FACULDADE GOVERNADOR OZANAM COELHO

BUSCA FONÉTICA EM BANCO DE DADOS

estudo da busca fonética para adaptação no mercado brasileiro

VINÍCIUS DE LUCENA BONOTO

UBÁ
MINAS GERAIS
2011

VINÍCIUS DE LUCENA BONOTO

BUSCA FONÉTICA EM BANCO DE DADOS:
estudo da busca fonética para adaptação em mercados brasileiros

Monografia apresentada como parte das exigências para a obtenção do título de Bacharel em Ciência da Computação da FAGOC - Faculdade Governador Ozanam Coelho,.

Orientador: Prof. Saulo Cunha Campos.

UBÁ

MINAS GERAIS

2011

VINÍCIUS DE LUCENA BONOTO

BUSCA FONÉTICA EM BANCO DE DADOS:
estudo da busca fonética para adaptação em mercados brasileiros

Monografia apresentada e aprovada em .__ de dezembro de 2011.

_____
Waldir Andrade Trevizano
(Examinador)

_____
Marcelo Santos Daibert
(Examinador)

_____
Saulo Cunha Campos
(Orientador)

_____
Waldir Andrade Trevizano
(Coordenador de TCC)

AGRADECIMENTOS

Durante todo o caminho percorrido para conclusão deste trabalho, passei por momentos em que precisei de ajuda para alcançar meus objetivos. Neste momento todas essas pessoas devem ser lembradas com imenso carinho e admiração que as tenho.

Inicialmente, agradeço a Deus por me dar força e saúde para construção de um caminho seguido de fé para o término deste trabalho, onde fui amparado a cada dia em que realizei o projeto.

Agradeço a minha mãe Mariluce de Lucena Bonoto, que com muito empenho, alegria, amor e dedicação sempre esteve comigo nos momentos felizes e difíceis que passei durante este longo caminho. Minha mãe que também é pai, e que a cada dia percebo o tanto que é especial para mim.

Agradeço ao meu pai Luíz Bonoto Filho, que esteve ao meu lado até meus 12 anos de idade, mas que hoje está presente em espírito para me trazer a alegria que sempre foi presente em sua vida. Alegria no qual me inspirou a dedicar este trabalho a ele.

Agradeço aos meus irmãos Carla de Lucena Bonoto Zanini e Rodrigo de Lucena Bonoto, que foram companheiros nos momentos que mais precisei. Ao meu cunhado Paulo Zanini, que com uma grande amizade, esteve sempre presente nos momentos mais felizes da minha. Juntamente com o Lauro Teles, pessoa pela qual sempre me passou belos exemplos.

Agradeço a minha namorada Jéssica Ciotti, pelo amor concedido durante este processo e por ser compreensiva nos momentos em que mais precisei para alcançar meus objetivos.

Agradeço aos professores e colegas de sala da FAGOC, em especial ao Saulo Cunha Campos que além de professor tornou-se amigo e que todo conhecimento repassado foi de grande valia para meu crescimento profissional, e ao amigo de sala Jacinto Junior, que juntos conseguimos concluir nossos objetivos.

Por fim, agradeço aos meus colegas de trabalho da Tek-System Informática, em especial ao José Ricardo Varella, Denis Pereira Raymundo, Antonio Gomes e Diogo Sudré. Pessoas que foram compreensíveis e estiveram presentes nos momentos que mais precisei durante esta caminhada.

RESUMO

Observando a diversidade da Língua Portuguesa e a grande variedade de palavras que podem ser lidas e escritas de formas diferentes devido ao som das letras nas palavras, é comum aplicações de banco de dados conterem palavras escritas de formas diferentes, embora foneticamente iguais. Apesar de já existirem estratégias para solução do problema para a Língua Inglesa, empresas voltadas para o mercado brasileiro não possuem soluções nativas eficientes nos diversos SGBD livres disponíveis no mercado. Este trabalho tem o intuito de promover o estudo da busca fonética, a implementação e o aperfeiçoamento de estratégias para a Língua Portuguesa nos SGBD Firebird, MYSQL e Postgree. O uso desse recurso contribui para uma crescente melhora em desempenho nas pesquisas realizadas e diminuição nas redundâncias dos bancos de dados.

Palavras-chave: Busca Fonética, Soundex, Metaphone, Double-Metaphone, Banco de Dados.

SUMÁRIO

LISTA DE FIGURAS vii
LISTA DE QUADROS viii
LISTA DE CÓDIGO-FONTE x
LISTA DE SIGLAS xii
1 INTRODUÇÃO 1
1.1 O PROBLEMA E SUA IMPORTÂNCIA 3
1.2 OBJETIVO 5
1.2.1 Objetivo Geral 5
1.2.2 Objetivos específicos 6
1.3 Organização do Trabalho 6
2 REFERENCIAL TEÓRICO 7
2.1 FONÉTICA DA LÍNGUA PORTUGUESA 7
2.2 BUSCA FONÉTICA 11
2.3 Soundex 112
2.4 Metaphone 15
2.5 Double Metaphone 17
2.6 NYSIIS 19
2.7 Comparativo Entre os Algoritmos de Busca Fonética 20
3 TRABALHOS RELACIONADOS 22
4 METODOLOGIA 24
5 ESTUDO DOS ALGORITMOS DE BUSCA FONÉTICA EXISTENTES EM SGBD LIVRES 26
5.1 Busca Fonética em SGBD Livres 26
5.1.1 MySQL 26
5.1.2 POSTEGRESQL 27
5.1.3 Firebird 28
5.2 Aplicaçao dos Algoritmos de Busca Fonética já Existentes para Língua Portuguesa 28
5.2.1 Soundex 28
5.2.2 Metaphone 29

5.3 Soluçoes Alternativas para Busca Fonética no idioma Português em SGBD Livres 30
5.3.1 SOLUÇAO 1 31
5.3.2 SOLUÇÃO 2 38
6 BUSCASONORA: PROPOSTA DE SOLUÇÃO PARA A BUSCA FONETICA NA LÍNGUA PORTUGUESA 40
6.1 PRIMEIRA FASE – CORREÇÃO DE FALHAS DO ALGORITMO 42
6.2 SEGUNDA FASE – TRATAMENTO PARA LETRAS S, X E Z 47
6.3 TERCEIRA FASE – DESENVOLVIMENTO DA INTERFACE 58
7 ESTUDO DE CASO 60
8 CONSIDERAÇÕES FINAIS 63
REFERÊNCIAS BIBLIOGRÁFICAS 64
APÊNDICE A: PREPARAÇÃO DO AMBIENTE PARA REALIZAÇÃO DA BUSCA FONÉTICA NO BANCO DE DADOS FIREBIRD 67
APÊNDICE B: INSTALAÇÃO DO FIREBIRD 68
APÊNDICE C: CÓDIGO FONTE ORIGINAL PUBLICADO NA REVISTA CLUBE DELPHI EM 2006 72
APÊNDICE D: PROCEDIMENTO (PROCEDURE) CRIADO NO FIREBIRD PARA BUSCA FONÉTICA 76
APÊNDICE E: PROCEDIMENTO (PROCEDURE) CRIADO NO FIREBIRD PARA RETIRAR AS PALAVRAS (E, DA, DAS, DE, DI, DO e DOS) 83
APÊNDICE F: PROCEDIMENTO (PROCEDURE) CRIADO NO FIREBIRD PARA ATRIBUIR NÚMEROS A PALAVRAS QUE CONTÉM S OU Z 84
APÊNDICE G: CÓDIGO FONTE DE CONSULTA DO SISTEMA QUE REALIZA A BUSCA FONÉTICA 86

LISTA DE FIGURAS

Figura 1: Gráfico informando aumento nos pedidos online que implementaram a busca fonética 5

Figura 2: Países falantes da Língua Portuguesa..... 11

Figura 3: Exemplo de novo fonema para a letra S de acordo com a palavra ‘vez’... . 46

Figura 4: Grafemas da letra Z. 49

Figura 5: Grafemas da letra X. 49

Figura 6: Tela para cadastro no sistema de busca fonética. 58

Figura 7: Tela para consultar nomes inseridos no sistema de busca fonética. 59

Figura 8: Visualização de um cadastro no sistema BuscaSonora. ..... .61

Figura 9:. Consulta no sistema BuscaSonora..... 62

Figura 10: Tela inicial da instalação do Firebird para seleção do idioma 69

Figura 11: Tela para configurar em que local deseja que a pasta do firebird seja criada 70

Figura 12: Tela para configurar instalação como servidor ou como cliente..... 71

Figura 13: Tela para configurar recursos adicionais oferecidos pelo Firebird 72

LISTA DE QUADROS

Quadro 1: Fonemas vocálicos e fonemas consonantais 8
Quadro 2: Regra de atribuição de números para consoantes no algoritmo Soundex 13
Quadro 3: Rastreio da palavra “Double” no algoritmo Soundex 14
Quadro 4: Rastreio da palavra Quadratically no algoritmo Soundex 14
Quadro 5: Rastreio do algoritmo Metaphone 16
Quadro 6: Códigos fonéticos e acordo com o algoritmo NYSIIS 20
Quadro 7: Resultado de palavras utilizando as funções Soundex, NYSIIS, Metaphone, Double-Metaphone 21
Quadro 8: Exemplos de utilização do Soundex no SGBD MySQL 27
Quadro 9: Exemplos de utilização do Soundex, Metaphone e Double Metaphone no SGBD PostGreSQL 27
Quadro 10 – Rastreio da palavra Casa e Caju no algoritmo Soundex 28
Quadro 11 – Exemplo da palavra Walter e Valter no algoritmo Soundex. 28
Quadro 12: Exemplo da palavra Numbers sendo tratada no Metaphone 29
Quadro 13: Exemplo da palavra Knowledge sendo tratada no Metaphone 29
Quadro 14: Exemplo dos nomes Xirlei e Shirlei sendo tratada no Metaphone 30
Quadro 15: Exemplo dos nomes Fatima e Fathima sendo tratada no Metaphone..... 30
Quadro 16: Dados inseridos após a execução do insert de acordo com a listagem de Código Fonte 8 33
Quadro 17: Resultado fonético para os nomes Karla e Carla 34
Quadro 18: Resultado fonético para os nomes Wagner e Vagner 35
Quadro 19: Resultado fonético para os nomes Fatima e Fathima 35
Quadro 20: Resultado fonético para o nome Acelino 35
Quadro 21: Demonstração de resultado para os nomes Acelino, Asselino e Ascelino36
Quadro 22: Resultado fonético para o nome Niltom 36
Quadro 23: Resultado fonético para o nome Nilton 36

Quadro 24: Resultado fonético para o nome Gisele..... 37
Quadro 25: Resultado fonético para o nome Jisele..... 37
Quadro 26: Resultado fonético para o nome Ualace..... 37
Quadro 27: Resultado fonético para o nome Walace..... 37
Quadro 28: Estratégia do algoritmo Soundex para busca fonética 38
Quadro 29: Regras para geração do resultado de acordo com a DLL MetaUDF..... 39
Quadro 30: Resultado fonético para o nome Carla 39
Quadro 31: Resultado fonético para o nome Karla 39
Quadro 32: Resultado fonético para o nome Chico..... 40
Quadro 33: Resultado fonético para o nome Xico..... 40
Quadro 34: Resultado de busca com o nome Luiz no algoritmo de BUBLITZ, 2006 . 43
Quadro 35: Resultado de busca com o nome Luis no algoritmo de BUBLITZ, 2006 . 43
Quadro 36: Atribuição de Caracteres especiais para os fonemas X, Z e S..... 50
Quadro 37: Resultados de todos os possíveis grafemas para a palavra EXAME..... 50
Quadro 38: Demonstração do novo parâmetro TEM_S_OU_Z criado..... 55
Quadro 39: Atribuição de números 1 ou 0 no parâmetro TEM_S_OU_Z criado..... 55
Quadro 40: Lista de nomes cadastros no sistema de busca fonética proposto no trabalho..... 58

LISTA DE CÓDIGO-FONTE

Listagem de Código-Fonte 1: Pseudocódigo do algoritmo Soundex.....13

Listagem de Código-Fonte 2: Pseudocódigo do Procedimento Metaphone.....15

Listagem de Código-Fonte 3: Código para o algoritmo Double Metaphone tratando a letra C.....17

Listagem de Código-Fonte 4 – Procedimento do pseudocódigo NYSIIS.....19

Listagem de Código-Fonte 5: Procedimento da busca fonética para o Firebird.....31

Listagem de Código-Fonte 6– SQL para criação do banco de dados e criação da tabela CLIENTE.....32

Listagem de Código-Fonte 7– SQL para vinculo da UDF no software IBexpert.....32

Listagem de Código-Fonte 8– SQL para criação do gatilho (trigger).....33

Listagem de Código-Fonte 9 – SQL para inserção dos dados na tabela Cliente.....33

Listagem de Código-Fonte 10 – SQL com seleção para o nome Karla.....34

Listagem de Código-Fonte 11 – SQL com seleção para o nome Wagner.....34

Listagem de Código-Fonte 12 – SQL com seleção para o nome Fatima.....35

Listagem de Código-Fonte 13 – SQL com seleção para o nome Acelino.....35

Listagem de Código-Fonte 14 – SQL com seleção para o nome Niltom.....36

Listagem de Código-Fonte 15 – SQL com seleção para o nome Nilton.....36

Listagem de Código-Fonte 16 – SQL com seleção para o nome Gisele.....37

Listagem de Código-Fonte 17 – SQL com seleção para o nome Jisele.....37

Listagem de Código-Fonte 18 – SQL com seleção para o nome Ualace.....37

Listagem de Código-Fonte 19 – SQL com seleção para o nome Walace.....37

Listagem de Código-Fonte 20 – SQL com seleção para o nome Carla.....39

Listagem de Código-Fonte 21 – SQL com seleção para o nome Karla.....39

Listagem de Código-Fonte 22 – SQL com seleção para o nome Xico.....40

Listagem de Código-Fonte 23 – SQL com seleção para o nome Chico.....40

Listagem de Código-Fonte 24 – Algoritmo incluído para tratamento da letra N e M no final da palavra.....42

Listagem de Código-Fonte 25 – Função criada para tratar a letra Z no final da palavra.....42

Listagem de Código-Fonte 26 – Algoritmo incluído para tratamento da letra S no final da palavra.....43

Listagem de Código-Fonte 27 – Algoritmo incluído para tratamento da letra I depois da letra G.....43

Listagem de Código-Fonte 28 – Algoritmo incluído para tratamento das vogais A, E, I, O, U antes da letra U depois da letra G.....44

Listagem de Código-Fonte 29 – SQL Algoritmo incluído para tratamento da letra “X” no final da palavra e a letra “C” antes da letra S”.....44

Listagem de Código-Fonte 30 – Algoritmo incluído para tratamento da letra “C”.....45

Listagem de Código-Fonte 31 – Algoritmo incluído para tratamento para os grafemas da letra “S”.....47

Listagem de Código-Fonte 32 – Algoritmo que Realiza as Variações de Acordo com os Grafemas das Palavras.....50

Listagem de Código-Fonte 33 – Algoritmo que Realiza busca fonética especial para palavras que contém letras S e Z.....53

Listagem de Código-Fonte 34 – Procedimento chamado CONTEMSOUZ.....56

LISTA DE SIGLAS

SPED - Sistema Pública de Escrituração Digital

XML - Linguagem extensível de marcação

INPI - Instituto nacional da propriedade industrial

SGBD - Sistema gerenciador de Banco de Dados

DLL - Biblioteca de ligação dinâmica

UDF - Funções definidas pelo usuário

PLSQL - Extensão da linguagem SQL

1 INTRODUÇÃO

Atualmente, o uso de software em empresas tornou-se indispensável, levando em consideração novas exigências contábeis, tais como: Sped (Sistema Público de Escrituração Digital) Fiscal, Pis/Cofins e Contábil. Trata-se de informações tributárias obrigatórias para envio à Receita Federal. O designo do projeto Sped é modificar a forma como são realizadas as escriturações contábeis e análises das notas fiscais eletrônicas emitidas pelas empresas, possibilitando um controle mais especifico das diversas administrações tributárias, o que se constitui em um avanço tecnológico (MAIA; OLIVEIRA, 2007).

Existem outros projetos que entraram em vigor em todos os estados do Brasil. É possível citar como exemplo a nota fiscal eletrônica e o conhecimento de transporte eletrônico, uma realidade no país que, através de sua obrigatoriedade, tornou prioridade o uso de software no país, pois todas as informações são transmitidas por um arquivo de extensão XML[1] (Linguagem extensível de marcação).

Obter essas informações arquivadas em banco de dados torna-se um facilitador tanto para geração de informações à Receita Federal quanto para gerenciamento e análises administrativas, possibilitando que esses procedimentos adquiram mais agilidade no controle das informações que atualmente se tornaram indispensáveis para as empresas.

O uso de cadastros é realizado com grande frequência por diversas pessoas da empresa. Esse processo, por ser manual, está sujeito a possíveis problemas como: erros por falha na interpretação das informações, falhas na digitação de

[1] Com o XML as informações são trafegadas com mais flexibilidade, resultando em um arquivo leve e de fácil compartilhamento eletrônico. Tornando uma opção aceitável para publicação e compartilhamento em grande quantidade (W3, 2011).

palavras, devido à riqueza ortográfica da Língua Portuguesa e, consequentemente, dificuldade de recuperar informações já cadastradas.

A busca fonética é um mecanismo que visa efetuar consultas no banco de dados pelo som das letras das palavras e não pela forma como elas são escritas. Assim, palavras que possuem escritas diferentes, porém possuem um mesmo som, são recuperadas como resultado de busca de uma determinada palavra específica.

As palavras “Chico” e “Xico” são exemplos de palavras que possuem escritas diferentes, mas possuem sons iguais. Atualmente, uma busca simples pela palavra “Chico” em um campo de um banco de dados não traria como resultado a palavra “Xico”. Porém, o intuito da busca fonética é que ambas sejam retornadas, pois possuem o mesmo som.

Atualmente esse problema já foi resolvido para o idioma Inglês através dos algoritmos Soundex, Metaphone e Double Metaphone, distribuídos nativamente na maioria dos SGDB. Porém, para o idioma português, esses algoritmos não são aplicados devido à diferença em questões gramaticais e ortográficas.

Além de agilidade nas consultas ao banco de dados, a busca fonética traz confiabilidade para o software, exercendo a função de um mecanismo de busca importante para o sistema. Com base nesses benefícios, o atendimento ao cliente se tornará mais flexível, diminuindo o risco de ocorrer cadastros duplicados.

O processo de busca através do som pode ser utilizado tanto para sistemas comerciais usados em aplicações *desktop* quanto para sites e-commerce, devido ao grande crescimento das lojas virtuais no Brasil. Portanto, o uso da busca fonética torna-se indispensável, levando em consideração que cerca de 17,6 milhões de consumidores compraram pela internet em 2009, ou seja, 26% dos internautas brasileiros (CURSODEECOMMERCE, 2011).

1.1 O problema e sua importância

Com o acesso cada vez mais popularizado à computação entre empresas, tanto de pequeno, médio ou grande porte, ter um gerenciador de banco de dados, que possibilite guardar as informações para análises comerciais, financeiras e administrativas tornou-se importante. Devido a essa importância, os cadastros são realizados com grande frequência, gerando como problema possíveis erros por falha na interpretação das informações ou por falha na digitação de palavras, levando em consideração a dificuldade em realizar uma escrita correta que a Língua Portuguesa traz no cotidiano.

Devido às crescentes exigências contábeis e empresariais para melhor administração dos negócios das empresas, há um aumento crescente do uso de softwares; consequentemente, ocorre também o aumento de bases de dados para controle das informações. A redundância de dados é um problema que pode ocorrer durante todo o processo de utilização do software, ocasionando perda de performance/desempenho.

A repetição de dados desnecessários (redundância) contribui para aumento do tamanho do banco de dados da empresa. É sabido que, quanto maior o volume de dados armazenados, maiores serão os tempos de busca e atualização dos dados. Sendo assim, a redundância gera prejuízos no funcionamento do banco de dados e, consequentemente, das aplicações.

É possível citar como exemplo o seguinte processo de venda em uma empresa: uma pessoa, ao realizar um compra via telefone, diz que seu nome é “Vanessa Assunção”, e o vendedor, para realizar a venda, efetua o cadastro com o nome tal qual ele escuta da pessoa, gerando o código de cliente 100. Porém, em outro momento, a mesma cliente se dirige à empresa para realizar uma nova compra no próprio estabelecimento. Ao chegar ao balcão, um vendedor pede a sua identidade para efetuar a busca do cadastro e verifica que o nome da pessoa é “Wanessa Assunssão”. Não encontrando esse nome, ele efetua outro cadastro, com código 150, que, na verdade, corresponde à mesma pessoa.

