atributo multivalorado, uma amostra pode possuir vários diagnosticos diferentes.

Relacionamento Possui

Representa o relacionamento entre as entidades paciente e amostra, apresenta cardinalidade de 1 para N, ou seja um paciente pode ter várias amostras.

Mapeamento

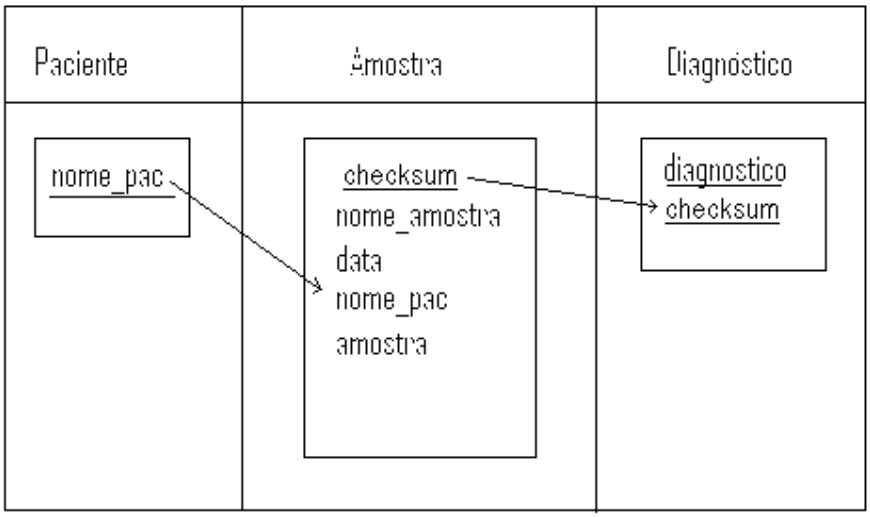

Figura 9 – O mapeamento

O mapeamento acima representa o modelo entidade relacionamento da fig X, ou seja mostra como o modelo é representado e de fato criado no banco no formato de tabelas.

A tabela Paciente foi mapeada com a chave primária nome_pac, não possuindo outros atributos.

A tabela Amostra possui uma chave primária, checksum. Isso significa que não podemos inserir duas amostras de voz identicas. Nesse caso, consideramos que as amostras podem ser iguais, e, antes de permitir tal tipo de redundância, devemos nos certificar se as amostras são realmente iguais ou não, o que é feito através do algoritmo MD5, citado na seção 2.9. Ela também apresenta uma chave estrangeira (nome_pac), que referencia o campo de mesmo nome da tabela paciente. Esse relacionamento permite a conexão entre essas tabelas no momento em que buscas forem realizadas.

A tabela Diagnóstico apresenta uma chave primária composta pelos seus dois atributos. Isso se faz necessário visto que cada uma das amostras podem ter vários diagnósticos diferentes. Essa tabela apresenta também uma chave estrangeira (checksum), que referencia o campo de mesmo nome da tabela amostra. Esse relacionamento permite que as três tabelas possam ser conectadas no momento de uma busca. Isso torna possível que possamos ter acesso a todos os diagnósticos de todas as amostras de um determinado paciente, por exemplo. A situação descrita ilustra um caso no qual as chaves estrangeiras foram utilizadas para acessar as três tabelas utilizando informações conhecidas de apenas uma delas (nome do paciente).

A tabela Diagnostico foi criada para respeitarmos as regras de normalização. Como o atributo diagnostico, da entidade Amostra, se tratava de um atributo multivalorado, ou seja uma amostra pode ter vários diagnósticos diferentes, foi necessário criar a tabela diagnóstico com a chave estrangeira checksum de amostra.

4.2 Dependência Funcional

Primeira Forma Normal (FN1)

Uma relação esta na primeira forma normal se os valores de seus atributos são atômicos (simples, indivisíveis) e mono valorados. 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.

Quarta Forma Normal (FN4)

A FN4 não permite que exista dependência multivalorada, e para isso sugere a criação de novas tabelas durante o processo de mapeamento aonde existam atributos multivalorados dependentes de outros atributos.

4.3 Criação da base

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:

Tabela paciente:

create table paciente (nome_pac varchar(40) primary key);

Tabela amostra:

create table amostra (Checksum integer not null primary key, nome_amostra varchar(20), data varchar(10), amostra oid, foreign key (nome_pac) references paciente (nome_pac));

O tipo de dados oid será explicado na seção 2.7.

Tabela diagnóstico:

create table diagnostico (diagnostico varchar (500) primary key, foreign key (checksum) references amostra(checksum) primary key);

4.4 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.

Abaixo segue uma figura da interface pgAdmin do Postgresql.