Esse exemplo mostra que uma mesma pessoa foi cadastrada em dois códigos diferentes (100 e 150) devido a problemas de leitura/escrita de palavras que possuem a mesma pronúncia. Trata-se de um problema para empresa, que não conseguirá, a partir desse momento, efetuar análises de crédito corretas, nem

cobranças, relatórios gerenciais de vendas, entre outros, para esse cliente. Isso porque, na visão do banco de dados, trata-se de dois clientes distintos (um com código 100 e outro 150),.

A ocorrência de situações como a descrita acima é comum nas empresas, devido à grande dificuldade em implementar as regras fonéticas do português e ao avanço que pode ocorrer nas empresas em termos de agilidade nos processos e conferência de dados nos bancos de dados. Neste trabalho é citado um estudo da fonética da Língua Portuguesa, com exemplos de palavras foneticamente parecidas, porém escritas de modo diferente. No projeto também é feita uma análise dos algoritmos já utilizados para solução do problema para a Língua Inglesa, comparando a eficiência dos resultados com exemplos tanto em inglês quanto em português.

Ter um banco de dados livre de duplicidades cadastrais ou até mesmo de dados que não serão usados para futuras análises é um desafio para seus administradores. Devido a essa dificuldade, a busca fonética ajuda – através de estratégias que analisam o fonema da palavra e não a forma como ela é escrita – para obtenção de resultados confiáveis.

O tema ‘busca fonética em banco de dados’ ainda não possui uma vasta publicação sobre o assunto, tanto para o idioma português quanto para o idioma inglês. Esse é um fator que dificulta a implementação de soluções sobre o assunto, o que aumenta a sua importância em mercados brasileiros, uma vez que o seu desenvolvimento acarretaria um melhor desempenho para empresas desenvolvedoras de softwares e gerenciadores de banco de dados. Com a implementação dessa função, o índice de melhoria e economia de tempo se tornaria um diferencial para as demais empresas.

Foi possível a obtenção de poucos exemplos atuais de utilização da pesquisa fonética em empresas brasileiras, nas quais o uso se limitou a aplicações *web*, fugindo das aplicações *desktop*. É possível citar o uso da busca fonética nos projetos do *website Help* Saúde, que implementa um inédito aperfeiçoamento em sites de buscas de remédios (HELPSAUDE, 2011). Considerando a importância de implementações além do território brasileiro, o projeto INPI realizou parceria de bases integradas com o mesmo órgão de Portugal, cujo maior diferencial foi o uso da busca fonética, visando ampliar o propósito e a confiabilidade do exame de marcas INPI (INPI, 2011).

Com a busca fonética, é possível resolver questões de economia de tempo e custo, pois os pedidos são realizados online com maior flexibilidade na consulta e na obtenção imediata de informação/resultado. Após a implementação da busca fonética no projeto do INPI (Instituto Nacional da Propriedade Industrial), o resultado encontrado pelo sistema revelou 85% de semelhança na busca realizada pelo usuário, utilizando busca fonética avançada e simples, em que o aumento de registros online foi crescente (PEREIRA, 2011), como pode ser visualizado na Figura 1.

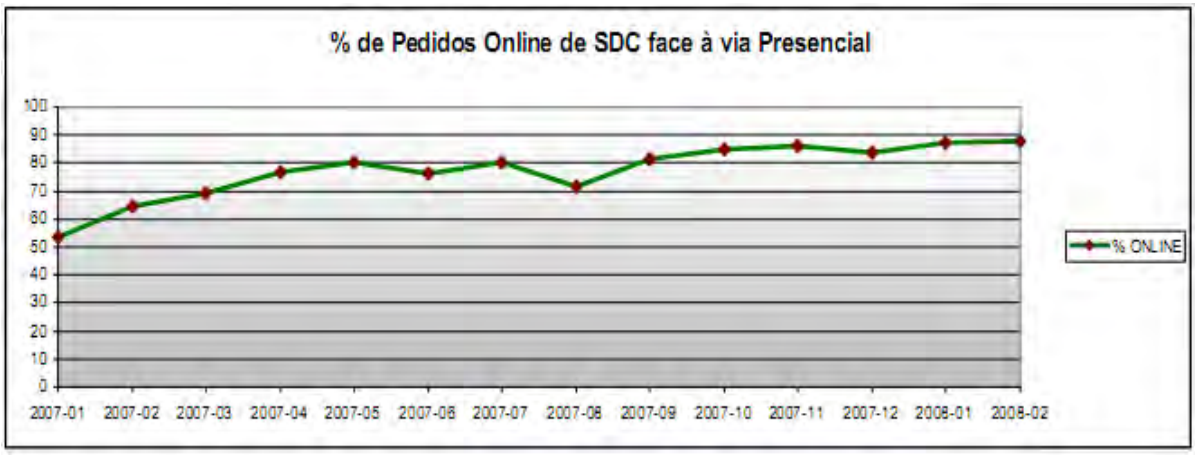

Figura 1 – Aumento nos pedidos online que implementaram a busca fonética

Fonte: Serviços Online INPI – Pesquisas e Registros.

1.2 Objetivos

1.2.1 Objetivo Geral

Propor soluções ou melhorias para a busca fonética no mercado brasileiro com os SGBD Firebird, MYSQL e PostGree.

1.2.2 Objetivos específicos

- Apresentar as regras fonéticas da Língua Portuguesa em contrapartida à fonética do idioma inglês.
- Detalhar e explicar os algoritmos mais utilizados na estratégia de busca fonética: Soundex, Metaphone e Double Metaphone.
- Verificar se existem soluções para a busca fonética no idioma Português nos SGBD Firebird, MySQL e PostGree.
- Com base nos resultados encontrados no item anterior, propor melhorias ou novas soluções para a busca fonética no idioma português.
- Desenvolver estudo de caso para validação do sistema BuscaSonora e comprovação do funcionamento do mesmo.

1.3 Organização do Trabalho

A Língua Portuguesa é um desafio, pois contém diversas regras gramaticais e não há padronização para criação de nomes. O capítulo 2 apresenta a explicação dos problemas que serão encontrados para elaboração do algoritmo fonético para a Língua Portuguesa. São apresentados os algoritmos já criados para a Língua Inglesa, com explicação detalhada do que a função executa, assim como exemplos que definem de forma clara os resultados obtidos através das pesquisas.

O capítulo 3 apresenta os trabalhos relacionados ao tema, explicando resumidamente os projetos e artigos Busca Fonética em Português do Brasil, Protótipo de Um Reconhecimento Fonético Aplicado ao Banco de Dados Oracle e Código Fonético no Firebird. O capítulo 4 apresenta a metodologia utilizada para o desenvolvimento do projeto. E no capitulo 5 é mostrado o desenvolvimento do trabalho e o apêndice trata da instalação do gerenciador de banco de dados Firebird, incluindo o detalhamento de todo o processo de instalação para o sistema operacional Windows junto a demonstração dos códigos utilizados para desenvolvimento da busca fonética.

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados estudos sobre os conceitos da busca fonética e a atribuição desses conceitos para solução computacional de acordo com as línguas Inglesa e Portuguesa. Também é mostrado como problema o não aprimoramento das soluções existentes para mercados brasileiros, no qual existem soluções que não resolvem satisfatoriamente a diversidade da Língua Portuguesa. Além disso, há o intuito de apresentar com clareza as soluções existentes e a importância da busca fonética tanto para empresas quanto para órgãos públicos.

2.1 Fonética da Língua Portuguesa

Para entender com mais clareza todos os algoritmos que serão citados neste trabalho para que, futuramente, seja possível implementá-los em sua aplicação, é necessário ter conhecimento do que é a fonética, seu histórico e quais as definições que são atribuídas a esse recurso.

Fonética é a preocupação de investigação mais antiga da humanidade. Atualmente, para a Língua Portuguesa, as pesquisas estão voltadas para geração de algoritmos que resolvam a grande diversidade que ela traz na sua gramática. Porém, as pesquisas ainda não trouxeram um resultado satisfatório (CAGLIARI, 2006). A fonética se preocupa com a parte significante da palavra e não com todo o seu conteúdo, tendo como exemplo os algoritmos que implementam exatamente o que é a sua definição. Atualmente, o idioma português utiliza 31 fonemas, sendo 12 vocálicos e 19 consonantais (UOL, 2011), apresentados no Quadro 1.

Quadro 1 – Fonemas vocálicos e fonemas consonantais

FONEMAS	REPRESENTAÇÃO	EXEMPLO
Tipo:Oral		
/a/	A	Amor
/e/	E	Beijo
/é/	E ou É	berro, café
/i/	I	Ilha
/o/	O	Olho
/ó/	O ou Ó	cola, mói
/u/	U	Uva
Tipo: Nasal		
/ã/	Ã	Rã
	AM	Campo
	NA	Anta
/ẽ/	EM	Sempre
	EM	Sente
/ĩ/	IM	Sim
	IN	Cinto
/õ/	Õ	Põe
	OM	Pomba
	ON	Ponta
/ũ/	UM	Bumbo
	UM	Mundo
Tipo: Fonema Consonantal		
/p/	P	Pai
/b/	B	Bola
/m/	M	Mãe
/f/	F	Faca
/v/	V	Vaca
/t/	T	Tio
/d/	D	Dado
/n/	N	Nada
/nh/	NH	Nhoque
/l/	L	Lua
/lh/	LH	Lhama
/r/	R	Caro
Tipo: Fonema Consonantal		
/rr/	RR	Carro
	R	rosa, terra
/z/	Z	Zebra

	S	Rosa
	X	Exato
/s/	S	seda, valsa
	SS	Massa
	Ç	Maçã
	C	Cedo
	SC	Descer
	SÇ	Desça
	X	Trouxe
	XC	Excesso
/j/	J	Jeito
	G	Giz
/x/	X	Xis
	CH	Chave
/g/	G	Gude
	GU	Gueixa
/q/	C	Cume
	QU	Queixa

Legenda

- Fonemas Orais
- Fonemas Nasais
- Fonemas Consonantais Simples
- Fonemas Consonantais com maior possibilidade de erros na interpretação

Fonte: UOL (2011), Adaptado pelo autor.

Quanto à representação dos fonemas consonantais, todo o entendimento fica um pouco mais complexo, porque a sua representação entre o som e a forma de escrita pode ser simples ou mais complexa. Por exemplo, a palavra “caminho” se escreve com o dígrafo “NH”, ou seja, duas letras representam apenas um fonema.

No Quadro 1, foram representados todos os fonemas consonantais, havendo uma separação entre os fonemas mais simples de interpretação e os mais complexos, onde uma letra pode exercer o mesmo som, mesmo sendo escrita com outra consoante na palavra. É possível citar como exemplo o fonema consonantal /Z/, que pode ser representado de três maneiras diferentes: na palavra ‘zebra’, usa-se a letra Z; na palavra ‘rosa’, a letra S; e na palavra ‘exato’, a letra X.

Em relação a nomes próprios, a Língua Portuguesa não tem padrões, uma vez que há possibilidade de se criarem nomes com diversas formas de escrita. É possível citar, como exemplo, os nomes Phelipe e Felipe, Karla e Carla, Acelino e Ascelino, Luis e Luiz.

A implementação da busca fonética para a Língua Portuguesa, analisando os algoritmos que resolvem o problema apenas no idioma inglês, é um desafio. Nesse sentido, é necessário saber a diferença de pronúncia entre os dois idiomas.

Segundo Schütz (2008),as diferenças são:

- **CORRELAÇÃO PRONÚNCIA x ORTOGRAFIA:** Pensando na correlação pronúncia x ortografia, é difícil a interpretação oral das palavras em inglês, quanto ao português a pronúncia fica um pouco mais clara e constante. Exemplo: *Table* [teibou]
- **RELAÇÃO VOGAIS x CONSOANTES: O ingl**ês exige um esforço muscular e uma movimentação de seus órgãos, especialmente da língua, significativamente diferentes, quando comparado à fonética do português. O inglês é rico em co*nsoan*tes enqua**nto que o português é abundan**te na ocorrência de vogais e combinações de vogais (ditongos e tritongos). Ex: *December is the twelfth month of the year*. / Eu vou ao Uruguai e o Áureo ao Piauí. / Eu sou europeu.
- **SINALIZAÇÃO FONÉTICA:** O inglês é uma língua mais econômica em sílabas do que o português. O número de palavras monossilábicas é mui*to superior quando comparado ao português*. Ex:

  beer / cer-ve-ja
  book / li-vro
  car / car-ro
  dream / s**o**-nho

- **NÚMERO DE FONEMAS:** Devido à economia no uso de sílabas, o inglês precisa de um número maior de sons vocálicos para diferenciar as inúmeras palavras monossilábicas, enquanto que o português apresenta um inventário de 7 vogais,
- **RITMO:** O ritmo da fala também é uma característica importante da língua. Enquanto que o português é uma língua *syllable-timed*[2], onde cada sílaba é pronunciada com certa clareza, o inglês é *stress-timed*[3], resultando numa compactação de sílabas, produzindo contrações e exibindo um fenômeno de redução de vogais como conseqüência.

Segundo Vieira (2010), o português é a língua oficial de sete países e a quinta língua mais falada no mundo, atrás apenas do chinês (quase 845 milhões de falantes), do espanhol (quase 358 milhões de falantes nativos), do inglês (quase 322 milhões de falantes nativos), e do hindu/urdu (quase 200 milhões de falantes nativos). É falado por aproximadamente 200 milhões de pessoas nos países (Angola, Brasil, Cabo Verde, Guiné Bissau, Moçambique, Portugal, São Tomé e Príncipe, Timor Leste), que se enquadram nos países lusófonos, ou seja, países que falantes da Língua Portuguesa, destacados na Figura 2.

[2] Cronometrada por sílabas (tradução proposta pelo autor).
[3] Cronometrada por acentuação tônica (tradução proposta pelo autor).

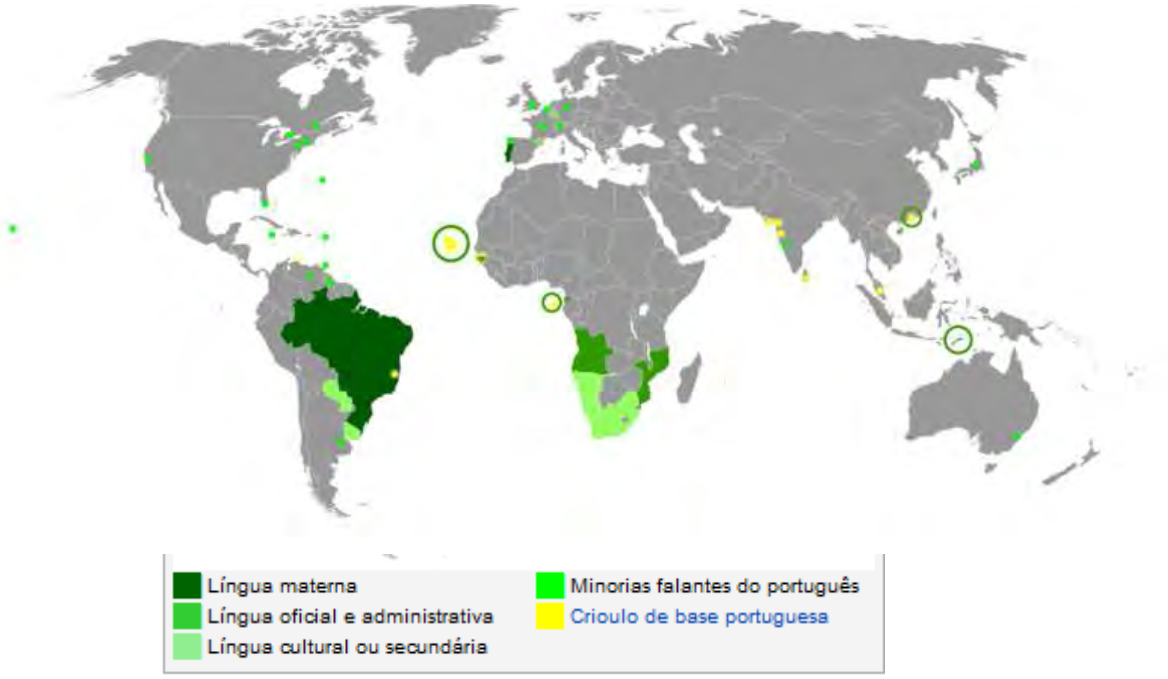

Figura 2: Países falantes da Língua Portuguesa
Fonte: WIKIPEDIA, 2011

Com o conhecimento mais aprofundado sobre a gramática da Língua Portuguesa, será possível um entendimento mais claro dos próximos capítulos, pois serão explicados com detalhes estratégias para solução da busca fonética.

2.2 Busca fonética

A busca por agilidade e resultados que facilitem o dia a dia de uma empresa ou mesmo de um órgão público é de extrema importância. Para isso, ter um banco de dados com função fonética é importante para adquirir confiabilidade nas pesquisas. A busca fonética é uma estratégia pela qual, de acordo com a palavra, são aplicadas regras gramaticais para obtenção de um código fonético, através do qual, palavras que contêm escritas diferenciadas, porém com a mesma fonética, possam ser encontradas na aplicação.

Com a busca de resultados positivos e com a grande competitividade no mercado empresarial, as informações contidas nos sistemas gerenciadores de banco de dados são importantes, tanto para análises comerciais quanto para gerenciamento das informações cadastradas.

Segundo Thorn (2011), o algoritmo de busca fonética dos bancos de dados citados neste trabalho funciona somente para a língua inglesa, apresentando

problemas com outros idiomas e gerando, portanto, uma limitação para o mercado brasileiro.

Existem diversas formas ou estratégias para programar a busca fonética no banco de dados. Nesse sentido, este trabalho visa implementar e aperfeiçoar a busca fonética para a Língua Portuguesa, a qual contém uma gama de palavras que, ao serem pronunciadas, apresentam o mesmo som, a mesma fonética, embora possam ser escritas de diversas formas diferentes, o que acaba por gerar um problema típico de aplicação computacional em relação à consulta correta do tema desejado.

Como se viu, não é possível a implementação da busca fonética para outros idiomas além do inglês, usando apenas as funções nativas do banco de dados. É necessário um estudo sobre o fonema da gramática para novas implementações ou aperfeiçoamentos no código existente. Existem 3 algoritmos que vêm nativos nos bancos de dados MYSQL e Postgree: Soundex, Metaphone e Double Metaphone; para o Firebird, existem implementações que possibilitam usar esse recurso. Os capítulos 2.3, 2.4, 2.5 e 2.6 apresentarão, em detalhes, cada uma das estratégias.

2.3 Soundex

O algoritmo Soundex foi criado por Robert C. Russell no início do século XX, tendo como principal objetivo rastrear a cidade e os nomes de famílias de imigrantes não americanos durante o período do censo de 1918 a 1920. O objetivo de Russell era ordenar as palavras não em ordem alfabética e sim pela forma como soam as palavras.

A patente original foi arquivada no dia 25 de outubro 1917, quando se intitulava “Index”, simplesmente. Com o passar do tempo, surgiram outras funções para a solução do problema;entretanto, de acordo com os SGBD estudados neste trabalho, o Soundex será detalhado pelo fato de ser uma função incluída nativamente na maioria dos sistemas de gerenciamento de bancos de dados. Na Listagem de Código Fonte 1, é apresentado o seu pseudocódigo.

```
Procedimento Soundex
Inicio
     1) Excluir todas as letras acentuadas;
     2) Manter a primeira letra da palavra e atribuir zero para as
     letras;
     (a, e, i, o, u, h, w, y);
     3) Atribuir os seguintes números para as letras restantes de
     acordo com o Quadro 2;
     4) Remover dígitos iguais (pares) resultantes do passo 2;
     5) Converter para a seqüência “letra, dígito, dígito, dígito”.
Fim
```

Listagem de Código-Fonte 1: Pseudocódigo do algoritmo Soundex.

Fonte: DONALD, 1973

O Quadro 2 mostra como é feita no Soundex a classificação das consoantes, além das regras de como deve ser feita a troca das consoantes por números. A regra segue a classificação de sons das consoantes pronunciados no idioma inglês (labiais e lábio-dental, guturais e sibilante, dental, palatal fricativa, lábio nasal e fricativa). Cada letra, de acordo com sua classificação, é substituída por um número correspondente (FRANTZ, 2009).