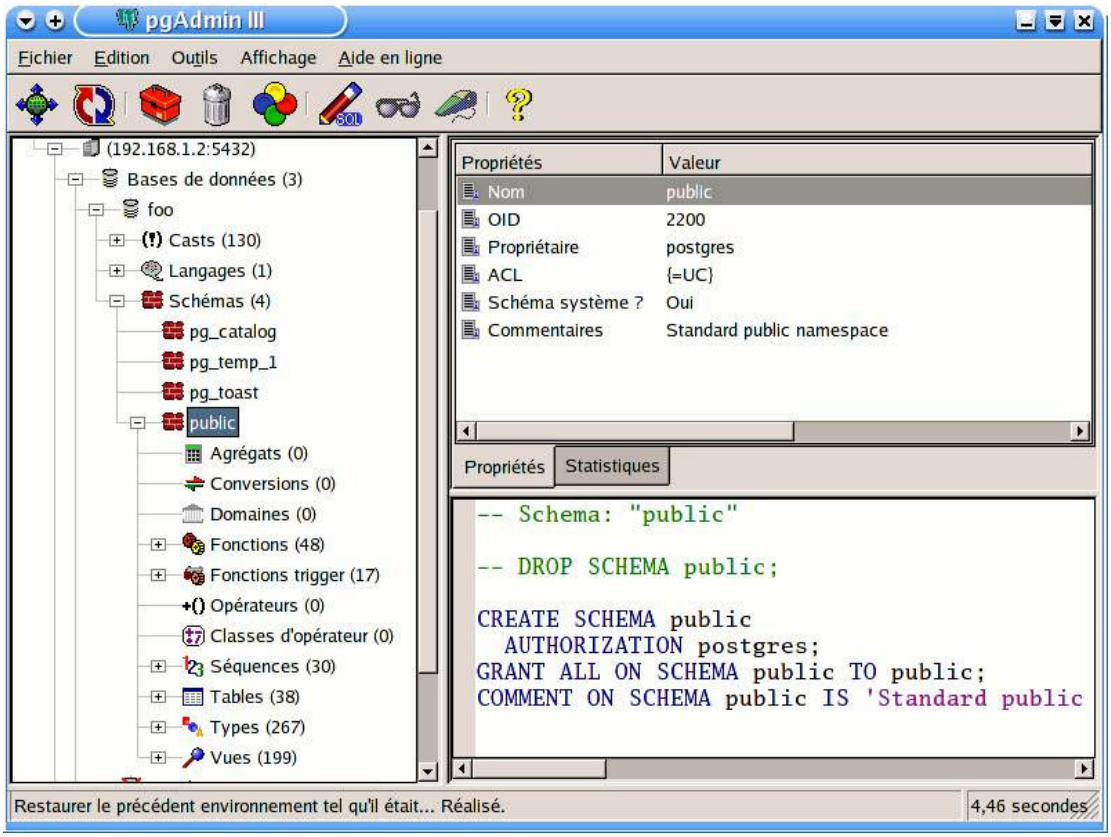

Figura 10 – A interface gráfica do Postgres

5. O Programa

O aplicativo desenvolvido para oferecer uma interface ao usuário capaz de gerenciar o banco de dados e realizar as buscas é constituído de uma janela principal que contém opções de inserir, remover ou editar um arquivo. Então uma janela dá ao usuário a opção de escolher qual tipo de arquivo ele quer inserir, remover ou editar suas informações.

Pode ser interessante ao usuário trabalhar na base de dados com uma grande quantidade de arquivos, nesta ocasião, será possível determinar o arquivo para ser inserido (ou sofrer qualquer outro tipo de operação), como também selecionar uma pasta de arquivos, com inúmeras amostras, facilitando a seleção de um grande número de arquivos.

Para que um arquivo possa ser inserido, excluído ou deletado da base de dados, é necessária uma autenticação, o que garante que apenas usuários que possuam tais privilégios alterem a base.

Um sistema de login para os usuários foi criado no banco de dados, deste modo é possível determinar quais usuários possuem quais tipos de privilégio, ou seja qual usuário pode apenas realizar consultas, qual usuário pode realizar ajustes nas informações armazenadas, quem pode gerenciar a base e inserir ou remover arquivos; e também determinar qual usuário tem acesso a possíveis campos restritos nas tabelas, como por exemplo o nome dos pacientes.

Vale lembrar que um usuário com privilégios para inserir, deletar ou editar um arquivo não necessariamente terá privilégio para acessar o nome de um paciente. Cada privilégio é determinado independentemente, sendo concedido apenas pelo usuário com poderes de administrador do banco.

5.1 Fluxo de Telas:

Neste tópico será apresentado uma visão geral do aplicativo em desenvolvimento, mostrando todas as funções principais do programa, vale lembrar que algumas modificações na interface gráfica podem ser apresentadas na versão final do mesmo, com objetivo de tornar mais fácil a manipulação deste software. As telas são mostradas nas figuras abaixo referentes a cada seção explanada.

O programa é iniciado com a seguinte tela:

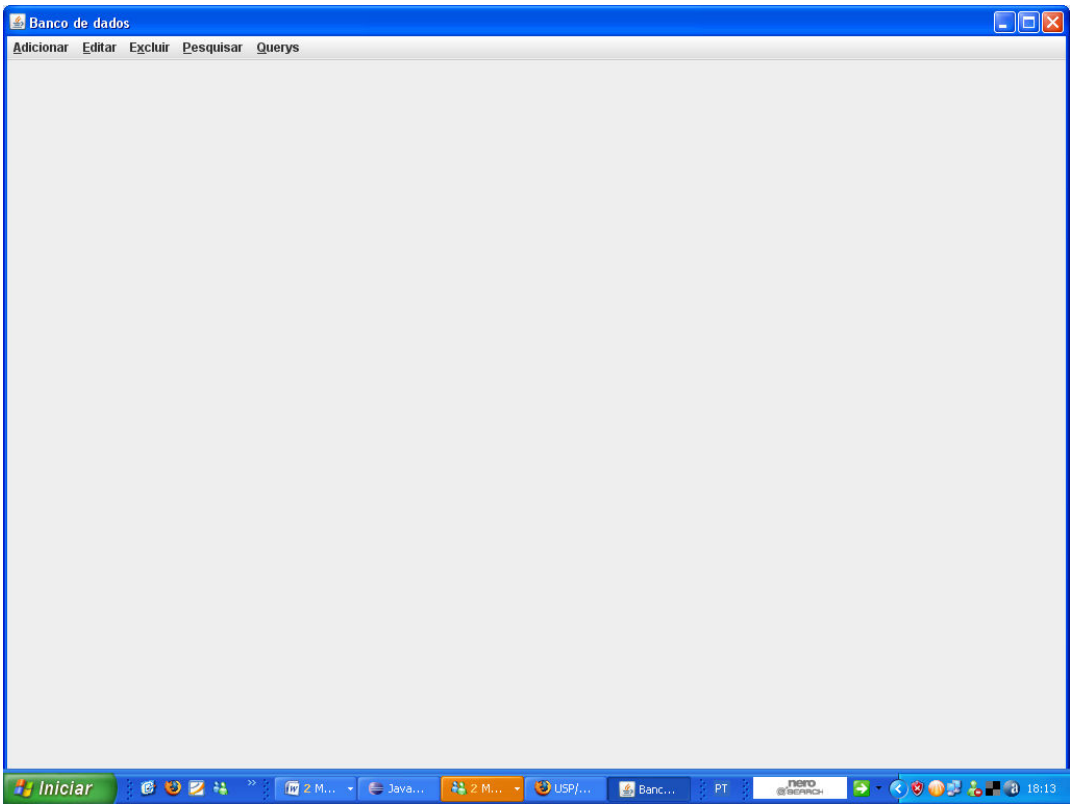

Figura 11 – Janela inicial do programa

O menu Adicionar (cuja tecla de atalho é Alt+A), deixa o usuário selecionar um arquivo ou uma pasta de arquivos, permitindo inserir os mesmos na base de dados.

Qualquer tipo de operação na base de dados, apenas será permitida após o usuário se logar no mesmo. Portanto a inserção de arquivos ou pastas na base de dados só é permitida a usuários que possuam tais privilégio.

Neste mesmo menu também é permitido escolher a opção sair. Essa função pode ser alcançada através da combinação de teclas CTRL+S.

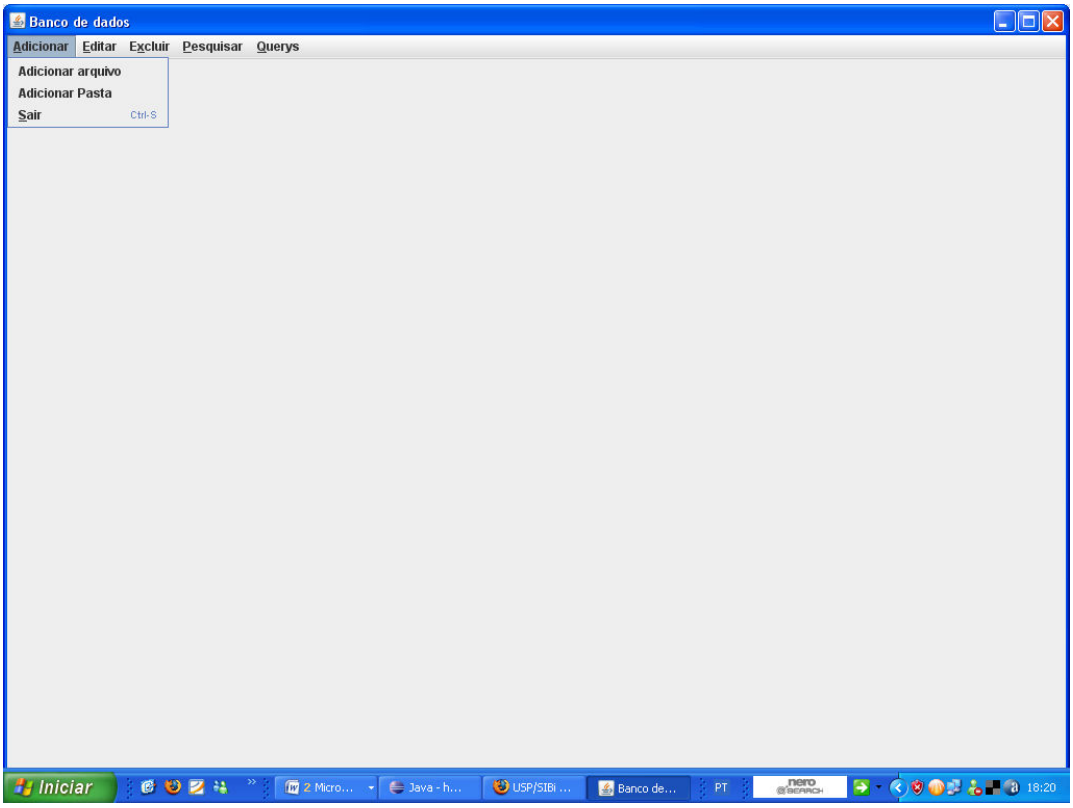

Figura 12– Menu adicionar

O menu editar permite a seleção de um arquivo para ser modificado, apenas usuários com tal privilegio serão capazes de concretizar a operação, uma vez que o próprio SGBD é quem gerencia o sistema de privilégios.