Quadro 2 – Regra de atribuição de números para consoantes no algoritmo Soundex

Classificação	Letras	Número
Labiais e Lábio-dental	B,F,P,V	1
Guturais e sibilante	C,G,J,K,Q,S,X,Z	2
Dental	D,T	3
Palatal fricativa	L	4
Lábio nasal	M,N	5
Fricativa	R	6

Fonte: FRANTZ, 2009 - Adaptada pelo autor.

Para maior facilidade de entendimento do algoritmo criado, será realizado um rastreio do pseudocódigo apresentado na Listagem de Código Fonte 1, mostrando o passo a passo das rotinas para as quais a função executa. O pseudocódigo mostrado pode ser encontrado no livro escrito por Donald E. Knuth, em 1973.

Para o rastreio foram escolhidos as palavras “*Double*” e “*Quadratically*” – a primeira contempla a atribuição de zeros à esquerda de acordo com o passo 5 do algoritmo e a outra irá completar a sequência desejada de “letra, dígito, dígito, dígito”. Os rastreios são apresentados nos Quadros 3 e 4.

Quadro 3 – Rastreio da palavra Double no algoritmo Soundex

Palavra	**Passo**	**Resultado**
Double	1- Manter a primeira letra da palavra (D), e atribuir zero para as letras O, U e E.	D00BL0
Double	2- Atribuir número 1 para letra B, atribuir número 4 para a letra L.	D00140
Double	3- Remover dígitos iguais ()	D00140
Double	4- Passar para sequência Letra, dígito, dígito, dígito e acrescentar zero a esquerda quando o resultado for menor que 3 dígitos.	D140

Fonte: Elaborada pelo autor.

Quadro 4 – Rastreio da palavra Quadratically no algoritmo Soundex

Palavra	**Passo**	**Resultado**
Quadratically	1- Manter a primeira letra da palavra (Q), e atribuir zero para as letras U, A, I, Y.	Q00DR0T0C0LL0
Quadratically	2- Atribuir número 3 para letra D, T, número 6 para a letra R, número 2 para a letra C, número 4 para a letra L.	Q00360302440
Quadratically	3- Remover dígitos iguais (LL)	Q360302040
Quadratically	4- Passar para sequência Letra, dígito, dígito, dígito sem acrescentar zeros à direita porque a palavra já contempla a sequência do algoritmo.	Q363

Fonte: Elaborada pelo autor.

Para que seja possível realizar a busca no banco de dados, existe um campo específico para receber o resultado fonético da palavra, e é neste campo que a busca será realizada. Assim, o comando de pesquisa consultará o código fonético Q363 ou D140 de acordo com os exemplos mostrados nos Quadros 3 e 4.

De acordo com o estudo realizado para as palavras *Double* e *Quadraticall,* o algoritmo Soundex contempla satisfatoriamente todos os passos que o algoritmo realiza, gerando um resultado fonético que pode ser utilizado para futuras consultas aos bancos de dados através dessa função.

O Soundex contém limitações quanto ao não funcionamento para outros idiomas, pelo fato de ter sido desenvolvido para a língua inglesa, mostrando ineficiência na atribuição e aplicação a nomes judeus, germânicos e eslavos (FRANTZ, 2009).

2.4 Metaphone

O algoritmo Metaphone foi desenvolvido por Lawrence Philips em 1990, muitos anos após a criação do algoritmo Soundex, desenvolvido por Russell em 1918. O código criado por Lawrence foi publicado em 1990 pela revista *Computer Language Magazine* (volume 7, número 12, página 38). Seu principal objetivo é resolver ineficiências do Soundex, sem limites de caracteres para o resultado final. O Metaphone contém uma cadeia de caracteres variável e realiza um conjunto maior de regras também para o idioma em Inglês, enquanto o Soundex limita-se em Letra-dígito-dígito-dígito.

O algoritmo do Metaphone é mais extenso, pelo fato de tratar mais detalhadamente a língua inglesa e de ter regras específicas para algumas consoantes. A Listagem de Código Fonte 2 mostra detalhadamente o procedimento do Metaphone.

```
Procedimento Metaphone
Inicio
     1) São realizados tratamentos em 16 consoantes
     0BFHJKLMNPRSTWXY, sendo que o 0 representa ‘TH’;
     2) Vogais A, E, I, O e U são usadas somente se estiverem no
     inicio da palavra;
     3) Tirar letras iguais, com exceção da letra c;
     4) Se a palavra começar com KN, GN, PN, AE, CR retirar a
     primeira letra;
     5) Retirar a letra B se for depois de M e no final da palavra;
     6) C se transforma em X seguido por IA ou H;
     C se transforma em S seguido por I, E ou Y, caso contrário se
     transforma em K;
     7) D se transforma em J seguido de GE, RG, ou IG. Caso
     contrário D se transforma em T;
     8) Tirar letra G se seguido por H e se o H não estiver no fim
     ou antes de uma vogal;
     Tirar letra G se seguido por N ou NED estando no final da
     palavra;
     9) G se transforma em J se antes de I, E ou Y. Caso contrário
     G se transforma em K;
     10) Retirar H se estiver depois de uma vogal;
     11) CK transforma-se em K;
     12) PK se transforma em F;
     13) Q se transforma em K;
     14) S se transforma em X se seguido por H, IO, IA;
     15) T transforma em X se seguido por IA, IO, TH (lembrando que
     para TH é atribuído o número 0). Retirar T se seguido por CH;
```

```
      16) V transforma em F;
      17) WH se transforma em W se estiverem no início da palavra e
      retirar o W se não estiverem seguidas por uma vogal;
      18) X se transforma em S se estiver no começo, caso contrário
      X se transforma em KS;
      19) Retirar Y se não estiver seguidas por uma vogal;
      20) Z se transforma em S.
Fim
```

Listagem de Código-Fonte 2: Pseudocódigo do Procedimento Metaphone.
Fonte: GALVEZ, 2006.

Quadro 5 – Rastreio do algoritmo Metaphone

Regra	**Condição/Exemplos**
São realizados tratamentos em 16 consoantes 0BFHJKLMNPRSTWXY	0 = th
Vogais A, E, I, O e U são usadas somente se estiverem no início da palavra	Ex: After, england, able
Tirar letras iguais, com exceção da letra C	Ex: All, fall Exemplo de palavra que não retira a letra c (according)
Se a palavra começar com KN, GN, PN, AE, CR	Retirar a primeira letra
Retirar a letra B se for depois de M e no final da palavra	Ex: Numbers
C transforma-se em X seguido por IA ou H C transforma-se em S seguido por I, E ou Y; caso contrário, transforma-se em K.	Ex: sufficient, especially
D transforma-se em J seguido de GE, RG, ou IG; caso contrário, D se transforma em T	
Tirar letra G se seguida por H e se o H não estiver no fim ou antes de uma vogal. Tirar letra G se seguido por N ou NED, estando no final da palavra.	
G transforma-se em J se antes de I, E ou Y; cso contrário, G, transforma-se em K.	Origin, give
Retirar H se estiver depois de uma vogal.	When
CK transforma-se em K	
PK transforma-se em F	
Q transforma-se em K	
S transforma-se em X se seguido por H, IO, IA	Shakespeare
T transforma-se em X se seguido por IA, IO, TH (lembrando que para TH é atribuído o número 0). Retirar T se seguido por CH	Examination
V transforma-se em F	
WH transforma-se em W se estiverem no início da palavra; retira-se o W se não estiverem seguidas por uma vogal.	Who

X transforma-se em S se estiver no começo; caso contrário X, transforma-se em KS	Except
Retirar Y se não estiver seguida por uma vogal	
Z transforma-se em S	

Fonte: Elaborada pelo autor.

2.5 Double Metaphone

O algoritmo Double Metaphone também foi criado e publicado por Lawrence Philips em 2000 na revista *C/C++ Users Journal,* através do artigo *"The Double Metaphone Search Algorithm* (PHILLIPS, 2011). Seu maior diferencial era o retorno de duas chaves se uma palavra tivesse duas pronúncias diferentes, por isso a inclusão da palavra Double no nome da função. Inicialmente em C++, foi implementado para a língua inglesa, assim como os demais algoritmos fonéticos.

O Double Metaphone corrige irregularidades de várias línguas, dentre elas: inglês, alemão, grego, francês, italiano e outras. Portanto, usa-se um conjunto de regras e análises mais complexas para resolver essas questões (PHILLIPS, 2011).

É possível citar, por exemplo, as regras que o algoritmo realiza quando se trata da letra ‘C’, o que pode ser observado na Listagem de Código-Fonte 3 parte do algoritmo.

```
      case 'C':
   if((current > 1  )//various germanic
      AND !IsVowel(current - 2)
      AND StringAt((current - 1), 3, "ACH", "")
      AND ((GetAt(current + 2) != 'I')
           AND ((GetAt(current + 2) != 'E')
      OR StringAt((current - 2), 6, "BACHER", "MACHER", "")) )) {
         MetaphAdd("K"); current +=2; break;
   }
   //special case 'caesar'
   if((current == 0) AND StringAt(current, 6, "CAESAR", "")) {
      MetaphAdd("S"); current +=2; break;
   }
   if(StringAt(current, 4, "CHIA", "")) {  //italian 'chianti'
      MetaphAdd("K"); current +=2; break;
   }
   if(StringAt(current, 2, "CH", "")) {
      //find 'michael'
      if((current > 0) AND StringAt(current, 4, "CHAE", "")) {
         MetaphAdd("K", "X"); current +=2; break;
      }
```

```
      if((current == 0)  //greek roots e.g. 'chemistry', 'chorus'
         AND (StringAt((current + 1), 5, "HARAC", "HARIS", "")
         OR StringAt((current + 1), 3, "HOR",
                     "HYM", "HIA", "HEM", ""))
         AND !StringAt(0, 5, "CHORE", "")) {

         MetaphAdd("K"); current +=2; break;
      }
      //germanic, greek, or otherwise 'ch' for 'kh' sound
      if((StringAt(0, 4, "VAN ", "VON ", "")
         OR StringAt(0, 3, "SCH", ""))

         // 'architect but not 'arch', 'orchestra', 'orchid'
         OR StringAt((current - 2), 6, "ORCHES", "ARCHIT",
                     "ORCHID")
         OR StringAt((current + 2), 1, "T", "S", "")
         OR ((StringAt((current - 1), 1, "A", "O", "U", "E", "")
             OR (current == 0))

         //e.g., 'wachtler', 'wechsler', but not 'tichner'
         AND StringAt((current + 2), 1, "L", "R", "N", "M", "B",
                      "H", "F", "V", "W", " ", ""))) {
         MetaphAdd("K");
      } else {
         if(current > 0) {
            if(StringAt(0, 2, "MC", ""))  //e.g., "McHugh"
               MetaphAdd("K");
            else MetaphAdd("X", "K");
         }
            else  MetaphAdd("X");
      }
         current +=2; break;
   }
   //e.g, 'czerny'
   if(StringAt(current, 2, "CZ", "")
      AND !StringAt((current - 2), 4, "WICZ", "")) {
         MetaphAdd("S", "X"); current += 2; break;
   }
   //e.g., 'focaccia'
   if(StringAt((current + 1), 3, "CIA", "")) {
      MetaphAdd("X"); current += 3; break;
   }
   //double 'C', but not if e.g. 'McClellan'
   if(StringAt(current, 2, "CC", "") AND !((current == 1)
      AND (GetAt(0) == 'M')))
      //'bellocchio' but not 'bacchus'
      if(StringAt((current + 2), 1, "I", "E", "H")
         AND !StringAt((current + 2), 2, "HU", "")) {

         //'accident', 'accede' 'succeed'
         if(((current == 1) AND (GetAt(current - 1) == 'A'))
            OR StringAt((current - 1), 5, "UCCEE", "UCCES", ""))

            MetaphAdd("KS");
         else  //'bacci', 'bertucci', other italian
```

```
            MetaphAdd("X");

         current += 3; break;
      } else {//Pierce's rule
         MetaphAdd("K"); current += 2; break;
      }
   if(StringAt(current, 2, "CK", "CG", "CQ", "")) {
      MetaphAdd("K"); current += 2; break;
   }
   if(StringAt(current, 2, "CI", "CE", "CY", "")) {
      //italian vs. english
      if(StringAt(current, 3, "CIO", "CIE", "CIA", ""))
         MetaphAdd("S", "X");
      else
         MetaphAdd("S");
      current += 2; break;
   }  //else
   MetaphAdd("K");

   //name sent in 'mac caffrey', 'mac gregor
   if(StringAt((current + 1), 2, " C", " Q", " G", ""))
      current += 3;
   else
      if(StringAt((current + 1), 1, "C", "K", "Q", "")
         AND !StringAt((current + 1), 2, "CE", "CI", ""))

         current += 2; else current += 1;
   break;
```

Listagem de Código-Fonte 3: Código para o algoritmo Double Metaphone tratando a letra C
Fonte: PHILLIPS, 2006.

2.6 NYSIIS

Conhecido como NYSIIS *(New York State Identification and Intelligence Systems),* o Sistema de Identificação e Inteligência do Estado de Nova York foi desenvolvido em 1970 por Taft e contém um aumento de 2,73% de precisão em relação ao Soundex. Ou seja, a função Soundex contém 95,99% de precisão, enquanto o NYSIIS possui precisão de 98,72 (GALVEZ, 2006). O algoritmo se resume em realizar as seguintes estratégias:

```
Procedimento NYSIIS
Inicio
     1) O primeiro caractere da chave fonética corresponde ao
primeiro caractere da palavra
     2)Traduz os primeiros caracteres da seguinte forma:
          MAC=>MCC
          PH=>FF
```

```
            KN=>NN
            K=>C
            SCH=>SSS
      3) Traduz os últimos caracteres da seguinte forma:
            EE=>Y
            IE=>Y
            DT=>D
            RT=>D
            RD=>D
            NT=>D
            ND=>D
      4) Se o último caractere for S, então elimina-o;
      5) Se o último caractere for A, então elimina-o;
      6) Se os últimos caracteres forem A ou Y, então substitui por
      Y.
Fim
```

Listagem de Código-Fonte 4: Procedimento do pseudocódigo NYSIIS
Fonte: GALVEZ, 2006

De acordo com o estudo de Gálvez (2006), executando o algoritmo para as palavras do Quadro 6, os resultados da função fonética serão:

Quadro 6 – Códigos fonéticos e acordo com o algoritmo NYSIIS

	Apellidos	Variantes	Códigos
Match	Appelt	Apelt	(APALT, APALT)
	Hobbs	Hubbs	(HAB, HAB)
Mismatch	Appelt	Appell	(APALT, APAL)
	Hobbs	Hobds	(HAB, HABD)

Fonte: GALVEZ, 2006.

2.7 Comparativo entre os algoritmos de busca fonética

No Quadro 7, é possível visualizar claramente os resultados e uma comparação de cada função fonética estudada nos capítulos acima. Observa-se que, de acordo com cada rotina específica, o tratamento para as consoantes e vogais será diferente e que cada algoritmo fornece novas estratégias com o intuito de aumentar o desempenho da busca fonética.

Quadro 7 – Resultado de palavras utilizando as funções Soundex, NYSIIS, Metaphone, Double-Metaphone.

Nome	Soundex	NYSIIS	Metaphone	Double-Metaphone
Bill	B400	BAL	BL	PL
Boyle	B400	BAYL	BL	PL
Carlson	C642	CARLSA	KRLSN	KRLS
Charles	C642	CHARL	XRLS	XRLS
Harry	H600	HARY	HR	HR
Hauer	H600	HAR	HR	HR
Jackson	J250	JACSAN	JKSN	JKSN
Lee	L000	LY	L	L
Robert	R163	RABAD	RBRT	RPRT
Robertson	R163	RABART	RBRTSN	RPRT
Rupert	R163	RAPAD	RPRT	RPRT
Washington	W252	WASANG	WXNKTN	AXNK

Fonte: HJORT et al, 2007.

De acordo com o Quadro 7, é possível visualizar os diferentes resultados que cada algoritmo propõe para cada nome descrito. A evolução das estratégias e dos estudos para melhorar o resultado e simplificar a busca com o intuito de exercer eficiência nas consultas utilizando a busca fonética.

Quanto ao algoritmo Soundex percebe-se que a primeira letra é mantida e números são atribuídos para as demais letras, diferente das funções NYSIIS, Metaphone e Double-Metaphone, que não atribuem número para as letras e realizam novas formas de geração do código fonético.

3 TRABALHOS RELACIONADOS

Neste capítulo, estão relacionados os trabalhos presentes na literatura semelhantes ao escopo deste.

Existem poucas pesquisas e trabalhos na área de banco de dados voltados para a obtenção de soluções mais eficazes para a busca fonética. A maioria das funções restringe-se somente em algoritmos que estão contidos nas instalações dos SGBD, tais funções foram descritas no decorrer de todo este projeto. Portanto, a obtenção de poucos trabalhos sobre o assunto é devido a pequena quantidade de pesquisas, mas possibilitando a citação dos trabalhos abaixo:

- *Busca Fonética em Português do Brasil (Lucena, 2007):* Dissertação que apresenta um novo código fonético para o Português do Brasil. Elabora uma nova estratégia para soluções mais eficazes, onde a chamou de função BUSCABR. Durante o projeto houve uma preocupação em exemplificar a diversidade da Língua Portuguesa e os possíveis erros que ocorrem na escrita e no entendimento das palavras. O objetivo do estudo é promover uma função mais abrangente, gerando como resultado mais agilidade nas consultas.

- *Protótipo de Um Reconhecimento Fonético Aplicado ao Banco de Dados Oracle(ROSSETTO, 2000):* Trabalho de conclusão de curso que exemplifica de forma clara a implementação da função fonética para o banco de dados Oracle. O projeto realizado específica em soluções para aplicações WEB. Tem como principal objetivo diminuir a quantidade de registros que não são encontrados no banco de dados ou que ficam perdidos por conta de cadastros incorretos. A função foi elaborada em PL/SQL e o Oracle Web Server.

- *Código Fonético no Firebird* (BUBLITZ, 2006): Artigo escrito na revista Clube Delphi, que exemplifique e ensina a criação de uma UDF para vinculo no SGBD Firebird, tendo como principal objetivo a solução da busca fonética para a Língua Portuguesa. O algoritmo foi escrito na linguagem Pascal e resolve algumas questões quanto a gramática da Língua Portuguesa, mas falha quando realizado o tratamento de palavras com “SC” e “Ç”.

4 METODOLOGIA

Desde o início do projeto e das pesquisas realizadas, o principal objetivo foi a obtenção de materiais com informações importantes sobre o assunto, para garantir mais confiabilidade aos resultados mostrados, pois o tema proposto reflete diretamente nas funções já criadas para solução do problema.

O projeto foi realizado em etapas, que serão apresentadas de forma organizada para que todo o raciocínio feito seja entendido com mais clareza.

1-Levantamento Bibliográfico: Estudo criterioso sobre o assunto, para que fosse possível um levantamento de informações que se tornou base para a criação de uma nova estratégia e desenvolvimento de um novo sistema abrangendo a busca fonética para a Língua Portuguesa.

2-Diferenciação da Língua Inglesa para a Língua Portuguesa e entendimento das principais dificuldades que o Português pode trazer para a implementação: Para entender as funções fonéticas já existentes, há necessidade de um entendimento da Língua Inglesa, e para que seja implementado na Língua Portuguesa, devem-se entender as principais dificuldades que as regras gramaticais podem trazer. Após o estudo sobre os fonemas consonantais e vocálicos e conhecimento mais abrangente sobre fonética da Língua Portuguesa, novas soluções e estratégias poderão ser desenvolvidas.

3-Entendimento dos principais algoritmos já existentes para busca fonética: Estudo realizado sobre as estratégias já existentes serve como base importante para mostrar as eficiências e ineficiências dos algoritmos. A partir deste

estudo, é possível obter informações e resultados das análises de cada função. Assim uma melhor visão do que precisa ser incluído para que a solução seja mais completa poderá ser obtida.

4-Verificar a existência de soluções para os gerenciadores de bancos de dados Firebird, MySQL e PostGreeSQL em termos de busca fonética, inclusive busca fonética para o Português: Inicialmente, o principal objetivo do projeto, foi a identificação de trabalhos que realizam esta função para os bancos de dados MYSQL, Postgree e Firebird. Em constantes pesquisas e conversas com gerenciadores de banco de dados, obteve-se a certeza de que não existem implementações para o MYSQL e Postgree, apenas os já citados algoritmos Soundex, Metaphone e Double Metaphone. Para o Firebird, foi encontrada uma UDF publicação que realiza a função fonética. A partir dessa função, será realizada o aperfeiçoamento melhoria de acordo com as regras gramaticais da Língua Portuguesa.

5-Implementação do algoritmo BuscaSonora para pelo menos um dos SGBD citados no trabalho: Realizadas implementações para cada SGBD citado no projeto, foi possível a obtenção de resultados de palavras tanto em Inglês quanto em Português. Este processo foi importante devido ao estudo realizado dos algoritmos, sendo possível analisar com detalhes as estratégias e soluções que cada função exerce.

5 ESTUDO DOS ALGORITMOS DE BUSCA FONÉTICA EXISTENTES EM SGBD LIVRES

Este capítulo apresenta os testes efetuados sobre os algoritmos de busca fonética descritos neste trabalho. O intuito é verificar a eficácia de tais algoritmos para a Língua Portuguesa.

Também foram realizados inúmeros testes com soluções publicadas especificamente para o idioma português. O objetivo de tais testes é comprovar a eficiência dos algoritmos para todas as variações fonéticas existentes na Língua Portuguesa.

Ao final do capítulo, encontra-se a sugestão de um possível algoritmo para efetuar a busca fonética em Sistemas Gerenciadores de Banco de Dados (SGBD) *open-source,* considerando as regras gramaticais e fonéticas do idioma português.

5.1 Busca fonética em SGBD livres.

Esta parte do trabalho visa investigar se alguns dos SGBD livres, como Firebird, MySQL e PostGreSQL, possuem soluções para busca fonética, e se estas soluções estão adequadas ao idioma português. Os ambientes preparados para execução dos testes estão relatados no Apêndice A deste trabalho.

5.1.1 MYSQL

O sistema gerenciador de banco de dados (SGBD) MySQL, possui nativamente o algoritmo Soundex, que por sua vez, pode ser consultado no manual disponibilizado pelo site do MySQL.

Os comandos para execução e obtenção de resultados utilizando a estratégia Soundex,podem ser executados de acordo com o Quadro 8.

Quadro 8 – Exemplos de utilização do Soundex no SGBD MySQL

COMANDO SQL	RESULTADO
`SELECT SOUNDEX('Hello');`	`'H400'`
`SELECT SOUNDEX('Quadratically');`	`'Q36324'`

Fonte: THORN, 2011.

Atualmente, o MySQL possui apenas o Soundex implementado diretamente em sua instalação, não sendo possível utilizar algoritmos como Metaphone ou Double Metaphone.

5.1.2 POSTGRESQL

O SGBD PostgreSQL possui três possibilidades de utilização da busca fonética em sua própria instalação, ou seja, é implementado diretamente no banco de dados e disponibilizado gratuitamente em suas versões. Após estabelecer um estudo sobre a documentação disponibilizada no site do PostgreSQL, verifica-se que esse gerenciador de banco de dados contém as seguintes funções:

- Soundex
- Metaphone
- Double Metaphone

Para utilização de cada uma das funções mencionadas acima, o processo de geração da SQL é simples. O Quadro 9 disponibiliza o comando para execução do processo.

Quadro 9 – Exemplos de utilização do Soundex, Metaphone e Double Metaphone no SGBD PostGreSQL.

FUNÇÃO	COMANDO	RESULTADO
Soundex	Select soundex(‘ANNE’)	A500
Metaphone	Select metaphone (‘GUMBO’, 4)	KM
Double Metaphone	Select dmetaphone (‘GUMBO’)	KMP

Fonte: POSTGRESQL, 2011.

5.1.3 FIREBIRD

Tendo em vista os algoritmos Soundex, Metaphone, Double Metaphone, que são disponibilizados nativamente tanto no MySQL quanto no PostgreSQL, segundo (GOMEZ, 2005) o Firebird não contempla nenhuma mecanismo de busca fonética internamente em sua instalação, mas permite implementações de funções definidas pelo usuários, ou seja, as UDF, agregando portanto, funções aos bancos de dados em questão.

5.2 Aplicação dos Algoritmos de Busca Fonética já Existentes para a Língua Portuguesa

5.2.1 Soundex

Foram realizados alguns testes para comprovar ou não funcionamento do algoritmo Soundex para a Língua Portuguesa,

Quando realizada uma consulta Soundex para as palavras Casa e Caju, de acordo com o algoritmo proposto, será obtido o mesmo resultado, ou seja, palavras com fonéticas totalmente distintas terão resultados semelhantes, ou seja, cadeia de caracteres igual a C200.

Quadro 10 – Rastreio da palavra Casa e Caju no algoritmo Soundex

Palavra 1	**Palavra 2**	**Resultado Soundex**	**Resultado do estudo**
Casa	Caju	C200	Não é válido

Fonte: Elaborada pelo autor.

Pensando em nomes próprios, o algoritmo falha nas situações em que se digitam Walter ou Valter, pois são obtidos resultados diferentes, ou seja, para palavras com a mesma fonética, foram atribuídas cadeias de caracteres distintas, como W436 para Walter e V436 para Valter.

Quadro 11 – Exemplo da palavra Walter e Valter no algoritmo Soundex

Palavra 1	**Palavra 2**	**Resultado Soundex 1**	**Resultado Soundex 2**	**Resultado do estudo**
Walter	Valter	W436	V436	Não é válido

Fonte: Elaborada pelo autor.

Segundo Lucena (2007), a substituição de todas as vogais por zero e mais as letras H, W e Y são estratégias que não funcionarão para a Língua Portuguesa, pois, a letra C seguida de vogais A/O/U possui som de K, enquanto que as seguidas de E/I têm som de Ç. Segundo Bublitz (2006), é possível ignorar as vogais e fazer o tratamento das consoantes que as antecedem, ou seja, além da letra C seguida de vogais E/I terem som de Ç, também podem ter fonética parecida com a letra S, de acordo com a Listagem de Código Fonte 5 mostrada na implementação da busca fonética no Firebird.

Lucena (2007) comparou também a não distribuição de números para as correspondentes letras, pois, a Língua Portuguesa contém 31 fonemas, havendo um tratamento mais específico para as letras e não a simples atribuição de números (UOL, 2011).

5.2.2 METAPHONE

A partir da explicação detalhada do que o algoritmo realiza para a geração da cadeia de caracteres fonética (Capítulo 2.4), é possível a realização de testes para comprovar o funcionamento. Foram escolhidas palavras que, de acordo com sua escrita, serão inseridas nas regras descritas na Listagem de Código Fonte 2.

Quadro 12 – Exemplo da palavra Numbers sendo tratada no Metaphone

Palavra	Passo	Resultado
Numeros	1- Manter letra N	NUMEROS
Numeros	2 – Retirar vogal se não estiver no início	NMRS
Numeros	4 – M recebe M, R recebe R e S recebe S não em X, pelo fato de não estar seguido de IA, IO, TH.	NMRS

Fonte: Elaborada pelo autor.

Quadro 13 – Exemplo da palavra Knowledge sendo tratada no Metaphone

Palavra	Passo	Resultado
Conhecimento	1- A letra C se transforma em K	KONHECIMENTO
Conhecimento	2 – Retirar vogais	KNHCMNT
Conhecimento	3 – Letra C se transforma em S pois estava seguido pela letra I.	KNHSMNT

Fonte: Elaborada pelo autor.

Segundo Hjort et al (2009), surgiram outros algoritmos de busca fonética para tentativa de solução mediante as falhas do Soundex, por exemplo o algoritmo NYSIIS, que foi desenvolvido pela divisão Estadual de Serviços de Justiça Criminal de Nova Iorque em 1970 e que será explicado no capitulo 2.6.

Ao se realizar o estudo sobre o Metaphone em palavras da Língua Portuguesa, chega-se à conclusão de que a estratégia não pode ser usada mediante a ortografia e as variedades da Língua Portuguesa. De acordo com estudos de nomes próprios, a função falhará em testes realizados com o nome “Shirlei” e “Xirlei”, havendo resultados diferentes para nomes com fonemas iguais.

Quadro 14 – Exemplo dos nomes Xirlei e Shirlei sendo tratados no Metaphon

Palavra 1	**Palavra 2**	**Resultado Metaphone Xirlei**	**Resultado Metaphone Shirlei**	**Resultado do estudo**
Xirlei	Shirlei	SRL	XRL	Não é válido

.Fonte: Elaborada pelo autor.

O código também se restringe a variações que a Língua Portuguesa proporciona, sendo mais um motivo para a não utilização dele em mercados brasileiros. É possível citar como exemplo o nome “Fatima” ou “Fathima”, um nome com variações e que a função Metaphone resulta em resultados fonéticos diferentes.

Quadro 15 – Exemplo dos nomes Fatima e Fathima sendo tratada no Metaphone.

Palavra 1	**Palavra 2**	**Resultado Metaphone Fatima**	**Resultado Metaphone Fathima**	**Resultado do estudo**
Fatima	Fathima	FTM	F0M	Não é válido

Fonte: Elaborada pelo autor.

A inclusão do número 0 ao resultado (F0M) do nome “Fathima” deve-se ao fato de o algoritmo incluir 0 quando a letra T for seguida da letra H, ou seja, TH. Sempre que houver essa situação, será incluído o número 0. Portanto, em nomes como “Thiago” e “Tiago”, sempre ocorrerão problemas.

5.3 Soluções alternativas para busca fonética no idioma português em SGBD livres

Esta seção apresenta algumas soluções para utilização de busca fonética no idioma português encontradas em publicações e na literatura. O objetivo aqui é

verificar se tais soluções são realmente válidas para a Língua Portuguesa e apresentar como tais soluções funcionam. A seguir, é mostrado como os testes foram realizados e a análise realizada sobre os resultados alcançados.

5.3.1 SOLUÇÃO 1

O Firebird não possui nativamente nenhum dos algoritmos de busca fonética tratados neste trabalho, porém a implementação da busca fonética, com suporte ao idioma português, já foi realizada e publicada por Jorge Luis Bublitz, publicada na revista Clube Delphi, 82, em 2006 (BUBLITZ, 2006). Esse trabalho considera tal publicação como uma solução de busca fonética para o SGBD Firebird.

O código de implementação do algoritmo foi desenvolvido em linguagem Delphi, baseado no algoritmo Metaphone. Ele realiza a conversão de uma palavra interpretada como sendo o nome em outra palavra equivalente, porém livre de caracteres que possam gerar duplicidades fonéticas de acordo com o alfabeto e regras ortográficas da Língua Portuguesa. As regras utilizadas são apresentadas na Listagem de Código Fonte 5.

```
Procedimento da Busca Fonética para  Língua Portuguesa no Firebird
Inicio
     1) Vogais(A, E, I, O, U), Y e o H são ignorados
     2) E, DA, DAS, DE, DI, DO e DOS são ignorados
     3) Havendo letras duplicadas a segunda letra é ignorada
     4) Manter Consoantes B, D, F, J, K, L, M, N, R, T, V e X
     5) Letra C Seguida de H tem som de X, seguida de A, O ou U tem
     som de K, seguida de E ou I tem som de S, senão é ignorado
     6) Letra G Seguida de E tem som de J
     7) Letra G Seguida de H tem som de F
     8) Letra Q Seguida de U tem som de K
     9) Letra S Seguida de H tem som de X, entre duas vogais tem
     som de Z Seguido de vogal é mantido, senão é ignorado
     10) Letra W Tem som de V
     11) Letra Z No final do nome tem som de S, senão é mantida
Fim
```

Listagem de Código-Fonte 5: Procedimento da busca fonética para o Firebird
Fonte: BUBLITZ, 2006.

O programa escrito pelo autor gera um arquivo de extensão DLL, que pode ser utilizada como uma UDF no Firebird. A DLL deve estar na pasta do Firebird, na

sub-pasta UDF e em seguida deve ser registrada no SGBD, através do comando mostrado na Listagem de Código Fonte 7.

A primeira solução encontrada foi publicada na revista Clube Delphi, 2006, Código Fonético no Firebird, por Jorge Luíz Bublitz.

Para o teste, foi criado um banco de dados contendo uma tabela, cujo nome é CLIENTE, que armazena os campos NOME e CAMPO_FONETICO. O código SQL para criação do banco de dados e para a criação da tabela CLIENTE pode ser visto na Listagem de Código Fonte 6.

```
CREATE TABLE CLIENTE (
NOME            VARCHAR(50) NOT NULL,
CAMPO_FONETICO  VARCHAR(50)
);
```

Listagem de Código-Fonte 6: SQL para criação do banco de dados e criação da tabela CLIENTE
Fonte: Elaborada pelo autor.

O campo NOME foi criado para receber o nome de um cliente, considerando uma aplicação qualquer. O campo CAMPO_FONETICO armazenará a codificação fonética da palavra considerada como nome, conforme apresentado.

```
DECLARE EXTERNAL FUNCTION FCCODIFONPT_BR
CSTRING(255) NULL
RETURNS CSTRING(255)
ENTRY_POINT 'CodiFonPT_BR' MODULE_NAME 'codfon';
```

Listagem de Código-Fonte 7: SQL para vinculo da UDF no software IBexpert.
Fonte: Elaborada pelo autor.

Após o conhecimento das regras mostradas para a busca fonética no Firebird de acordo com a Listagem de Código Fonte 5, são efetuados testes para verificar a eficiência da solução. A estratégia aqui é criar um gatilho que vai atuar atualizando o campo “CAMPO_FONETICO”, com a cadeia de caracteres fonética gerada pelo algoritmo proposto, de acordo com as palavras inseridas na tabela CLIENTE, no campo “NOME”.

Na Listagem de Código Fonte 8 é apresentado o script de criação do gatilho e na Listagem de Código Fonte 9 é mostrado o script de inserção na tabela CLIENTE com palavras que podem ser escritas de maneiras diferentes na Língua Portuguesa.

```
CREATE OR ALTER TRIGGER FONETICO_BI0 FOR CLIENTE
ACTIVE BEFORE INSERT OR UPDATE POSITION 0
AS
begin
     new.campo_fonetico = fccodifonpt_br(new.nome);
end
```

Listagem de Código-Fonte 8: SQL para criação do gatilho (trigger)
Fonte: Elaborada pelo autor.

```
INSERT INTO CLIENTE (NOME) VALUES ('JUSSARA’);
INSERT INTO CLIENTE (NOME) VALUES ('JUSARA');
INSERT INTO CLIENTE (NOME) VALUES ('JUÇARA');
INSERT INTO CLIENTE (NOME) VALUES ('LUIS');
INSERT INTO CLIENTE (NOME) VALUES ('LUIZ');
INSERT INTO CLIENTE (NOME) VALUES ('WAGNER');
INSERT INTO CLIENTE (NOME) VALUES ('VAGNER');
INSERT INTO CLIENTE (NOME) VALUES ('CARLA');
INSERT INTO CLIENTE (NOME) VALUES ('KARLA');
INSERT INTO CLIENTE (NOME) VALUES ('CHICO');
INSERT INTO CLIENTE (NOME) VALUES ('FATIMA');
INSERT INTO CLIENTE (NOME) VALUES ('FATHIMA');
INSERT INTO CLIENTE (NOME) VALUES ('FELIPE');
INSERT INTO CLIENTE (NOME) VALUES ('PHELIPE');
INSERT INTO CLIENTE (NOME) VALUES ('TIAGO');
INSERT INTO CLIENTE (NOME) VALUES ('THIAGO');
INSERT INTO CLIENTE (NOME) VALUES ('ACELINO');
INSERT INTO CLIENTE (NOME) VALUES ('ASSELINO');
INSERT INTO CLIENTE (NOME) VALUES ('ASCELINO');
```

Listagem de Código-Fonte 9: SQL para inserção dos dados na tabela Cliente
Fonte: Elaborada pelo autor.

Quadro 16 – Dados inseridos após a execução do insert de acordo com a listagem de Código Fonte 8.

NOME	CAMPO_FONETICO
JUSSARA	JZR
JUSARA	JZR
JUÇARA	JKR
WAGNER	VGN
VAGNER	VGN
CARLA	KRL
KARLA	KRL
CHICO	XK

FATIMA	FTM
FATHIMA	FTM
FELIPE	FLP
PHELIPE	FLP
ACELINO	SLN
ASSELINO	ZLN
ASCELINO	SSLN

Fonte: Elaborada pelo autor.

Após tudo configurado e todos os comandos executados, é possível realizar consultas e ver se os resultados correspondem ao esperado. Para realização das consultas, deve-se vincular o comando criado para utilizar a função fonética. A seguir, o exemplo da SQL para o nome Karla:

```
SELECT * FROM CLIENTE C
WHERE c.campo_fonetico = fccodifonpt_br('KARLA')
```

Listagem de Código-Fonte 10: SQL com seleção para o nome Karla
Fonte: Elaborada pelo autor.

O resultado obtido após a execução do comando de seleção é apresentado no Quadro 17. É possível perceber que ambas as palavras “Carla” e “Karla” foram retornadas, como esperado, por terem a mesma pronúncia.

Quadro 17 – Resultado fonético para os nomes Karla e Carla

NOME	CAMPO_FONETICO
CARLA	KRL
KARLA	KRL

Fonte: Elaborada pelo autor.

Foram realizados outros comandos para constatar o funcionamento da busca fonética no Firebird. Para os nomes Wagner (Listagem de Código Fonte 11) e Fatima (Listagem de Código Fonte 12), houve um resultado satisfatório, pois o objetivo foi alcançado e o retorno da busca contemplou as estratégias que a função realiza.

```
SELECT * FROM CLIENTE C
WHERE c.campo_fonetico = fccodifonpt_br('WAGNER')
```

Listagem de Código-Fonte 11: SQL com seleção para o nome Wagner
Fonte: Elaborada pelo autor.

Quadro 18 – Resultado fonético para os nomes Wagner e Vagner

NOME	CAMPO_FONETICO
WAGNER	VGN
VAGNER	VGN

Fonte: Elaborada pelo autor.

```
SELECT * FROM CLIENTE C
WHERE c.campo_fonetico = fccodifonpt_br('FATIMA')
```

Listagem de Código-Fonte 12: SQL com seleção para o nome Fatima
Fonte: Elaborada pelo autor.

Quadro 19 – Resultado fonético para os nomes Fatima e Fathima

NOME	CAMPO_FONETICO
FATIMA	FTM
FATHIMA	FTM

Fonte: Elaborada pelo autor.

Contudo, como a Língua Portuguesa contém inúmeras variações e formas de se criar um nome e interpretar palavras do dicionário, foram observados, após os testes, a não utilização e o não tratamento de algumas regras que devem facilitar a busca nos SGBD.

Com o cadastro desses três novos clientes – Acelino, Ascelino e Asselino –, os resultados foram três campos fonéticos distintos: SLN, ZLN e SSLN (Quadro 16), ou seja, não é possível utilizar o algoritmo para esta situação.

Abaixo, a constatação com a busca pelo nome Acelino, conforme Listagem de Código-Fonte 12, e o retorno de apenas um cadastro, de acordo com o Quadro 20, ocorrendo ineficiência. Para obter o resultado completo, a busca teria que contemplar os nomes Acelino, Ascelino e Asselino, de acordo com o Quadro 21.

```
SELECT * FROM CLIENTE C
WHERE c.campo_fonetico = fccodifonpt_br('ACELINO')
```

Listagem de Código-Fonte 13: SQL com seleção para o nome Acelino.
Fonte: Elaborado pelo autor.

Quadro 20 – Resultado fonético para o nome Acelino

NOME	CAMPO_FONETICO
ACELINO	SLN

Fonte: Elaborada pelo autor.

Quadro 21 – Demonstração de resultado para os nomes Acelino, Asselino e Ascelino

NOME
ACELINO
ASSELINO
ASCELINO

Fonte: Elaborada pelo autor.

O algoritmo falha também em situações em que a fonética do nome é a mesma, enquanto a forma de escrita é diferente. Os testes realizados abrangeram diversas possibilidades de nomes, dentre os quais é possível citar os nomes “Niltom” e “Nilton”. Quando executado o algoritmo descrito na revista Clube Delphi, 82, em 2006 (BUBLITZ, 2006), o retorno da busca fonética realizada será:

```
SELECT * FROM CLIENTE C
WHERE c.campo_fonetico = fccodifonpt_br('NILTOM')
```

Listagem de Código-Fonte 14: SQL com seleção para o nome Niltom
Fonte: Elaborada pelo autor.

```
SELECT * FROM CLIENTE C
WHERE c.campo_fonetico = fccodifonpt_br('NILTON')
```

Listagem de Código-Fonte 15: SQL com seleção para o nome Nilton.
Fonte: Elaborado pelo autor.

Quadro 22 – Resultado fonético para o nome Niltom

NOME	CAMPO_FONETICO
NILTOM	NLTM

Fonte: Elaborada pelo autor.

Quadro 23 – Resultado fonético para o nome Nilton

NOME	CAMPO_FONETICO
NILTON	NLTN

Fonte: Elaborada pelo autor.