A combinação de teclas para acessar esse menu é ALT+E.

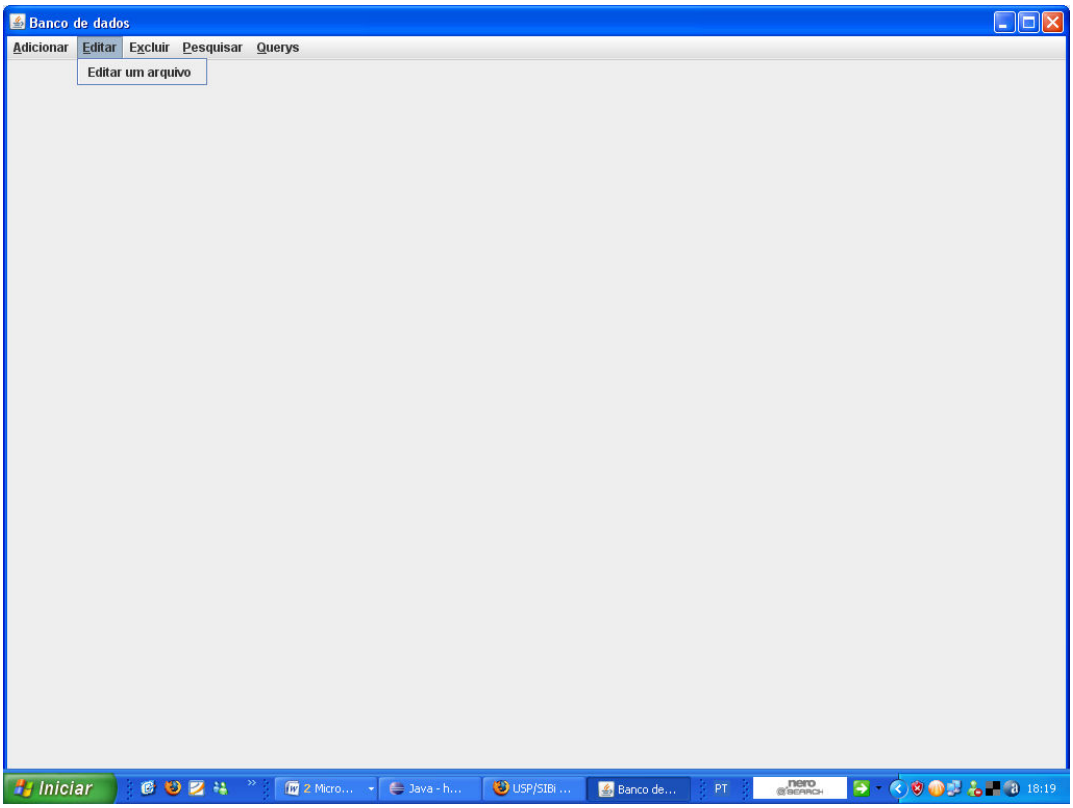

Figura 13 – Menu editar

O menu excluir permite excluir um arquivo.

Assim como nas outras sessões apenas usuários com este tipo de permissão conseguem concretizar este tipo de operação.

O atalho no teclado para este menu é ALT+X.