O algoritmo contém falhas quando realizados testes e estudos da letra G, tendo em vista que Bublitz (2006) já realiza um tratamento no código original quando a letra G for acompanhada da letra E. Porém, em situações em que a letra G é acompanhada da letra I, a função não retorna um valor válido, ou seja, considerando-se os nomes “Gisele” e “Jisele”, o retorno será códigos fonéticos distintos.

```
SELECT * FROM CLIENTE C
WHERE c.campo_fonetico = fccodifonpt_br('GISELE')
```

Listagem de Código-Fonte 16: SQL com seleção para o nome Gisele
Fonte: Elaborada pelo autor.

```
SELECT * FROM CLIENTE C
WHERE c.campo_fonetico = fccodifonpt_br('JISELE')
```

Listagem de Código-Fonte 17: SQL com seleção para o nome Jisele
Fonte: Elaborada pelo autor.

Quadro 24 – Resultado fonético para o nome Niltom

NOME	CAMPO_FONETICO
GISELE	GSL

Fonte: Elaborado pelo autor.

Quadro 25 – Resultado fonético para o nome Nilton

NOME	CAMPO_FONETICO
JISELE	JSL

Fonte: Elaborado pelo autor.

Quando o estudo parte para a letra W, o sistema trata apenas se estiver a letra W, transformando em V, como por exemplo, nos nomes Walter e Valter. Mas é possível ter exemplos em que, no início da palavra, o W tem som de U, como nos nomes Walace e Ualace.

```
SELECT * FROM CLIENTE C
WHERE c.campo_fonetico = fccodifonpt_br('UALACE')
```

Listagem de Código-Fonte 18: SQL com seleção para o nome Ualace
Fonte: Elaborada pelo autor.

```
SELECT * FROM CLIENTE C
WHERE c.campo_fonetico = fccodifonpt_br('WALACE')
```

Listagem de Código-Fonte 19: SQL com seleção para o nome Walace
Fonte: Elaborada pelo autor.

Quadro 26 – Resultado fonético para o nome Niltom

NOME	CAMPO_FONETICO
UALACE	LC

Fonte: Elaborada pelo autor.

Quadro 27 – Resultado fonético para o nome Nilton

NOME	CAMPO_FONETICO
WALACE	WLC

Fonte: Elaborada pelo autor.

A partir dos testes realizados, chegou-se à conclusão deque o algoritmo publicado na revista Clube Delphi não é totalmente completo, porque falhou para várias das regras ortográficas do idioma português.

O algoritmo deve ser reavaliado para correção do problema aqui apontado e, a partir de então, vincular novamente a função a um novo banco de dados para novos testes e verificação se o que foi corrigido será satisfatório ou não.

5.3.2 SOLUÇÃO 2

Em 2005 foi realizado também um novo estudo sobre a busca fonética no Firebird, baseado no algoritmo Soundex, ou seja, seguindo o Quadro 28. A publicação do estudo foi realizado na revista DBFree Magazine, Volume 6, por Flavio Yamil Gomez.

Quadro 28 – Estratégia do algoritmo Soundex para busca fonética

Número	Letras
1	B, F, P, V, W
2	C, Ç, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

Fonte: Gomes, 2005.

Segundo Gomes (2005), a UDF foi implementada por meio de 4 funções (MV_soundex, soundex, SoundexPalavra e GrupoSoundex). Em ordem, a função MV_soundex é a função principal e chama a função Soundex; esta chama a função SoundexPalavra, que, por sua vez, chama a função GrupoSoundex.

Segundo Gomes (2005), torna-se necessário o vinculo da DLL MetaUDF na pasta padrão do Firebird C:\Program Files\Firebird\Firebird_2_5\UDF, para que as funções e o algoritmo funcionem corretamente.

De acordo com o Quadro 29, o algoritmo realiza as seguintes regras para geração do resultado final.

Quadro 29 – Regras para geração do resultado de acordo com a DLL MetaUDF

Regras
As vogais são descartadas, com exceção da primeira;
As consoantes H e Y são descartadas;
Se duas ou mais consoantes consecutivas equivalem ao mesmo grupo de conversão, a segunda e restante são descartadas;
Se a primeira letra da palavra for uma consoante e a segunda pertence ao mesmo grupo de conversão desta, a segunda é descartada.

Fonte: Gomes, 2005.

Após seguir as regras e vínculos para utilização da DLL MetaUDF, o sistema falhará em testes simples com nomes cuja pronúncia seja a mesma. O nome Carla, por exemplo, pode ser descritos como Karla, tendo como diferença as letras iniciais C e K. Outro exemplo que pode ser usado como referência é o nome Chico, cuja escrita pode variar, tendo em vista que pode ser escrito como Chico ou Xico

Executando o algoritmo proposto, os resultados obtidos serão distintos, mostrando valores fonéticos diferentes; portanto, é inviável a utilização desse algoritmo para a Língua Portuguesa.

```
select Mv_Soundex('carla') from
rdb$database
```

Listagem de Código-Fonte 20 – SQL com seleção para o nome Carla
Fonte: Elaborado pelo autor.

```
select Mv_Soundex('karla') from
rdb$database
```

Listagem de Código-Fonte 21 – SQL com seleção para o nome Karla
Fonte: Elaborado pelo autor.

Quadro 30 – Resultado fonético para o nome Carla

NOME	CAMPO_FONETICO
CARLA	C640

Fonte: Elaborado pelo autor.

Quadro 31 – Resultado fonético para o nome Karla

NOME	CAMPO_FONETICO
KARLA	K640

Fonte: Elaborado pelo autor.

```
select Mv_Soundex('chico') from
rdb$database
```

Listagem de Código-Fonte 22 – SQL com seleção para o nome Chico
Fonte: Elaborada pelo autor.

```
select Mv_Soundex('xico') from
rdb$database
```

Listagem de Código-Fonte 23 – SQL com seleção para o nome Xico
Fonte: Elaborada pelo autor.

Quadro 32 – Resultado fonético para o nome Chico

NOME	CAMPO_FONETICO
Chico	C200

Fonte: Elaborada pelo autor.

Quadro 33 – Resultado fonético para o nome Xico

NOME	CAMPO_FONETICO
Xico	X200

Fonte: Elaborado pelo autor.

É possível perceber que a variação numérica do campo fonético é calculada corretamente de acordo com as premisses do Soundex, porém, para a Língua Portuguesa, manter a primeira letra para uma analise fonética não é uma estratégia confiável, pois ocorrerão os problemas informados.

6 BUSCASONORA: PROPOSTA DE SOLUÇÃO PARA A BUSCA FONETICA NA LÍNGUA PORTUGUESA

Para desenvolver um algoritmo que possa abranger todas as regras gramaticais ou conseguir pelo menos um resultado mais favorável quanto à busca fonética na Língua Portuguesa, o estudo teve que ser minucioso, tendo em vista que as possibilidades para a criação de nomes são extensas.

Todos os algoritmos analisados apresentaram diversos problemas, que já foram relatados anteriormente, o algoritmo que se mostrou mais próximo de uma solução ideal, foi publicado na revista Clube Delphi, 82, em 2006 (BUBLITZ, 2006). Portando, esta solução foi escolhida para as correções e novas implementações para uma busca fonética mais completa no idioma Português.

Inicialmente concentrou-se em ajustar problemas encontrados no algoritmo publicado fazendo novas implementações. Analisando o código fonte original, as mudanças que ocorreram para melhoria da função, foram referente às regras fonéticas, ou seja, às analises que o código faz para encontrar o fonema correspondente a letra.

Pode-se descrever o processo de desenvolvimento em três fases. A primeira fase é a criação da PL/SQL baseado no algoritmo original com as correções simples para as falhas encontradas no mesmo. A PL/SQL são blocos que aceitam parâmetros de entrada e saída, retornam resultados e são instruções compiladas em um único plano de execução, ou seja, são procedimentos programados no banco de dados, tendo em vista que a PL/SQL é uma extensão a linguagem SQL (GOYA, 2011).

A justificativa para a implementação em PL/SQL é para o funcionamento em diversos SGBD, principalmente, Firebird, MySQL e PostgreSQL, tendo em vista que a PL/SQL é uma extensão da SQL. Apesar de haver diferenças de escrita de comandos nas *Storeds Procedures* de um SGBD para o outro, a adequação é perfeitamente possível, bastando trocar comandos que são específicos de um SGBD por outro equivalente.

Outro fato que influenciou o desenvolvimento em PL/SQL são questões de Sistemas Operacionais. O código publicado por (BUBLITZ, 2006) explica a criação de uma DLL, porém arquivos com esta extensão não funcionaria em outros sistemas operacionais, como por exemplo, no Linux. Havendo um problema para futuras implementações, pois no Linux são utilizados arquivo com extensão “.SO”.

Na segunda fase, foi realizado um extenso estudo dos fonemas “S”, “X” e “Z” que são os mais complicados de tratar, pois possuem muitas formas de escrita diferentes. Para este caso desenvolve-se um novo algoritmo que é explicado mais a frente no trabalho.

Na terceira fase, foi desenvolvido um sistema de cadastro e consulta ao banco de dados, para comprovação do funcionamento do algoritmo e também para que seja possível a visualização na prática do que foi elaborado durante o projeto. O sistema foi desenvolvido em Object Pascal, utilizando-se o ambiente de desenvolvimento Delphi, um dos ambientes de desenvolvimento mais usados no mundo (EVARISTO, 2004).

6.1 PRIMEIRA FASE – CORREÇÃO DE FALHAS DO ALGORITMO

Tendo em vista que Bublitz (2006) preocupou-se em resolver questões quanto ao fonema da letra Z e S no final da palavra, como nos nomes Luiz e Luis, deve-se tratar também as letras N e M, como nos nomes Nilton e Niltom. Portanto, foi incluída o seguinte trecho de código, de acordo com a Listagem de Código-Fonte 24, para resolver este problema.

```
'N':
        if (i = Length(aux)) or (aux[i+1] = ' ') then
        novo := novo + 'M'
        else
        novo := novo + 'N';
```

Listagem de Código-Fonte 24 – Algoritmo incluído para tratamento da letra N e M no final da palavra.
Fonte: Elaborada pelo autor.

Essa parte de código determina que, se no final da palavra ou depois do nome houver um espaço vazio, por exemplo, Nilton Silva, e a última letra for N, o sistema entenderá como a letra M. Em contrapartida, se a última letra for M, será mantida a letra M como campo fonético. Há, portanto, um embasamento lógico para que o sistema encontre nomes com escrita diferente, porém com a fonética igual.

Quando executado o código fonte original, é possível perceber falhas ao se realizarem buscas com os nomes Luiz e Luis. O sistema retorna dois campos fonéticos distintos, gerando, portanto, um problema para ser solucionado. Inicialmente, Bublitz (2006) criou a seguinte função, de acordo com a Listagem de Código-Fonte 25:

```
'Z':
        if (i = Length(aux)) or (aux[i+1] = ' ') then
        novo := novo + 'S'
        else
        novo := novo + 'Z';
```

Listagem de Código-Fonte 25 – Função criada para tratar a letra Z no final da palavra
Fonte: BUBLITZ, 2006.

Para correção do problema, foi necessária a inclusão de um tratamento na letra S, tratamento pelo qual realiza a verificação se a letra “S” está também no final da palavra, caso esteja, mantêm a letra “S”, resolvendo o problema e mantendo similaridade fonética nas buscas..

```
Else
       if (i = Length(aux)) or (aux[i+1] = ' ') then
        novo := novo + 'S' ;
```

Listagem de Código-Fonte 26 – Algoritmo incluído para tratamento da letra S no final da palavra
Fonte: Elaborada pelo autor.

A criação dessa nova função foi gerada após testes em que o sistema estava gerando resultados distintos, onde o tratamento para o S no final da palavra não havia sido incluído no algoritmo.

Quadro 34 – Resultado de busca com o nome Luiz no algoritmo de BUBLITZ, 2006

NOME	CAMPO_FONETICO
LUIZ	LS

Fonte: Elaborada pelo autor.

Quadro 35 – Resultado de busca com o nome Luis no algoritmo de BUBLITZ, 2006

NOME	CAMPO_FONETICO
LUIS	L

Fonte: Elaborada pelo autor.

O código de Bublitz (2006) falha também é realizada busca da letra G, pois somente é tratada a vogal E depois da consoante G, tendo em vista que pode haver ter similaridade fonética quando, após a consoante G, houver a vogal I. É possível citar como exemplo os nomes Gisele ou Jisele.

```
Else
        if aux[i + 1] = 'I' then
        novo := novo + 'J'
        else
        novo := novo + 'G';
```

Listagem de Código-Fonte 27 – Algoritmo incluído para tratamento da letra I depois da letra G
Fonte: Elaborada pelo autor.

Quando o estudo parte para a letra W, o sistema trata apenas se estiver a letra W transformando em V, como nos nomes Walter e Valter. Mas é possível ter exemplos em que, no início da palavra, o W tem som de U, ou seja, nos nomes Walace e Ualace. Foi incluído também um tratamento para vogais antes da letra U terem som de L, como nos nomes Algusto e Augusto.

```
 'U':
       if (i = 1) and (aux[i] = 'U') then
                novo := novo + 'W'
        else if aux[i - 1] in ['A','E','I','O','U'] then
          novo := novo + 'L';
```

Listagem de Código-Fonte 28 – Algoritmo incluído para tratamento das vogais A, E, I, O, U antes da letra U depois da letra G

Fonte: Elaborada pelo autor.

Foi realizada uma inclusão para o tratamento da letra “X” no final da palavra, onde há possibilidades de fonética “KS”, ou seja, em nomes como Max ou Maks. A solução para o problema pode ser realizado com um rastreio na palavra e verificação se a letra “X” está no final da palavra, caso esteja, acrescenta como fonema as letras “KS”. Ocorre também possibilidade de fonema parecido quando a letra “C” é acompanha da letra “S”. De acordo com a Listagem de Código Fonte 29 é descrito como foi resolvido o problema:

```
else if (:STR1 = 'X') then
      begin
        if ((:I = char_length(:NOME)) or (:STR2 = ' ')) then
         NOVO = :NOVO || + 'KS';
        else
          NOVO = :NOVO || + 'X';
      End
-----------------------------------------------------------------
else if (:STR1 = 'S') then
begin
else if (:STRANT = 'C') then
          NOVO = NOVO || 'KS';
End
```

Listagem de Código-Fonte 29 – Algoritmo incluído para tratamento da letra “X” no final da palavra e a letra “C” antes da letra S”.

Fonte: Elaborado pelo autor.

O algoritmo original descartava a fonética da letra “C” quando era iniciada na palavra, havendo um tratamento apenas quando a letra “C” era acompanha das vogais (“A”, “O” ou “U”) incluindo um fonema “K” e quando a letra “C” era acompanha da letra “H”, atribuindo um fonema “X”. Porém, analisando as possibilidades de criação de nomes, ocorrem casos em que a letra “C” tem som de “K” sem ser acompanhada de vogais. É possível citar como exemplo os nomes Claus e Klaus ou nos nomes Cristiana e Kristiana. Portanto, para solução do problema, foi incluído no código um tratamento em que se analisa as letras “L” e “R” depois da letra “C”.

Houve uma preocupação também quanto as vogais “E” e “I” depois da letra “C”, onde em exemplos simples é possível descrever um fonema “S”, havendo portanto a necessidade de mudança no código para melhorar a busca fonética para a Língua Portuguesa. Os exemplos são os nomes Cintia e Sintia. A Listagem de Código Fonte 30 descrevera as estratégias para correções dos problemas descritos:

```
 else if (:STR1 = 'C') then
      begin
        if (:STR2 = 'H') then
          NOVO = :NOVO || 'X';
        else if (:STR2 in ('A', 'O', 'U')) then
          NOVO = :NOVO || 'K';
        else if (:STRANT = 'X') then
          NOVO = :NOVO || + 'S';
        else if (:STR2 in ('E', 'I')) then
          NOVO = :NOVO || 'S';
        else if (:STR2 in ('R', 'L')) then
          NOVO = :NOVO || 'K';
      End
```

Listagem de Código-Fonte 30 – Algoritmo incluído para tratamento da letra “C”.

Fonte: Elaborado pelo autor.

6.2 SEGUNDA FASE – TRATAMENTO PARA LETRAS S, X E Z

O tratamento para as letras “S”, “X”, e “Z” devem ser realizados e estudados de forma mais especifica, pois as possibilidades de diferentes formas para se escrever palavras que contêm essas letras são diversas. Mediante essas variações, que surgiu a possibilidade de criar soluções diferentes para encontrar um resultado mais satisfatório no que diz respeito a realidade de pronuncia e escrita existente atualmente no Brasil.

Quando realizado o estudo dos fonemas existentes, surgiu a oportunidade de estudar os grafemas, que é definido por (SCLIAR, 2003) como uma ou duas letras que representam um fonema. Ou seja, nos sistemas de escrita, os fonemas são representados por grafemas, onde se denomina grafema-fonema (REGINA, 2009). Na Figura 3 é mostrado um exemplo:

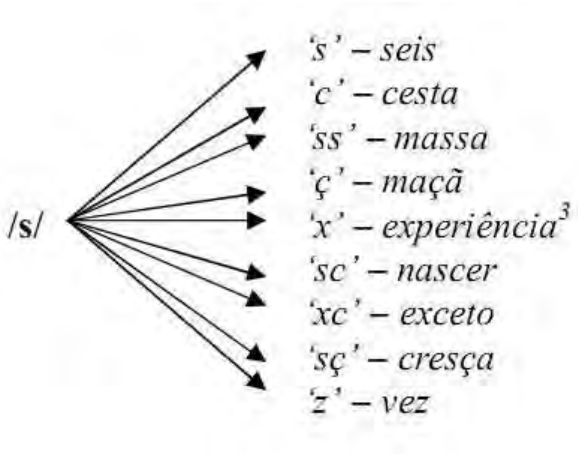

Figura 3: Exemplo de novo fonema para a letra S de acordo com a palavra ‘vez’.
Fonte: GARCIA et all, 2011

Como descrito na Figura 3, o fonema é a letra “S”, porém, o grafema é a forma com que a palavra é escrita. No inicio do projeto, de acordo com o Quadro 1 – Fonemas vocálicos e fonemas consonantais, foi descrito os fonemas da Língua Portuguesa, sua representação e o exemplo para cada tipo de fonema. Após estudos para criação do algoritmo, foi necessária uma atenção especial para a letra “S”, pois suas variações são diversas, ou seja, temos diversos grafemas e diversos fonemas para suas representações. Inicialmente, incluímos 8 grafemas para o

estudo da letra “S”, mas segundo (GARCIA et all, 2011) a letra “S” contém 9 fonemas, incluindo a letra “Z” na palavra Vez, por exemplo.

Mediante as dificuldades para resolver a fonética da letra “S”, encontra-se exemplos nos quais é possível descrever a palavra “ROÇA”, onde facilmente é possível realizar uma busca digitando a palavra “ROSA” ou “ROSSA”. Levando em consideração possibilidades de erros em variações gramaticais ou dificuldades em escrita. Porém, a palavra “ROSA” (nome próprio) contém a fonética igual à letra “Z”, havendo um novo problema para a busca fonética. Se o usuário deseja encontrar a palavra “ROÇA”, não seria necessário mostrar um nome cadastrado como “ROZA”.

Foi realizada a inclusão de novos tratamentos para a letra “S”, a fim de ajustar o seu fonema para alguns grafemas, que são eles: “XC”, “SC” e “SÇ”. Para solução do problema, veja a Listagem de Código Fonte 31:

```
if (:STR1 = 'S') then
else if (:STR2 = 'Ç') then
          NOVO = :NOVO || 'S';
```

```
if (:STR1 = 'S') then
else if (:STR2 = 'C') then
          NOVO = NOVO || 'S';
```

```
if (:STR1 = 'C') then
else if (:STRANT = 'X') then
          NOVO = :NOVO || + 'S';
```

Listagem de Código-Fonte 31 – Algoritmo incluído para tratamento para os grafemas da letra “S”.
Fonte: Elaborado pelo autor.

Após analisar o fonema “S”, deve-se atenção especial a letra “Z”, que também contem variações e possibilidades diferentes tanto para fala quanto para erros em escrita. Como descrito na Figura 4, o fonema é a letra “Z”, e seus respectivos grafemas são a própria letra “Z” como na palavra zebra, a letra “S” como na palavra casamento e na letra “X” de acordo com a palavra exato.

Inicialmente, o algoritmo original tratou a letra “Z” caso estivesse entre as vogais “A, “E”, “I”, “O”, “U”. Mas se for analisar as possibilidades como um todo, o tratamento tem que ser mais criterioso, para obtenções de soluções mais próximas e satisfatórias para o controle de busca.

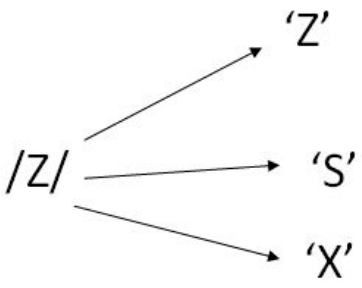

Figura 4: Grafemas da letra Z.
Fonte: Elaborado pelo Autor.

Conforme descrito no início do capítulo, além das letras “S” e “Z”, os estudos e pesquisas durante a realização do trabalho, constatou que a letra “X” contém diversas possibilidades de grafemas, onde é possível concluir que tratamentos especiais para este fonema foi desenvolvido, afim de soluções eficazes para a busca fonética na Língua Portuguesa.

De acordo com a Figura 5, é possível visualizar que o fonema “X” contém os grafemas “Z” de acordo com a palavra exagero, na própria letra “X” e no grafema “KS” como na palavra toxico.

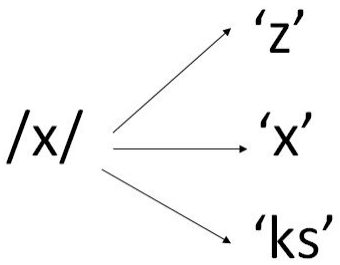

Figura 5: Grafemas da letra X.
Fonte: Elaborado pelo Autor.

A solução sugerida pelo orientador do projeto foi a de buscar todos os grafemas possíveis para um mesmo fonema levando em consideração que não é possível definir, através da escrita, qual é o fonema desejado pelo usuário. Ou seja, se é digitado a palavra “ROSA”, não é possível definir se o usuário está indicando o nome o fonema do “Z”, obtendo a palavra “ROZA”, ou, o fonema do “S”, obtendo a palavra “ROSSA” e “ROÇA”.

Assim, caso o usuário digite a palavra “ROSA” é necessário retornar todos os grafemas possíveis para todos os fonemas possíveis. Ou seja, retornaria as palavras, “ROZA”, “ROSA”, “ROSSA” e “ROÇA”. Porém se o usuário informa a escrever a palavra “ROZA”, é obvio que não se trata do fonema “S” e sim “Z”, neste caso, o retorno deveria ser apenas as palavras “ROZA” e “ROSA”.

Caso digite a palavra “EXAME” seria necessário retornar palavras como “EXAME”, “EKSAME”, “EZAME”. Apesar da palavra “EXAME” não ser um nome próprio, o sistema abre precedentes para buscas não só de nomes, mas também de qualquer palavra em um texto, havendo possibilidade de ser usado em um editor de textos, onde não seria necessariamente obrigatório digitar a palavra como ela está no texto.

Para desenvolver esta solução, o orientador do projeto propôs um algoritmo que realiza trocas nas letras S, Z e X, da string fonética, por caracteres especiais e depois substitui todos estes caracteres por todas as possibilidades de grafemas para os possíveis fonemas da escrita, gerando diversos novos códigos fonéticos que serão usados para efetuar as consultas no banco de dados. Abaixo segue um exemplo do algoritmo.

Os caracteres especiais escolhidos para substituição dos possíveis grafemas são (*, & e $), sendo atribuídos de acordo com o Quadro 36 abaixo:

Quadro 36 – Atribuição de Caracteres especiais para os fonemas X, Z e S.

Caractere Especial	Fonema	Possíveis Grafemas
*	X	X, Z e KS
&	Z	S, Z e X
$	S	S e Z

Fonte: Elaborada pelo autor.

Portanto, quando o algoritmo é executado recebendo como parâmetro a palavra “EXAME”, é gerado todas as possibilidades de fonemas para esta palavra levando em consideração os grafemas existentes, ou seja, de acordo com o Quadro 37 abaixo, o algoritmo gera como possibilidades os seguintes campos fonéticos (KSM, ZM, XM).

Quadro 37 – Resultados de todos os possíveis grafemas para a palavra EXAME.

Palavra	Fonema	Resultados Possíveis
EXAME	X	KSM, ZM e XM

Fonte: Elaborada pelo autor.

O algoritmo que realiza as criações das possibilidades de grafemas para uma palavra foi desenvolvido no sistema de busca, sendo acoplado a consulta SQL para melhor filtrar e encontrar o que necessita, mesmo sendo digitado incorretamente a palavra, havendo um avanço proporcional no que diz respeito a melhoria do software.

É possível visualizar de acordo com a Listagem de Código Fonte 32, as funções que o sistema realiza para criação das variações:

```
function TFConsultaCliente.GrafemasPossiveis(Palavra: string): TStringList;
var
  ListaPalavras: TStringList;
  ListaPalavrasAux: TStringList;
  PalavraCodFonetico: String;
  I: Integer;
begin

  // Passo 1 - Declaração de variáveis
  ListaPalavras := TStringList.Create;
  ListaPalavrasAux := TStringList.Create;

  // Passo 2 - Passando a palavra para código fonetico
  PalavraCodFonetico := PegaCodigoFonetico(Palavra);

  // Passo 3 - Fazendo substituição de X, Z e S para caracteres especiais
  PalavraCodFonetico := StringReplace(PalavraCodFonetico, 'KS', '*', [rfReplaceAll, rfIgnoreCase]);
  PalavraCodFonetico := StringReplace(PalavraCodFonetico, 'X', '*', [rfReplaceAll, rfIgnoreCase]);
  PalavraCodFonetico := StringReplace(PalavraCodFonetico, 'Z', '&',
```

```
[rfReplaceAll, rfIgnoreCase]);
  PalavraCodFonetico := StringReplace(PalavraCodFonetico, 'S', '$',
[rfReplaceAll, rfIgnoreCase]);
  // Passo 4 - Substituindo caracteres especiais pelos possíveis
grafemas
  I := 0;
  ListaPalavras.Add(PalavraCodFonetico);
  if (Pos('*', ListaPalavras.Strings[I]) > 0) then
    begin
      ListaPalavrasAux.Add(StringReplace(ListaPalavras.Strings[I],
'*', 'KS', [rfReplaceAll, rfIgnoreCase]));
      ListaPalavrasAux.Add(StringReplace(ListaPalavras.Strings[I],
'*', 'Z', [rfReplaceAll, rfIgnoreCase]));
      ListaPalavrasAux.Add(StringReplace(ListaPalavras.Strings[I],
'*', 'X', [rfReplaceAll, rfIgnoreCase]));
    end;

  if (ListaPalavrasAux.Count > 0) then
    begin
      ListaPalavras.Clear;
      ListaPalavras.AddStrings(ListaPalavrasAux);
      ListaPalavrasAux.Clear;
    end;

  for I := 0 to ListaPalavras.Count -1 do
    begin
      if (Pos('&', ListaPalavras.Strings[I]) > 0) then
        begin

ListaPalavrasAux.Add(StringReplace(ListaPalavras.Strings[I],    '&',
'S', [rfReplaceAll, rfIgnoreCase]));

ListaPalavrasAux.Add(StringReplace(ListaPalavras.Strings[I],    '&',
'Z', [rfReplaceAll, rfIgnoreCase]));

ListaPalavrasAux.Add(StringReplace(ListaPalavras.Strings[I],    '&',
'X', [rfReplaceAll, rfIgnoreCase]));
```

```
        end;
    end;

  if (ListaPalavrasAux.Count > 0) then
    begin
      ListaPalavras.Clear;
      ListaPalavras.AddStrings(ListaPalavrasAux);
      ListaPalavrasAux.Clear;
    end;

  for I := 0 to ListaPalavras.Count -1 do
    begin
      if (Pos('$', ListaPalavras.Strings[I]) > 0) then
        begin

ListaPalavrasAux.Add(StringReplace(ListaPalavras.Strings[I],    '$',
'S', [rfReplaceAll, rfIgnoreCase]));

ListaPalavrasAux.Add(StringReplace(ListaPalavras.Strings[I],    '$',
'Z', [rfReplaceAll, rfIgnoreCase]));
        end;
    end;

  if (ListaPalavrasAux.Count > 0) then
    begin
      ListaPalavras.Clear;
      ListaPalavras.AddStrings(ListaPalavrasAux);
      ListaPalavrasAux.Clear;
    end;

  // Passo 5 - Retornando Possíveis Códigos Fóneticos (de acordo com
os grafemas)
  Result := ListaPalavras;
end;
```

Listagem de Código-Fonte 32 – Algoritmo que Realiza as Variações de Acordo com os Grafemas das Palavras.

Fonte: Elaborado pelo autor.

O algoritmo que realiza as consultas SQL para palavras que contém letras “S” ou “Z” foi denominado especial, pois os tratamentos para geração de um resultado satisfatório são diferenciados das outras palavras que não contém tais variações. É possível visualizar de acordo com a Listagem de Código Fonte 33 a parte de código que efetua esta função:

```
procedure TFConsultaCliente.BuscaFoneticaEspecial;
var
  CodFonetico: TStringList;
  ListaWhere: TStringList;
  ListaCodigosFoneticos: TStringList;
  CondicaoLimiteFonemaZ: string;
  Palavra: string;
  FonemaSouZ: string;
  Operador: String;
  I: Integer;

  function PalavraRegraSouZ(Palavra: string): string;
  var
    I: Integer;
  begin
    Result := '';
    for I := 2 to Pred(Length(Palavra)) do
      begin
        if (Palavra[I] = 'Z') then
          Result := 'Z'
        else if (Palavra[I] = 'S') then
        begin
                if (Palavra[I + 1] <> 'S') and (Palavra[I + 1] <>
'Ç') and (Palavra[I + 1] <> 'C') then
           Result := 'SZ'
          else
           Result := 'S';
        end;
```

```
        if (Result = 'S') or (Result = 'Z') or (Result = 'SZ') then
           break;

      end;
  end;

begin
  // Declaração de variaveis
  CodFonetico := TStringList.Create;
  ListaWhere := TStringList.Create;
  ListaCodigosFoneticos := TStringList.Create;
  Palavra := UpperCase(Trim(Edit1.Text));

  ListaCodigosFoneticos.AddStrings(GrafemasPossiveis(Edit1.Text));
  FonemaSouZ := PalavraRegraSouZ(Palavra);
  if (FonemaSouZ = 'Z') then
     CondicaoLimiteFonemaZ := ' and (CLIENTE.TEM_S_OU_Z = ''1'')'
  else
    CondicaoLimiteFonemaZ := '';

  //
  for I := 0 to ListaCodigosFoneticos.Count - 1 do
    begin
     if (I=0) then
          Operador := ' where '
      else
          Operador := ' or ';

      if    (CondicaoLimiteFonemaZ    <>    '')    and    (Pos('S',
ListaCodigosFoneticos[I]) > 0)  then
          ListaWhere.Add( Operador + '(CLIENTE.CAMPO_FONETICO = ' +
QuotedStr(ListaCodigosFoneticos[I]) + CondicaoLimiteFonemaZ + ')')
      else
          ListaWhere.Add( Operador + '(CLIENTE.CAMPO_FONETICO = ' +
QuotedStr(ListaCodigosFoneticos[I]) + ')');
    end;
```

```
  IBQuery1.SQL.Clear;
  IBQuery1.SQL.Text := 'select                          ' +
    '   CLIENTE.NOME_CLIENTE,      ' +
    '   CLIENTE.CAMPO_FONETICO,    ' +
    '   CLIENTE.TEM_S_OU_Z         ' +
    'from CLIENTE                  ';

  for I := 0 to ListaWhere.Count - 1 do
     IBQuery1.SQL.Text := IBQuery1.SQL.Text + ListaWhere[I];

  IBQuery1.SQL.Text    :=    IBQuery1.SQL.Text    +    'order    by
CLIENTE.NOME_CLIENTE ';

  try
    //ShowMessage(IBQuery1.SQL.Text);
    IBQuery1.Close;
    IBQuery1.Open;
  except
    on E: Exception do
      ShowMessage(E.Message);
  end;
end;
```

Listagem de Código-Fonte 33 – Algoritmo que Realiza busca fonética especial para palavras que contém letras S e Z.

Fonte: Elaborado pelo autor.

Uma outra regra necessária, para solução do problema foi a inclusão de um novo campo na tabela CLIENTE, tabela pela qual foi usada para os testes e comprovação dos resultados apresentados no trabalho. Este novo campo foi criado com o nome de “TEM_S_OU_Z”, onde recebe como parâmetro o número 1 ou o número 0, estes números são padronizados através de um procedimento criado no banco de dados, chamado “CONTEMSOUZ”.

Esse procedimento visa indicar se a palavra original cadastrada no banco de dados segue o fonema “Z” ou “S”, que será utilizado para distinguir quando é necessário realizar as buscas pelo fonema do “Z” ou pelo fonema do “S”. Assim o valor 1 é gravado quando a palavra corresponde somente ao fonema do “S” e zero quando a palavra pode ter a possibilidade de fonemas “S” e “Z”.

Este campo será utilizado somente quando se tratar de analise de palavras com “S” e “Z”, quando a palavra não possuir uma dessas letras é gravado zero, mas o campo não é utilizado para retorno dos valores. É possível visualizar o código do procedimento criado na Listagem de código Fonte 33 abaixo:

```
CREATE PROCEDURE CONTEMSOUZ (
    nome varchar(255))
returns (
    temsouz varchar(1))
as
declare variable strant varchar(1);
declare variable aux varchar(1);
declare variable j integer;
declare variable str2 varchar(1);
declare variable tamstring varchar(255);
begin
  J = 2;
  temsouz = '0';
  nome = upper(:nome);
  TamString = (char_length(:NOME) -1 );
  while (J <= TamString) do
  begin
      strant = substring (:NOME FROM :J-1 FOR 1);
      STR2 = substring(:NOME from :J + 1 for 1);
      AUX = substring (:NOME FROM :J FOR 1);
      if (:AUX = 'Z') then
      begin
          temsouz = '1';
          break;
      end
      else if (((strant <> 'S') and (:aux = 'S') and (:str2 <> 'S'))
and ((:aux = 'S') and (:str2 <> 'Ç')) and ((:aux = 'S') and (:str2
<> 'C'))) then
      begin
         temsouz = '1';
          break ;
```

```
      end
      J = :J + 1;
  end
  suspend;
end
```

Listagem de Código-Fonte 34 – Procedimento chamado CONTEMSOUZ.

Fonte: Elaborado pelo autor.

Após a criação desta regra, é possível verificar com exemplos a divisão que o sistema realiza quanto às palavras que contém diversas possibilidades de pronuncia. Realizando um filtro mais preciso para as consultas no sistema de busca fonética.

Quadro 38: Demonstração do novo parâmetro TEM_S_OU_Z criado.

NOME_CLIENTE	CAMPO_FONETICO	TEM_S_OU_Z
ROÇA	RS	0
ROSA	RS	1
ROZA	RZ	1

Fonte: Elaborado pelo Autor.

Quanto a análises de palavras que são escritas com a letra “X” ou “Z”, é possível verificar que o sistema realiza o tratamento corretamente e gerando seus respectivos campos fonéticos de acordo com os tratamentos explicados durante o trabalho.

Quadro 39: Atribuição de números 1 ou 0 no parâmetro TEM_S_OU_Z criado.

NOME_CLIENTE	CAMPO_FONETICO	TEM_S_OU_Z
EZAME	ZM	1
EXAME	XM	0
TOXICO	TXK	0
TOZICO	TZK	1
TOKSICO	TKSK	1

Fonte: Elaborado pelo Autor.

Neste sentido, o principal é verificar o campo TEM_S_OU_Z, que de acordo com sua importância para o filtro na consulta SQL, tornou-se um diferencial para a divisão de palavras que serão encontradas de acordo com o fonema e os possíveis grafemas, não sendo mostrados resultados incompatíveis.

6.3 TERCEIRA FASE – DESENVOLVIMENTO DA INTERFACE

Na terceira fase, o objetivo foi desenvolver um sistema no qual fosse possível cadastrar e consultar diretamente no banco de dados os nomes de acordo com sua fonética e não como a palavra é escrita.

O sistema desenvolvido abrange, portanto, uma tela de cadastro, quando este é executado, automaticamente seu código fonético é calculado e mostrado na tela para análise (Figura 6).

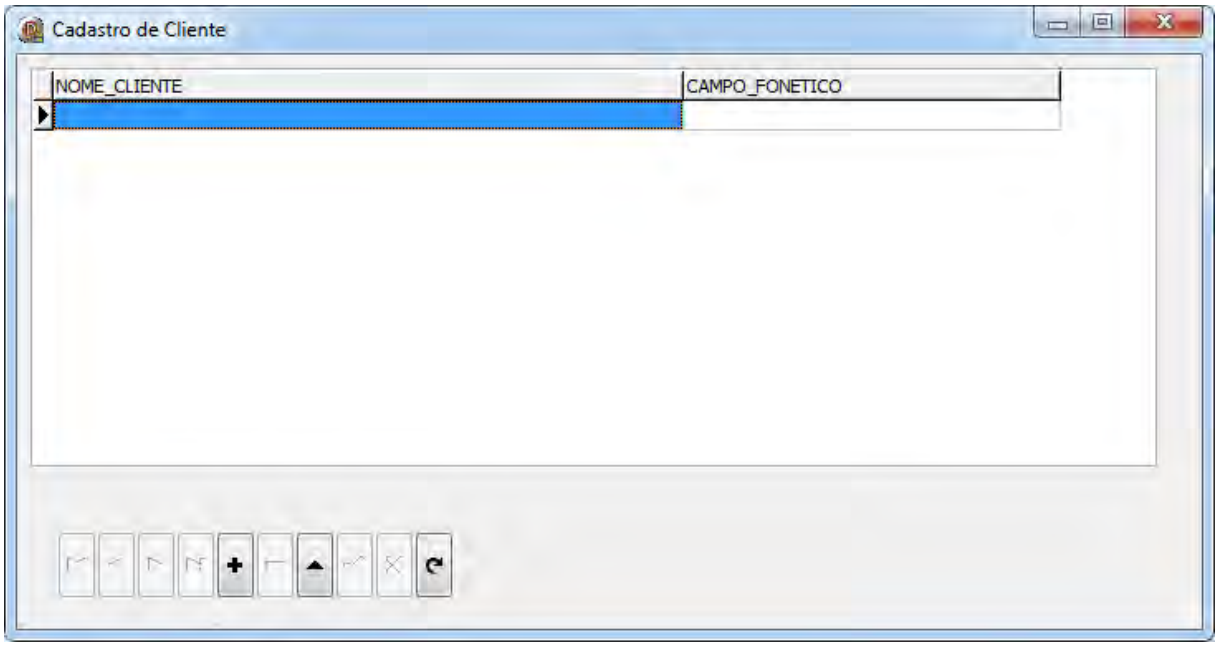

Figura 6 – Tela para cadastro no sistema de busca fonética
Fonte: Elaborada pelo autor.

O processo de consulta é simples. Apresenta-se uma tela onde o nome terá que ser preenchido; ao clicar em consultar, os nomes que contêm os campos_fonéticos iguais serão apresentados de acordo com o que foi pedido na consulta. Para exemplificar melhor, a Figura 7 mostra com detalhes um processo de consulta.