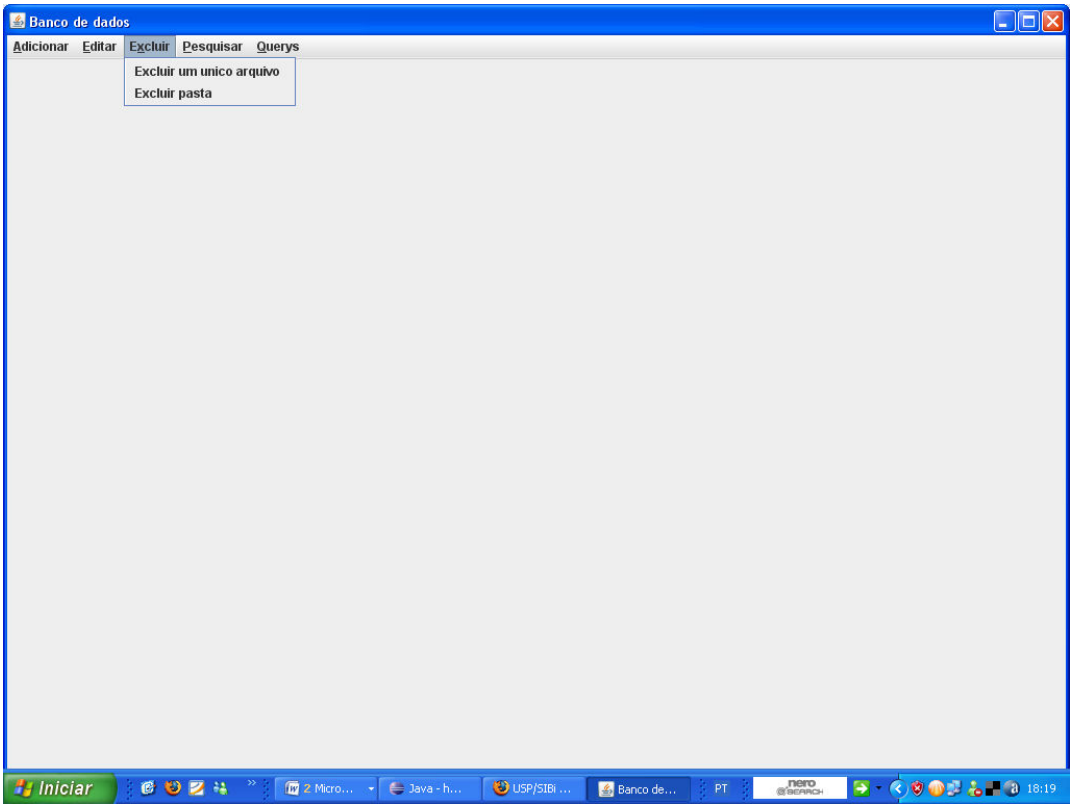

Figura 14 – Menu Excluir

O menu pesquisar permite a qualquer usuário realizar consultas na base de dados, e obter como resultado as amostras que atendam aos requisitos das consultas. Então o usuário tem a opção de extrair uma cópia da mesma para o banco e ou ouvi-lá. Esta funcionalidade a principio é permitida para todos os usuários.

O atalho para este menu é ALT+P.

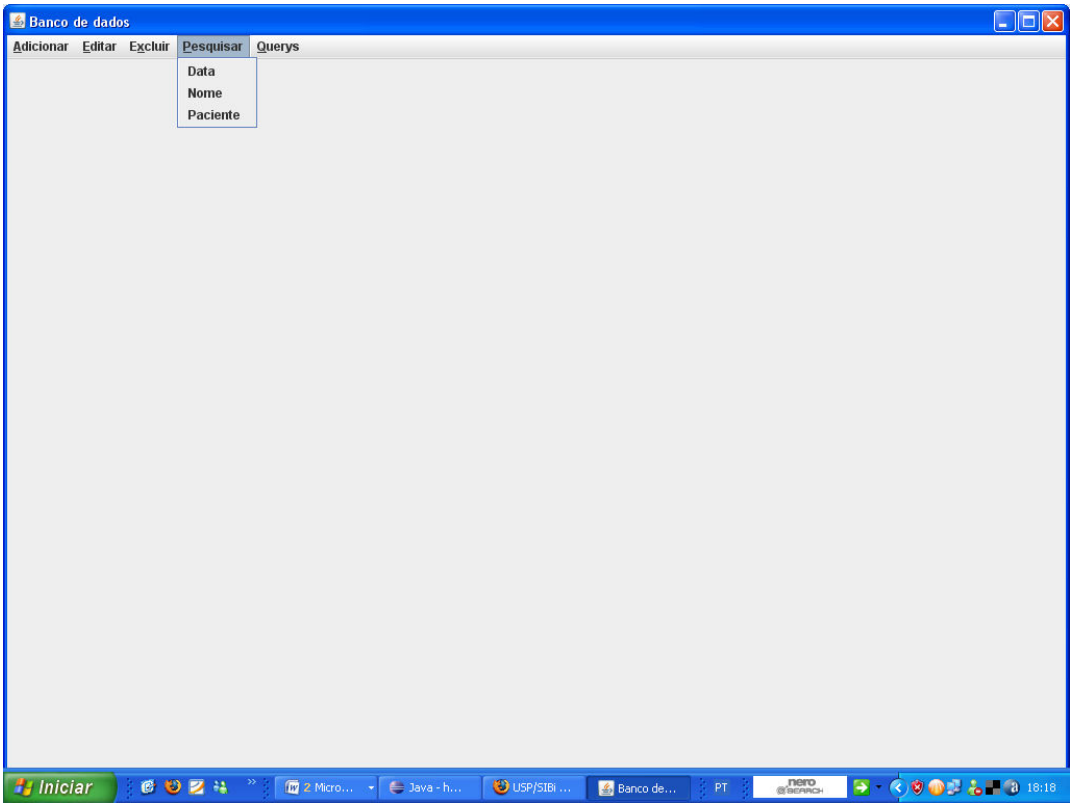

Figura 15 – Menu Pesquisar

Este menu apresenta consultas já pré estabelecidas e escritas em sql, bastando ao usuário escolher qual tipo de busca ele irá realizar e entrar com os dados referentes ao tipo de busca, podendo eles ser referentes por enquanto a: nome da amostra, data da amostra e para o administrador do sistema, através do nome do paciente. Mas também serão adicionadas buscas pelo diagnóstico das amostras.

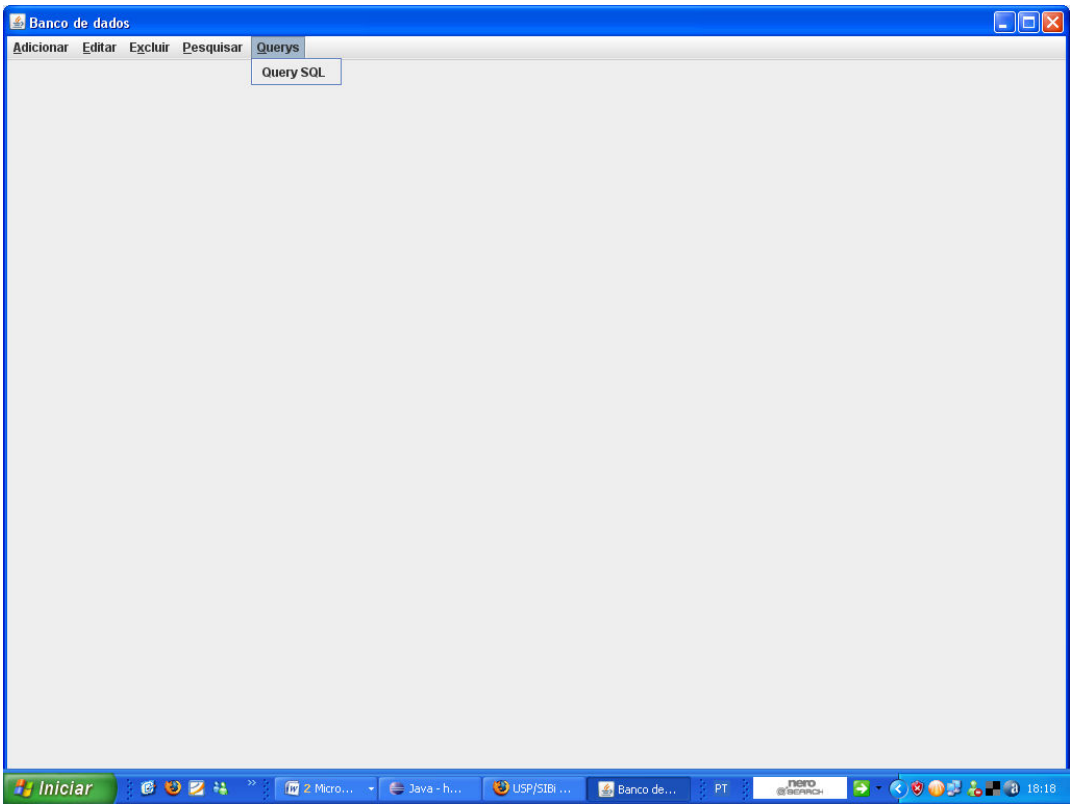

Figura 16 – Menu Querys

O menu querys exibido acima permite que um usuário envie consultas, ou gerencie o banco diretamente em SQL. Para tanto é necessário que o usuário possua um conhecimento prévio da linguagem SQL.

O atalho para este menu é ALT+Q.

5.2 Escolhendo um arquivo

Em praticamente todas as funcionalidades descritas no aplicativo, é pedido ao usuário para escolher um arquivo ou diretório, o qual sofreria as ações designadas. Para facilitar esta operação ao usuário, e evitar que o mesmo tivesse de entrar com o endereço completo do arquivo ou diretório, foi utilizada a classe JFileChooser, presente no componente javax.swing.JComponent, fornecido pela própria linguagem JAVA, o qual gera uma janela para o usuário navegar nos diretórios do seu sistema, facilitando a tarefa de encontrar o arquivo ou pasta desejada.

Este componente se encarrega então de facilitar a busca no sistema pelo arquivo, e ainda elimina a necessidade de que o usuário precise entrar com o endereço do mesmo, apenas tendo o trabalho de clicar no arquivo desejado. Como mostra a figura abaixo.