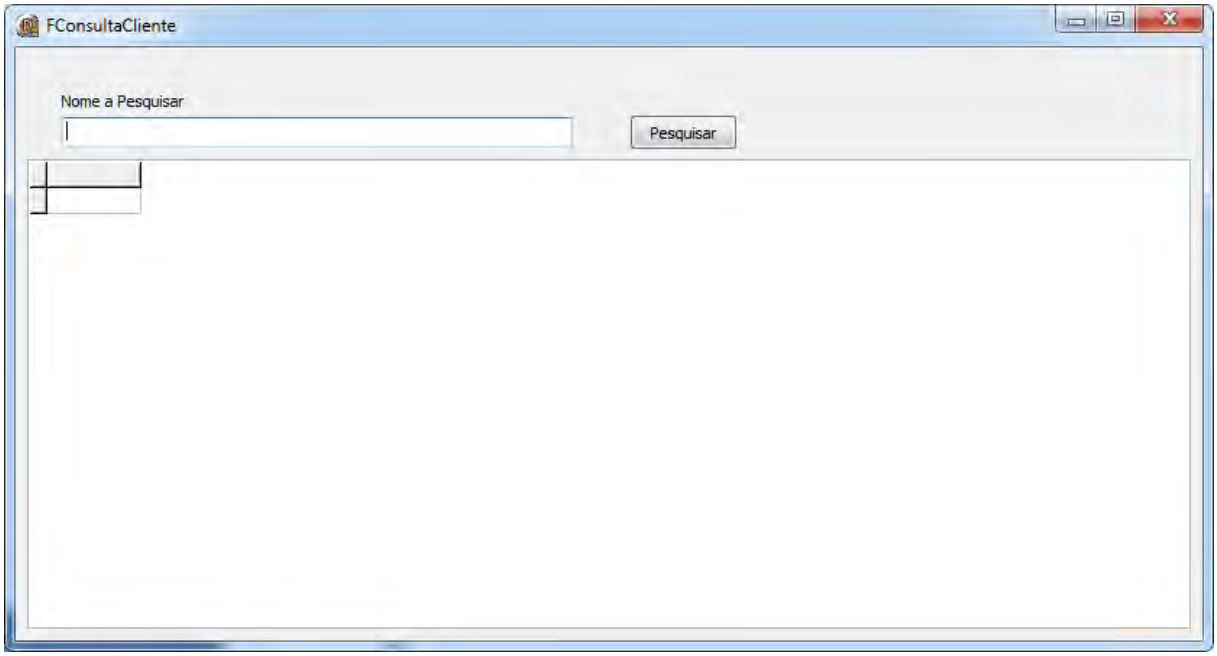

Figura 7 – Tela para consultar nomes inseridos no sistema de busca fonética
Fonte: Elaborada pelo autor.

7 ESTUDO DE CASO

Após realizar uma análise de cenário sobre as funcionalidades do sistema desenvolvido para consulta através de busca fonética, é possível obtenção de uma série de nomes que podem ser usados e avaliados quanto ao sucesso que o algoritmo proposto realiza no que diz respeito a regras para obtenção de um campo fonético igual para palavras que contem possibilidades de escritas diferentes.

No Quadro 40 é possível visualizar de acordo com a coluna campo_fonético, que o sistema estipula resultados sucintos e que são compatíveis com as diversas formas de cadastrar nomes em bancos de dados.

Quadro 40: Lista de nomes cadastros no sistema de busca fonética proposto no trabalho.

NOME_CLIENTE	CAMPO_FONETICO
CARLA	KRL
CARLA	KRL
CHICO	XK
XICO	XK
FELIPE	FLP
PHELIPE	FLP
FATHIMA	FTM
FATIMA	FTM
CLAUS	KLLS
KLAUS	KLLS
MAX	MKS
MAKS	MKS
LUIZ	LS
LUIS	LS
NILTON	NLTM
NILTOM	NLTM
CLEUZA	KLLZ
CLEUSA	KLLS
KELLY	KL
QUELI	KL
UALACE	WLS
WALACE	WLS
WALTER	WLTR
VALTER	WLTR
CRISTIANE	KRSTN
KRISTIANE	KRSTN
KRYSTIANY	KRSTN
CELSO	SLS
SELSO	SLS
SELMA	SLM
CELMA	SLM
GISELE	JSL
JISELE	JSL
GESSICA	JSK
JESSICA	JSK

GESSIKA	JSK
JESSICA	JSK
MADRIKSANY	MDRKSN
MADRICSANI	MDRKSN
HEITOR	TR
EITOR	TR
ALGUSTO	LGST
AUGUSTO	LGST
KELI	KL
ASSELINO	SLN
PATRICK	PTRK
ASSUNÇÃO	SNS
ASUNÇÃO	SNS
ASUNSSÃO	SNS
CAIC	KK
KAIK	KK
CAIK	KK
CAIQUE	KK

Fonte: Elaborado pelo Autor.

Utilizando o processo de cadastro do sistema BuscaSonora, é possível visualizar que após cadastrar o nome, o campo fonético é preenchido de acordo com todas as estratégias utilizadas para obtenção do resultado final.

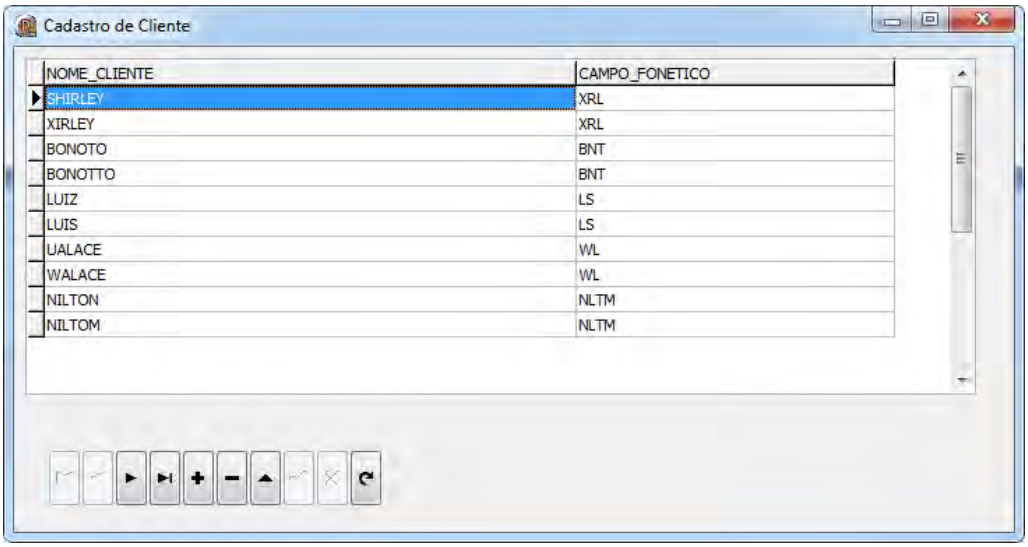

Figura 8 – Visualização de um cadastro no sistema BuscaSonora.
Fonte: Elaborada pelo autor.

E assim que utilizar a consulta disponibilizada no mesmo sistema, é possível obtençao de resultados satisfatórios para nomes que contem o mesmo som, porém, com escrita diferente.

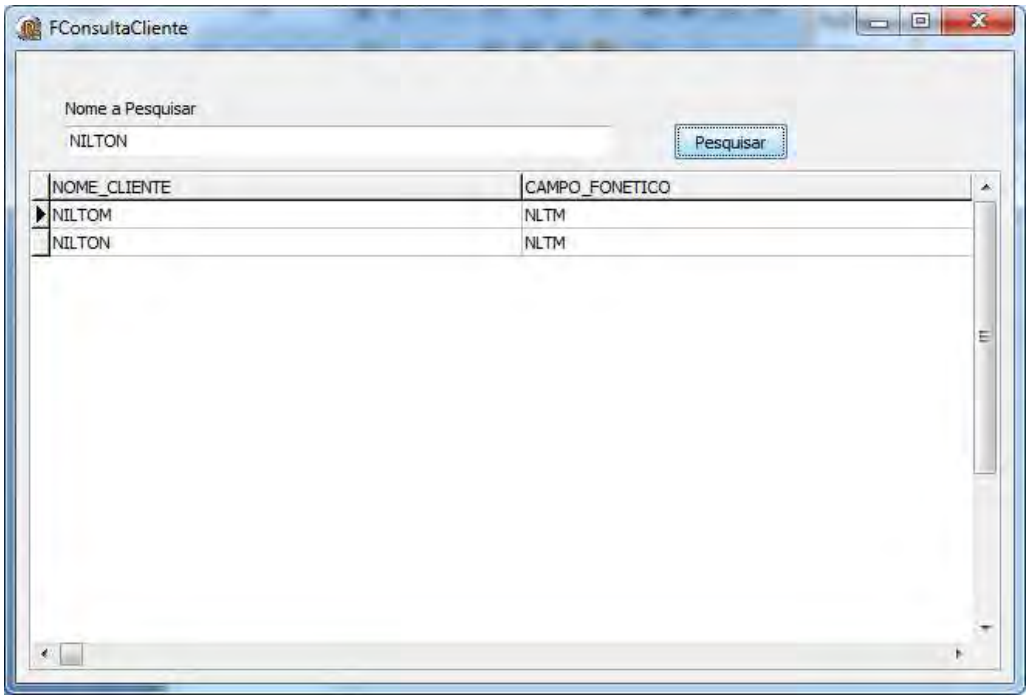

Figura 9 – Consulta no sistema BuscaSonora.
Fonte: Elaborada pelo autor.

8 CONSIDERAÇÕES FINAIS

Durante a realização desse trabalho foram realizados estudos sobre a fonética da Língua Portuguesa em contra partida com a fonética da Língua Inglesa, detalhando suas principais diferenças e focando nas dificuldades das regras ortográficas do Português.

O trabalho permite observar soluções já existentes para a busca fonética nos SGBD Firebird, MySQL e PostgreSQL, havendo limitações para o uso na Língua Portuguesa tanto para soluções nativas aos bancos de dados descritos, tais como soluções alternativas descritas no trabalho.

O trabalho apontou problemas quanto às regras ortográficas que não foram tratadas em funções já existentes para a busca fonética, onde iniciou o processo de desenvolvimento de uma nova solução para a busca fonética mais abrangente. O sistema de busca fonética criado neste trabalho, pode ser implementado tanto no Firebird, quanto no MySQL e PostgreSQl.

Os testes realizados com a nova busca fonética, comprovou eficiência em nomes pelos quais funções já existentes não resolveram o problema, levando em consideração o tratamento da letra “S”, “X” e “Z”. Porém, a criação de nomes e novas possibilidades fonéticas é crescente, sendo necessário contínuos estudos para solução de novos problemas que serão decorrentes dessas criações.

Como sugestão para trabalhos futuros, a criação de procedimentos (procedures) para os SGBD MySQL e PostGreSQL, pois deve-se levar em consideração para a transição dos algoritmos propostos para esses bancos de dados apenas as estruturas e funções que são diferentes para cada SGBD.

REFERÊNCIAS BIBLIOGRÁFICAS

BUBLITZ, J. L. **Código Fonética no Firebird**. Clube Delphi, Ubá, v. 82, 8-11, 2006.

CAGLIARI, Luiz Carlos. **Fonética: uma entrevista com Luiz Carlos Cagliari**. Revista Virtual de Estudos da Linguagem - ReVEL. Vol. 4, n. 7, agosto de 2006).

CURSOECOMMERCE. **Crescimento do E-Commerce gera oportunidades**. Disponível em: < http://www.cursodeecommerce.com.br/blog/crescimento-ecommerce/>. Acesso em: 11 jun. 2011.

PEREIRA. **Os serviços Online do INPI**. Disponível em: < http://www.oa.pt/upl/%7B637b2194-47de-42fd-9660-b840b930f279%7D.pdf >. Acesso em: 21 maio. 2011.

DONALD, E. K. **The art of computer programming**: sorting and searching. Canada: Addison-Wesley Publishing Company, 1973. Vol 3.

EVARISTO, J. **Programando com Pascal**. A Linguagem do Turbo Pascal e do Delphi. Alagoas: Evaristo, 2004. Cap. 1, p. 1-2

FRANTZ, Roni Rui Ruben. J. **Recuperação de Informação por similaridade de fonemas adaptada à Língua Portuguesa**. 2009. 124f. Trabalho de Graduação – Centro Universitário Ritter dos Reis, Porto Alegre, 2009.

GARCIA, Mirian Álvaro Costa,; ARAÚJO, Pâmela Renata Machado; MIRANDA, Ana Ruth Moresco. **Um estudo sobre a grafia do fonema /s/.** UFPEL.

GÁLVEZ, Carmen. Identificación de nombres personales por medio de sistemas de codificacion fonética. **Encontros bibli**: revista eletrônica de biblioteconomia e ciência da informação. florianópolis: Vol. 11, nº 22, 2º semestre 2006.

GOMES, F. Y. Busca fonética no Firebird. DBFree Magazine, Ubá, v. 6, 6-9, 2005.

GOYA, M. **PL/SQL – Procedure e funções**. Disponível em: <http://www.linhadecodigo.com.br/Artigo.aspx?id=335> Acessado em Nov. de 2011.

HELPSAUDE. **Help Saude lança busca fonética de remédios**. Disponível em: <http://www.helpsaude.com/Imprensa/Release/BuscadeRemedios>. Acesso em: 11 jun. 2011.

HJORT, Rodrigo; BORTOLETO, Silvio. **MetaBusca:a** implementação de um algoritmo fonético no PostgreSQL. Companhia de Informática do Paraná. Curitiba – PR, 2007.

INPI. INPI de Portugal licencia ao Brasil sistema de busca fonética. Disponível em: < http://www.inpi.gov.br/noticias/escritorios-brasileiro-e-portugues-criam-bases-de-dados-de-marcas>. Acesso em: 11 jun. 2011.

JOHN REPICE. **Understanding classic Soundex Algorithms**. Disponível em: < http://creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm >. Acesso em: 21 maio. 2011.

PHILLIPS, Lawrence**. The double metaphone search algorithm**. Disponível em: <http://drdobbs.com/184401251?pgno=2> Acessado em nov. de 2011.

POSTGRESQL**. PostgreSQL 8.3.16. Documentation**. Disponível em: < http://www.postgresql.org/docs/8.3/static/fuzzystrmatch.html> Acessado em nov. de 2011.

LUCENA, F. J. T. **Busca Fonética em Português do Brasil**. 2007. 23f. Trabalho de Graduação – Faculdade de Tecnologia IBRATEC (Instituto Brasileiro de Tecnologia), UNIBRATEC, Recife, 2007.

OLIVEIRA, Wolney Resende; MAIA, Diulie Fernandes. **Nota Fiscal Eletrônica: Pojeto Nacional e a Iniciativa Municipal De São PAulo – Uma Análise Comparativa**. Universidade de Brasília. Brasília – DF, 2007.

REGINA, R. L. **Consciência dos sons da língua**: subsídios teóricos e práticos para alfabetizadores, fonoaudiólogos e professores da Língua Inglesa. Brasil: EDIPUCRS, 2009.

ROSSETTO, F. J. **Protótipo de um reconhecimento fonético aplicado ao banco de dados Oracle**. 2000. 85f. Trabalho de Graduação – Universidade Regional de Blumenal, Centro de Ciências Exatas e Naturais, Blumenal, 2000.

SCLIAR- CABRAL, Leonor. Princípios do sistema alfabético do Português do Brasil. São Paulo: Contexto,2003.

schütz**,** ricardo. **diferenças de Pronúncia entre Inglês e Português. English Made in Brazil** <http://www.sk.com.br/sk-pron.html>. Online. 21 de maio de 2011.

THORN VECTOR.**String function**. Disponível em: <http://dev.mysql.com/doc/refman/5.1/en/string-functions.html#function_soundex >. Acesso em: 21 maio. 2011.

UOL. **Dígrafos e fonemas**. Disponível em: <http://educacao.uol.com.br/portugues/ult1693u49.jhtm>. Acesso em: 05 jun. 2011.

VIEIRA, Francisco de Castro. **Língua Portuguesa no Mundo**. 2010. 14f. Trabalho de Graduação – Universidade de Trás-os-Montes e Alto Douro, Vila Real, 2010.

W3. **Extensible Markup Language (XML)**. Disponível em: <http://www.w3.org/XML/> Acessado jul. 2011.

APÊNDICE A
PREPARAÇÃO DO AMBIENTE PARA REALIZAÇÃO DA BUSCA FONÉTICA NO BANCO DE DADOS FIREBIRD

A preparação do ambiente para realização do teste da busca fonética em banco de dados Firebird envolve os seguintes passos: instalação de uma ferramenta gerenciadora de banco de dados e criação de um banco de dados com uma tabela para inserção de palavras. Os próximos parágrafos demonstram a confecção desse ambiente.

Foi utilizado o software IBExpert[4] como ferramenta para acesso e manipulação ao banco de dados.

[4] *Ferramenta disponível para download através do site: http://ibexpert.net/ibe/.*

APÊNDICE B
INSTALAÇÃO DO FIREBIRD

Esta instalação foi realizada em um computador cujo processador é um core 2 duo 2.20 GHz, 3GB de memória. Foi usado um sistema operacional Windows Seven 32 Bits. O início da instalação é mostrado na Figura 8.

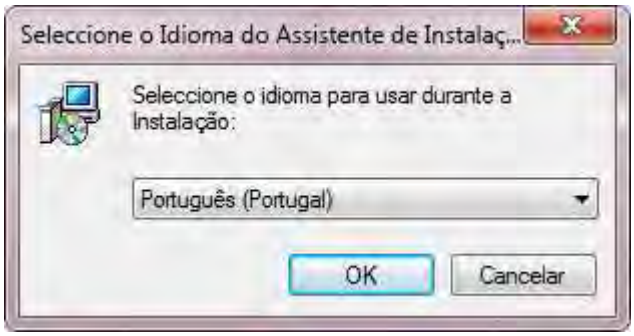

Figura 10 – Tela inicial da instalação do Firebird para seleção do idioma
Fonte: Elaborada pelo autor.

Após a seleção do idioma (Figura 8), será mostrada uma tela de apresentação do Firebird. Essa tela não oferece opções de configuração para a instalação ,e sua finalidade é de cunho informativo.

Após clicar no botão com a descrição “seguinte” na tela de apresentação, é mostrada a tela onde está sendo informado o contrato de licença do Firebird. É necessário estar de acordo com as condições impostas no contrato de licença para continuar a instalação.

No passo seguinte, é necessário escolher o local de instalação do Firebird, como mostrado na Figura 9 abaixo.

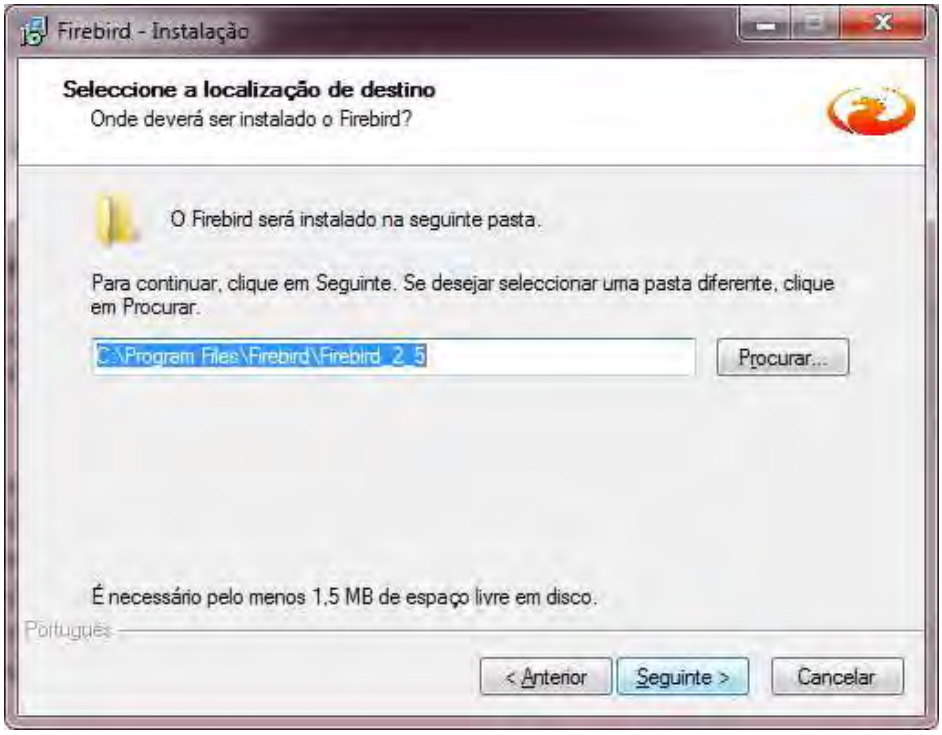

Figura 11 – Tela para configurar em que local deseja que a pasta do Firebird seja criada.
Fonte: Elaborad pelo autor.

Durante o processo de instalação tanto do Firebird versão 2.5 quanto de versões anteriores, é possível configurar em que local será realizada a criação da pasta do Firebird. Esse processo é importante, pois é possível realizar configurações de arquivos e UDFs,s que estão localizadas na pasta de instalação. Como padrão o Firebird realiza a criação da pasta em “C:\Program Files\Firebird\Firebird_2_5”.

Após a escolha do local de instalação do Firebird, é mostrado (Figura 10) um processo importante na instalação, onde é decidido se o Firebird irá funcionar como servidor ou como estação.