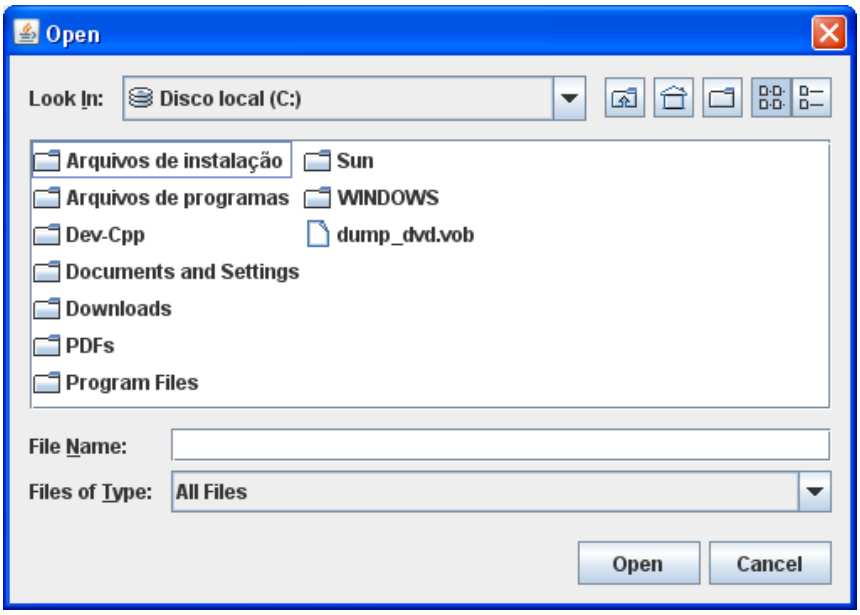

Figura 17 – Escolhendo um Arquivo

Praticamente todos os softwares que trabalham com arquivos que precisam ser inseridos ou salvos em algum diretório, trabalham com uma interface semelhante a essa. Sendo corriqueira a sua utilização pela maior parte dos usuários, não necessitando, portanto de maiores explicações.

5.3 Login do usuário

Cada usuário precisa logar com um user name e senha para poder utilizar o banco de dados, sendo que o próprio banco de dados possui registrado quais os privilégios cada usuário possui, determinando assim qual usuário pode ter acesso a modificações nos dados ou acesso ao nome dos pacientes, mas todos a princípio possuem acesso para realizar consultas simples aos campos que não são restritos, como por exemplo, o caso dos nomes dos pacientes que se tratam de informações sigilosas.

Como mostra a imagem abaixo:

Figura 18 - Login

Caso o usuário não entre com o login correto, uma mensagem de erro avisando que a senha ou user name estão incorretos é retornada, a mensagem de erro é mostrada abaixo:

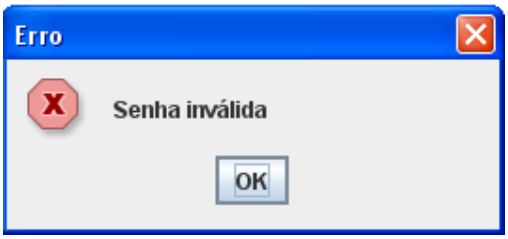

Figura 19 – Senha inválida

5.4 Armazenando os arquivos na base de dados

Para armazenar os arquivos de voz, é recomendado que os mesmos se encontrem gravados em um formato que seja lossless, ou seja, não aplique nenhum tipo de compactação no arquivo, o importante neste escopo é a integridade das informações relacionadas às vozes dos pacientes, um exemplo de formato que permite este tipo de informação se trata do .wav. No entanto vale ressaltar que a aplicação não impede que arquivos gravados em outros formatos sejam inseridos, com o objetivo de não restringir as opções do usuário, ou possíveis evoluções que possam ocorrer neste tipo de tecnologia.

A inserção é feita através da seguinte query:

insert into amostra values (checksum, nome_amostra, lo_import('caminho_da_amostra'), nome_pac, data);

Esse processo é realizado pelo file chooser, já descrito acima.

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

Todas as outras variáveis também são obtidas sem a interferência do usuário, sendo que a única tarefa para o usuário é selecionar o arquivo (ou pasta de arquivos) a ser inserido através do file chooser.

O nome do paciente é o nome do diretório no qual suas amostra se encontram armazenadas. O programa então insere no campo nome do paciente, o nome do diretório no qual a amostra está registrada.

Na ocasião de ocorrer uma inserção de 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. Se por acaso existirem sub pastas, a função a analisa do mesmo jeito.

A verificação para prevenir a inserção de arquivos redundantes no banco é realizada pela própria base, que checa a chave primária de cada amostra, o checksum então é checado, caso existam valores iguais significa que o arquivo já foi inserido na base e, portanto ele não poderá ser inserido de novo. Caso ocorra o fato de que as amostras possuam nomes iguais, mas se tratem de arquivos distintos pois possuem checksums diferentes, uma mensagem é enviada, pedindo para que a amostra seja renomeada :

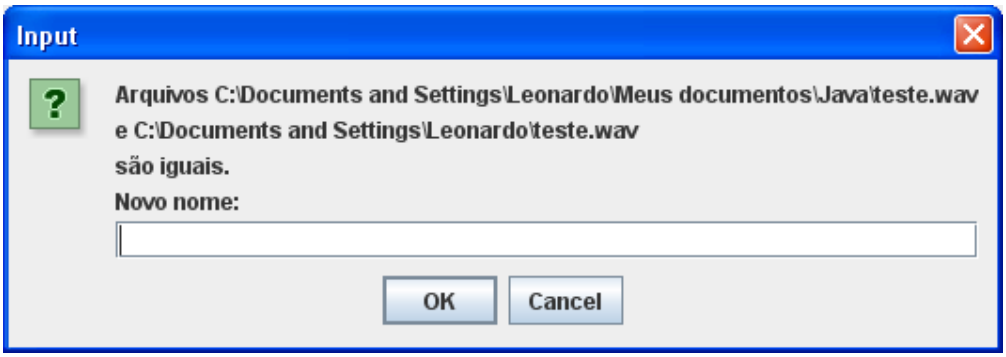

Figura 20 – Renomeando arquivos

5.5 Análise

Até o presente momento é possível dizer que a implementação do banco de dados proporcionará a organização e estruturação das amostras de vozes. A eliminação de amostras redundantes também será obtida. Além de facilitar e padronizar um sistema de busca, facilitando assim o acesso a amostras de vozes com características semelhantes.

Sendo portanto uma aquisição importante para o laboratório, uma vez que se trata de uma ferramenta que pode ser considerada a evolução natural para o armazenamento e gerenciamento do acervo de amostra de vozes.

5.6 Dificuldades encontradas

Um grupo maior de desenvolvedores, ou um prazo maior, com toda a certeza permitiria a implementação de funções mais complexas para o aplicativo. Proporcionando soluções melhores e mais adequadas para os problemas expostos e abordados.

Como por exemplo seria apresentada uma solução para o parser mencionado como possível solução para o problema de nomes e arquivos não elegíveis, lembrando que a implementação de um parser, para solucionar este tipo de problema, já poderia ser considerado um outro projeto, dado a necessidade de que se teria de reconhecer padrões entre os nomes, o que o tornaria capaz de agrupar amostras de um mesmo paciente, mesmo com elas apresentando o nome do arquivo de amostra com o nome do paciente escrito de forma distinta em cada amostra.

Também poderia ser apresentado algum tipo de solução para evitar que arquivo que não fossem amostras seja inserido nas bases. Sendo que a solução apresentada no trabalho se mostrou não satisfatória e limitante da flexibilidade do projeto.

Outro fator limitante é o fato de que não existe uma dedicação exclusiva por parte dos alunos para a implementação e desenvolvimento do projeto, uma vez que eles ainda estão cursando os últimos semestres da graduação, e precisam também cumprir a disciplina de estágio supervisionado.

6. Conclusões

O desenvolvimento e a participação no projeto foi de significativa importância para o processo de aprendizado e crescimento pessoal do aluno. Uma vez que representou uma experiência nova para o mesmo.

Apresentando uma situação mais próxima da realidade de um profissional da área, experiência a qual é bem escassa durante o curso. O projeto permitiu que o estudante vivencia-se a experiência de ter que lidar com um problema diretamente, tendo que criar soluções a partir de seu conhecimento, e de pesquisas e estudos sobre assuntos do qual era apenas familiarizado. Foi possível então vivenciar o famoso "correr atrás", fato importante para um futuro profissional da área.

O aluno também foi exposto a um novo grau de responsabilidade, tendo de lidar com a confiança de seu orientador e respeitar o prazo de entrega para o projeto, o que envolve todo um processo de planejamento e organização pessoal do mesmo.

O projeto, que aliás trata-se de um trabalho de um porte muito maior que os rotineiros trabalhos de graduação, materializou boa parte da teoria vista sobre como deve ser abordado e planejado o processo de desenvolvimento e implementação de um sistema um pouco maior, que representa relativamente bem problemas encontrados comumente no mundo real.

Por fim é valido comentar que o projeto veio de encontro com uma necessidade existente no laboratório, e que envolveu muito da capacidade de interpretação do problema por parte do aluno, de modo que ele fosse capaz de entender a necessidades a qual o seu projeto precisava atender e suprir, solucionando o problema de acordo com a vontade do mesmo, ou seja estimulando o diálogo entre desenvolvedor e usuário, característica fundamental para qualquer profissional, que significa que ele é capaz de entender e executar as funções as quais ele está sendo chamado para solucionar.

A disciplina de projeto, pode então ser considerada uma experiência válida e necessária durante o curso de graduação para engenheiros.

7 Referências Bibliográficas

http://pt.wikipedia.org/wiki/Md5

http://java.sun.com/docs/books/tutorial/jdbc/overview/index.html

http://www.sqlsummit.com/People/JMelton.htm

http://java.sun.com/javase/6/docs/technotes/guides/jdbc

C. Date, An Introduction to Database Systems, Eighth Edition, Addison Wesley, 2003.

J. Gray, A. Reuter, Transaction Processing: Concepts and Techniques, 1st edition, Morgan Kaufmann Publishers, 1992.

David M. Kroenke, Database Processing: Fundamentals, Design, and Implementation (1997), Prentice-Hall, Inc., pages 130-144

S. Lightstone, T. Teorey, T. Nadeau, “Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more”, Morgan Kaufmann Press, 2007. ISBN: 0123693896

T. Teorey, S. Lightstone, T. Nadeau, “Database Modeling & Design: Logical Design, 4th edition”, Morgan Kaufmann Press, 2005. ISBN: 0-12-685352-5

Ostrovsky, L. A. and Potapov, A. S. (1999). *Modulated Waves, Theory and Applications*. Baltimore: The Johns Hopkins University Press.

Jepson, Brian -Java database programming

Mello, Rodrigo – Aprendendo Java 2.

Momjian, Bruce – PostgreSQL Introduction and Concepts

Elmasri, RFundamentals of database systems

Gorman, M MDatabase management systems

Date, C. J.Introdução a sistemas de bancos de dados

Ramakrishnan, Raghu; Gehrke, Johannes - Database Management Systems 2Ed

Fowler, Martin - Refactoring: Improving the Design of Existing Code

Douglas, Korry; Douglas, Susan – The Comprehesive guide to building, programming and administering PostgreSQL databases, 2nd Edition