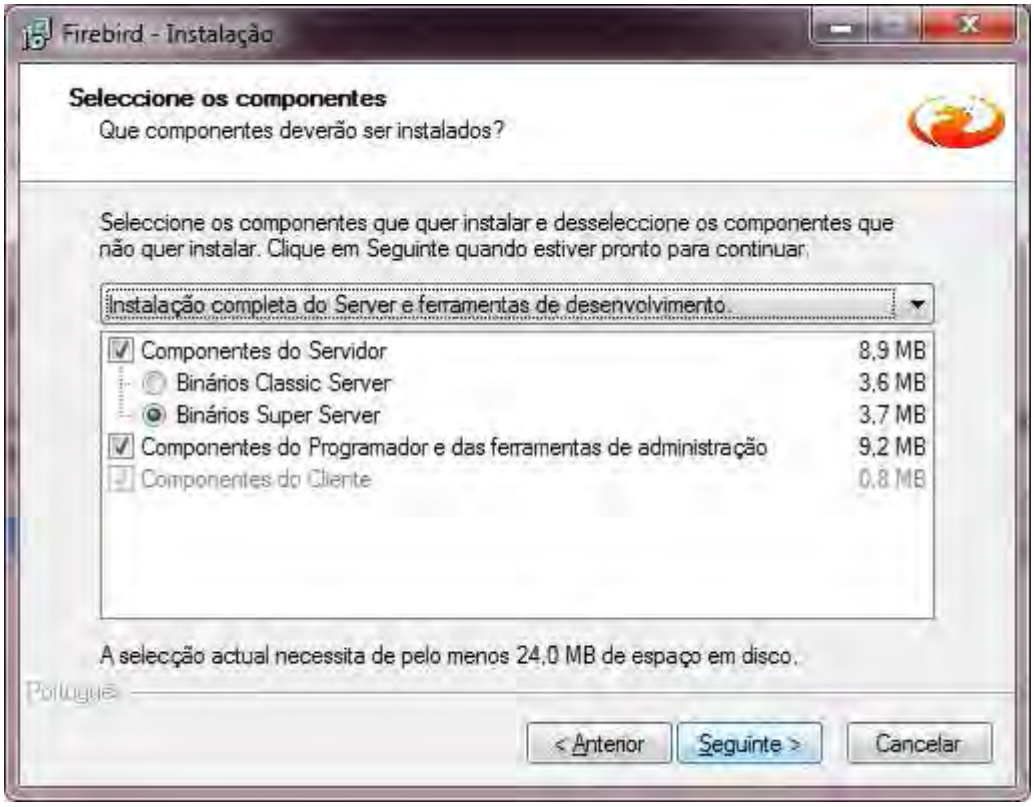

Figura 12 – Tela para configurar instalação como servidor ou como cliente
Fonte: Elaborada pelo autor.

No caso desta pesquisa, a opção adotada foi a instalação para servidor, pois não serão utilizadas estações neste trabalho.

A partir do momento em que é decidido como o Firebird funcionará na máquina, são disponibilizadas algumas informações adicionais, onde é possível que o Firebird seja executado no painel de controle do sistema operacional, seja iniciado sempre que o Windows inicie ou que crie as suas bibliotecas na pasta System de sua máquina. Na instalação realizada, foram escolhidas todas as opções, conforme apresentado na Figura 11.

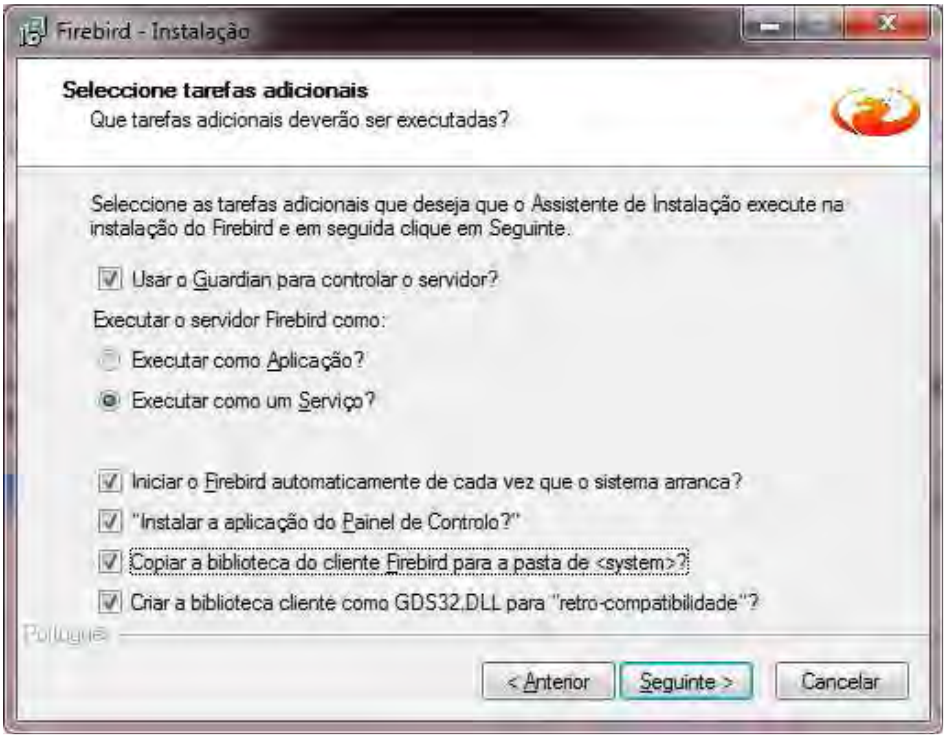

Figura 13 – Tela para configurar recursos adicionais oferecidos pelo Firebird.
Fonte: Elaborado pelo autor.

Depois da escolha dessas configurações, a próxima encerra o processo de instalação. Logo após a instalação, o serviço do SGBD Firebird já é iniciado e o computador está pronto para acessar bases de dados Firebird.

APÊNDICE C

CÓDIGO FONTE ORIGINAL PUBLICADO NA REVISTA CLUBE DELPHI EM 2006

```
unit untMain;
interface
uses SysUtils;
function CodifonPT_BR(nome: PChar): PChar; cdecl; export;
implementation
function CodifonPT_BR(nome: PChar): PChar;
var
  i, p: integer;
  novo, aux: string;
begin
  try
    aux := AnsiUpperCase(nome);
    novo := '';
    //tira acentos e cedilha
    for i := 1 to Length(aux) do
      begin
        case aux[i] of
          'Á', 'Â', 'Ã', 'À', 'Ä': aux[i] := 'A';
          'É', 'Ê', 'È', 'Ë': aux[i] := 'E';
          'Í', 'Î', 'Ì', 'Ï': aux[i] := 'I';
          'Ó', 'Ô', 'Õ', 'Ò', 'Ö': aux[i] := 'O';
          'Ú', 'Û', 'Ù', 'Ü': aux[i] := 'U';
          'Ç': aux[i] := 'C';
          'Ñ': aux[i] := 'N';
          'Ý', 'Y': aux[i] := 'I';
          else
            if Ord(aux[i]) > 127 then
              aux[i] := #32;
        end;
      end;
    //retira E, DA, DAS, DE, DI, DO E DOS do nome

    p := Pos(' DA ', aux);
    while p > 0 do
```

```
      begin
        Delete(aux, P, 3);
        p := Pos(' DA ', aux);
      end;

    p := Pos(' DAS ', aux);
    while p > 0 do
      begin
        Delete(aux, P, 4);
        p := Pos(' DAS ', aux);
      end;

    p := Pos(' DE ', aux);
    while p > 0 do
      begin
        Delete(aux, P, 3);
        p := Pos(' DE ', aux);
      end;

    p := Pos(' DI ', aux);
    while p > 0 do
      begin
        Delete(aux, P, 3);
        p := Pos(' DI ', aux);
      end;

    p := Pos(' DO ', aux);
    while p > 0 do
      begin
        Delete(aux, P, 3);
        p := Pos(' DO ', aux);
      end;

    p := Pos(' DOS ', aux);
    while p > 0 do
      begin
        Delete(aux, P, 4);
```

```
        p := Pos(' DOS ', aux);
      end;

    p := Pos(' E ', aux);
    while p > 0 do
      begin
        Delete(aux, P, 2);
        p := Pos(' E ', aux);
      end;
    //RETIRA LETRAS DULICADAS
    for i := 1 to Length(aux) - 1 do
      if aux[i] = aux[i + 1] then
        delete(aux, i, 1);
    for i := 1 to Length(aux) do
      begin
        case aux[i] of
    'B','D','F','J','K','L','M','N','R','T','V','X':
    novo := novo + aux[i];
    'C':
        if aux[i + 1] = 'H' then
        novo := novo + 'X'
        else
        if aux[i + 1] in ['A', 'O', 'U'] then
        novo := novo + 'K'
        else
        if aux[i + 1] in ['E', 'I'] then
        novo := novo + 'S';
    'G':
       if aux[i + 1] = 'E' then
        novo := novo + 'J'
        else
        novo := novo + 'G';

    'P':
      if aux[i + 1] = 'H' then
        novo := novo + 'F'
        else
```

```
        novo := novo + 'P';

    'Q':
    if aux[i + 1] = 'U' then
       novo := novo + 'K'
       else
       novo := novo + 'Q';

    'S':
      case aux[i+1] of
      'H':
      novo := novo + 'X';
      'A','E','I','O','U':
      if aux[i - 1] in ['A','E','I','O','U'] then
       novo := novo + 'Z'
       else
       novo := novo + 'S';
      end;
    'W':
       novo := novo + 'V';
    'Z':
       if (i = Length(aux)) or (aux[i+1] = ' ') then
       novo := novo + 'S'
       else
       novo := novo + 'Z';
   end;
  end;
    novo := novo + ' ';
    CodiFonPT_BR := PChar(novo);
  except
    CodiFonPT_BR := PChar('erro');
  end;
end;
end.
```

APÊNDICE D

PROCEDIMENTO (PROCEDURE) CRIADO NO FIREBIRD PARA BUSCA FONÉTICA

```
CREATE PROCEDURE CODIFONPT_BR (
    nome varchar(255))
returns (
    nome_fonetico varchar(255),
    parametro_2 varchar(1),
    parametro_3 varchar(1))
as
declare variable i integer;
declare variable k integer;
declare variable novo varchar(255);
declare variable v_com_acento varchar(50);
declare variable v_sem_acento varchar(50);
declare variable str1 varchar(1);
declare variable str2 varchar(1);
declare variable strant varchar(1);
begin
  NOME = upper(:NOME);
  NOVO = '';
  /*/ / TIRA ACENTOS E CEDILHA */

  V_COM_ACENTO = 'àâêôûãõáéíóúüÀÂÊÛÃÁÉÊÈËÍÎÌÏÝYÓÔÕÒÖÚÛÙÜÑ';
  V_SEM_ACENTO = 'aaeouaoaeiouuAAEUAAEEEEIIIIIIOOOOOUUUUN';

-- Substituir caracteres acentuados ------------------------------
--------
  I = 1;
  NOVO = '';
  while (:I < (char_length(:NOME) + 1)) do
  begin
    if (V_COM_ACENTO containing substring(:NOME from :I for 1)) then
    begin
      K = 1;
      while (K < 50) do
      begin
```

```
        if (substring(:NOME from :I for 1) = substring(:V_COM_ACENTO
from :K for 1)) then
          NOVO = :NOVO || substring(:V_SEM_ACENTO from :K for 1);
        K = :K + 1;
      end
    end
    else
      NOVO = :NOVO || substring(:NOME from :I for 1);
    I = :I + 1;
  end
  NOME = :NOVO;
-------------------------------------------------------------------
-------

 select contemsouz.temsouz
 from contemsouz (:nome)
 into :parametro_2;

 select contemx.temsouz
 from contemx (:nome)
 into :parametro_3;

--/ / RETIRA E, DA, DAS, DE, DI, do E DOS do NOME
  select TROCA.SAIDA
  from TROCA(:NOME, ' DA ')
  into :NOME;

  select TROCA.SAIDA
  from TROCA(:NOME, ' DAS ')
  into :NOME;

  select TROCA.SAIDA
  from TROCA(:NOME, ' DE ')
  into :NOME;

  select TROCA.SAIDA
  from TROCA(:NOME, ' DI ')
  into :NOME;
```

```
  select TROCA.SAIDA
  from TROCA(:NOME, ' DO ')
  into :NOME;

  select TROCA.SAIDA
  from TROCA(:NOME, ' DOS ')
  into :NOME;

  select TROCA.SAIDA
  from TROCA(:NOME, ' E ')
  into :NOME;

-- Substituir caracteres repetidos ---------------------------------
-------
  I = 1;
  NOVO = '';
  while (:I <= (char_length(:NOME))) do
  begin
    STR1 = substring(:NOME from :I for 1);-- JJOAO DA SILVAA
    STR2 = :STR1;

    if (:STR1 || :STR2 = substring(:NOME from :I for 2)) then
    begin
      NOVO = :NOVO || :STR1;
      I = :I + 2;
    end
    else
    begin
      NOVO = :NOVO || substring(:NOME from :I for 1);
      I = :I + 1;
    end
  end
  NOME = :NOVO;

  I = 1;
  NOVO = '';
  while (:I <= (char_length(:NOME))) do
```

```
  begin
    STR1 = substring(:NOME from :I for 1);

    if (:STR1 in ('B', 'D', 'F', 'J', 'K', 'M', 'R', 'T', 'W', 'L',
'*', '&', '$')) then
      NOVO = :NOVO || :STR1;
    else
    begin
      STR1 = substring(:NOME from :I for 1);
      if (:I < (char_length(:NOME))) then
        STR2 = substring(:NOME from :I + 1 for 1);
      else
        STR2 = null;
      if (:I > 1) then
        STRANT = substring(:NOME from :I - 1 for 1);
      else
        STRANT = null;

     if (:STR1 = 'N') then
      begin
        if ((:I = char_length(:NOME)) or (:STR2 = ' ')) then
          NOVO = :NOVO || 'M';
        else
          NOVO = :NOVO || 'N';
      end


     else if (:STR1 = 'X') then
      begin
        if ((:I = char_length(:NOME)) or (:STR2 = ' ')) then
         NOVO = :NOVO ||  'KS';
        else
          NOVO = :NOVO ||  'X';
      end

      else if (:STR1 = 'U') then
      begin
        if (:I = 1) then
```

```
        NOVO = :NOVO ||  'W';
      else if (:STRANT in ('A', 'E', 'O', 'U')) then
        NOVO = :NOVO ||  'L';
    end

    else if (:STR1 = 'C') then
    begin
      if ((:I = char_length(:NOME)) or (:STR2 = ' ')) then
      NOVO = :NOVO ||  'K';
      else if (:STR2 = 'H') then
        NOVO = :NOVO || 'X';
      else if (:STR2 in ('A', 'O', 'U')) then
        NOVO = :NOVO || 'K';
      else if (:STRANT = 'X') then
        NOVO = :NOVO ||  'S';
      else if (:STR2 in ('E', 'I')) then
        NOVO = :NOVO || 'S';
      else if (:STR2 in ('R', 'L')) then
        NOVO = :NOVO || 'K';
    end


    else if (:STR1 = 'Ç') then
      NOVO = :NOVO || 'S';

    else if (:STR1 = 'S') then
    begin
      if (:STR2 = 'H') then
        NOVO = :NOVO || 'X';
      else if (:STR2 = 'Ç') then
        NOVO = :NOVO || 'S';
      else if (:STRANT = 'C') then
        NOVO = NOVO || 'KS';
      else if ((:I = char_length(:NOME)) or (:STR2 = ' ')) then
        NOVO = :NOVO || 'S';
      else if (:STR2 = 'C') then
        NOVO = NOVO || 'S';
```

```
        else
          NOVO = NOVO || 'S';
      end

      else if (:STR1 = 'G') then
      begin
        if (:STR2 = 'E') then
          NOVO = NOVO || 'J';
        else if (:STR2 = 'I') then
          NOVO = NOVO || 'J';
        else
          NOVO = NOVO || 'G';
      end

      else if (:STR1 = 'P') then
      begin
        if (:STR2 = 'H') then
          NOVO = NOVO || 'F';
        else
          NOVO = NOVO || 'P';
      end

      else if (:STR1 = 'Q') then
      begin
        if (:STR2 = 'U') then
          NOVO = NOVO || 'K';
        else
          NOVO = NOVO || 'Q';
      end

      else if (:STR1 = 'Z') then -- IZADORA   IZABELA    'LUIZ SOUZA
E FILHO'
      begin
        if ((:STR1 = 'Z') and (I = 1)) then
          NOVO = :NOVO || 'Z';

        else if ((I = char_length(:NOME)) or (:STR2 = ' ')) then
          NOVO = :NOVO || 'S';
```

```
        else
          NOVO = :NOVO || 'Z';
      end


      else if (:STR1 = 'V') then
        NOVO = :NOVO || 'W';

      end
    I = :I + 1;
  end

  NOVO = :NOVO || ' ';

  NOME_FONETICO = Trim(:NOVO);
  suspend;
end
```

APÊNDICE E

PROCEDIMENTO (PROCEDURE) CRIADO NO FIREBIRD PARA RETIRAR AS PALAVRAS (E, DA, DAS, DE, DI, DO e DOS)

```
CREATE PROCEDURE TROCA (
    nome varchar(255),
    texto varchar(5))
returns (
    saida varchar(255))
as
declare variable aux varchar(255);
declare variable j integer;
declare variable tamstring integer;
begin
  J = 1;
  AUX = '';
  while (J <= (char_length(:NOME))) do
  begin
    TAMSTRING = char_length(:TEXTO);
    if (:TEXTO = substring(:NOME from :J for :TAMSTRING)) then
    begin
      J = :J + :TAMSTRING;
      AUX = :AUX || ' ';
    end
    else
    begin
      AUX = :AUX || substring(:NOME from J for 1);
      J = :J + 1;
    end
  end
  SAIDA = :AUX;
  suspend;
end
```

APÊNDICE F

PROCEDIMENTO (PROCEDURE) CRIADO NO FIREBIRD PARA ATRIBUIR NÚMEROS A PALAVRAS QUE CONTÉM S OU Z

```
CREATE PROCEDURE CONTEMSOUZ (
    nome varchar(255))
returns (
    temsouz varchar(1))
as
declare variable strant varchar(1);
declare variable aux varchar(1);
declare variable j integer;
declare variable str2 varchar(1);
declare variable tamstring varchar(255);
begin
  J = 2;
  temsouz = '0';
  nome = upper(:nome);
  TamString = (char_length(:NOME) -1 );
  while (J <= TamString) do
  begin
      strant = substring (:NOME FROM :J-1 FOR 1);
      STR2 = substring(:NOME from :J + 1 for 1);
      AUX = substring (:NOME FROM :J FOR 1);
      if (:AUX = 'Z') then
      begin
          temsouz = '1';
          break;
      end
      else if (((strant <> 'S') and (:aux = 'S') and (:str2 <> 'S'))
and ((:aux = 'S') and (:str2 <> 'Ç')) and ((:aux = 'S') and (:str2
<> 'C'))) then
      begin
         temsouz = '1';
          break ;
       end
```

```
      J = :J + 1;
  end
  suspend;
end
```

APÊNDICE G

CÓDIGO FONTE DE CONSULTA DO SISTEMA QUE REALIZA A BUSCA FONÉTICA

```
procedure TFConsultaCliente.BitBtn1Click(Sender: TObject);
var Palavra: string;
  RegraSouZ: string;
  FiltroAuxiliar: string;
  CodFonetico1, CodFonetico2: string;

  function PalavraRegraEspecial(P: string): Boolean;
  var I: integer;
  begin
    Result := false;
    for I := 2 to length(Palavra) - 1 do
      begin
        if (Palavra[I] = 'Z') then
          Result := true
        else if (Palavra[I] = 'S') then
          Result := true
        else if (Palavra[I] = 'X') then
          Result := true;

        if (Result) then
          break;
      end;
  end;

begin
  Palavra := Trim(UpperCase(Edit1.Text));
  if (PalavraRegraEspecial(Palavra)) then
  begin
    BuscaFoneticaEspecial;
  end
  else
  begin
      IBQuery1.Close;
      with IBQuery1.SQL do
```

```
        begin
          Clear;
          Add('select CLIENTE.NOME_CLIENTE, CLIENTE.CAMPO_FONETICO,
CLIENTE.TEM_S_OU_Z, CLIENTE.TEM_X');
          Add('from CLIENTE');
          Add('where CLIENTE.CAMPO_FONETICO like ''' +
            ''' || (select CODIFONPT_BR.NOME_FONETICO');
          Add('       from CODIFONPT_BR(' +
            QuotedStr(AnsiUpperCase(Edit1.Text))
            + ')) || ' + QuotedStr(''));
          Add('order by CLIENTE.NOME_CLIENTE');
        end;
  end;

  try

    IBQuery1.Open;
  except
    on E: Exception do
      ShowMessage(E.Message);
  end;
end;
```