Universidade Federal de Santa Catarina

FERRAMENTA DE AUXÍLIO AO DESENVOLVIMENTO DE SOFTWARE MULTILINGUAS

Michel Jean Grando

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

FERRAMENTA DE AUXÍLIO AO DESENVOLVIMENTO DE SOFTWARE MULTILINGUAS

Autor: Michel Jean Grando
Orientador: Ricardo Luiz Delfino Cunha
Co-orientador: Dr. José Eduardo De Lucca
Banca Examinadora: Luiz Fernando Bier Melgarejo
Palavras-Chave: Internacionalização de Software, Software Multilingue

Florianópolis 22 de maio de 2004

A minha família.
Ao meu orientador e professores.

Agradecimentos

Agradeço em primeiro lugar aos meus pais, por terem me dado a oportunidade de ingressar nesta instituição, a UFSC, e pelo apoio no decorrer desta caminhada.

Aos meus orientadores Ricardo Luiz Delfino Cunha e José Eduardo De Lucca pela orientação que me deram ao longo do desenvolvimento deste projeto. Aos membros da banca, pelas críticas e sugestões.

Aos meus amigos, principalmente os que foram colegas de curso e junto estiveram nessa jornada.

Lista de Abreviaturas

ID : *Identificador*

TP : *Agrupamento de Ids*

TXT : *Sequência de caracteres que designa um texto literal*

LOP : *Log de Operações do Programa*

AE : *Árvore de Elementos*

GRIDEDIT : *Grid de Edição de Textos Literais*

AML : *referente à extensão de arquivos Audaces MultiLinguas*

GLIBC : *biblioteca padrão POSIX do compilador C para LINUX*

POSIX : *Portable Operating System Interface*

Lista de Figuras

Resumo

O trabalho apresenta uma alternativa para o desenvolvimento de software, assim como manuais, tutoriais ou qualquer outro documento eletrônico ao qual se deseja dar suporte multilingue. Aqui, será dada ênfase especial ao desenvolvimento de software multilingue, que é o objeto pelo qual tem-se maior interesse.

Buscando suprir as necessidades para alcançar o objetivo supracitado, o trabalho demonstra a implementação de um Gerenciador Multilinguas: como o próprio nome sugere, tem o papel de fazer o gerenciamento dos idiomas com os quais se deseja dar suporte a uma certa aplicação. Desta forma, o desenvolvedor tem seu trabalho otimizado, evitando esforços redundantes e ficando livre para tarefas cruciais.

Abstract

This work presents an alternative approarch to software development, as well as manuals, tutorials or any other electronic document in wich multilanguage support is required. Here, it will be given special emphasis to multilanguage software development, wich is the most relevant object.

Having in mind to supress the job to achieve this objective, this papper proposes an implementation for a Multilanguage Manager: as its name is meant to be, manages the languages that you need to give support with, in a seamless way to the developer, leaving it free for other tasks.

Capítulo 1

Introdução

Neste capitulo, serão apresentadas as motivações que levaram a realização deste trabalho, assim como os objetivos a serem alcançados.

A última sessão do capítulo detalha como os próximos capítulos estão organizados.

1.1 Motivação

A contínua expansão da Internet pelo mundo abre e amplia mercados para todos os tipos de negócios, banalizando a capacidade que culturas que vivem a milhares de kilometros de distância tem de se intercambiar. Esta nova ordem no mercado mundial engloba todos os setores, e não seria diferente com a produção de software, onde as empresas responsáveis por este importante setor na economia mundial devem adequar seus processos de produção. O desenvolvimento de software com suporte multilingue é uma destas adequações impostas pelo mercado globalizado, e é onde a proposta deste trabalho de conclusão de curso dará uma importante contribuição.

Assim sendo, um software que dê suporte multilingue é de fundamental importância para as empresas que pretendem exportar e conquistar mercados internacionais. Tal condição vem tendo uma importância crescente nos últimos anos, exigindo atenção da comunidade internacional para legislar e criar procedimentos que facilitem o processo de desenvolvimento de software internacionalizável.

Além dos motivos comerciais gerais que dão a este projeto um bom conteúdo para a realização de um trabalho de conclusão de curso, o autor aqui apresenta um motivo especial, o qual lhe conferiu grande motivação para a realização do mesmo: a necessidade de um software de gerenciamento multilingue por parte da empresa onde o autor constitui

vínculos empregatícios. Na referida empresa, são desenvolvidos softwares de médio e grande porte que, atualmente, possuem versões em 5 idiomas.

Uma vez que não havia sido encontrada nenhuma ferramenta disponível que auxiliasse neste escopo, todo o gerenciamento e administração do código fonte com relação à textualização de interfaces, entre outros, era feita através da criação de arquivos *resource* (um mero arquivo textual, recurso este disponível no compilador *Borland Builder*) para cada um dos idiomas existentes, que será detalhado nos capítulos a seguir. Com o constante crescimento dos softwares, através da adição de novas funcionalidades e também casuais correções das funcionalidades já existentes, se tornou impraticável a sua manutenção, pois cada qual dos arquivos contendo textos literais pertencentes a cada idioma, continha de centenas até milhares de identificadores.

E, por último, outra motivação também válida para este trabalho é o fato deste possuir um forte apelo comercial no que se refere a pacotes de linguagens de programação, pois a maioria destes trazem algum tipo de ferramenta de auxílio para internacionalização, mas nenhuma tem um enfoque profissional ou praticável, quando severamente exigidos. Tais pacotes não tratam o tema da proposta deste trabalho como parte importante no processo de criação de software, fazendo com que o desenvolvedor se responsabilize pela maior parte do trabalho referente a todo o processo de gerenciamento, pesquisa, conhecimento e modificação dos textos em cada idioma. Consequentemente, o programador perde em produtividade, ao passo que um outro processo, um software no caso, poderia fazer praticamente todo o trabalho, de maneira mais fácil e inteligente, centralizando todas as informações fora da aplicação do usuário.

1.2 Objetivos

O que é o software multilingue? Até onde a internacionalização de software dentro de uma empresa deve interferir no processo de produção?

Entende-se por software multilingue como sendo aquele que dá suporte a mais de um idioma, possibilitando a sua utilização em uma das línguas incorporadas pelo mesmo. Assim sendo, um software produzido no Brasil, na língua portuguesa, deverá ter também

ter uma versão em outro idioma qualquer, de acordo com o país ou localidade ao qual o software se destina.

A proposta deste trabalho é um aplicativo, capaz de realizar um serviço redundante ao programador durante o desenvolvimento de uma ferramenta qualquer. Este aplicativo é capaz de reunir toda a informação textual de uma aplicação em um único arquivo seguro, para ser utilizado pelo desenvolvedor no seu código fonte, tornando-o mais elegante e fácil de manter.

Assim como a utilização deste aplicativo proposto, deve também existir todo um processo de produção e gerenciamento inteligente e organizado de software multilingue, envolvendo etapas no processo de produção do software, de maneira que possibilite e facilite a manutenção do suporte a idiomas estrangeiros. O mesmo conceito pode ser aplicado à criação de documentação, manuais, tutorias e outros.

Os softwares de médio e grande porte, onde a quantidade de informações nos mais variados idiomas cresce continuamente, tornando-se vasta e difícil de gerenciar, são os que farão melhor uso desta ferramenta.

São estes os termos que se deseja definir com este projeto: Inclusão digital e facilidade de uso por parte do cliente, alta lucratividade e facilidade de manutenção por parte do desenvolvedor do software.

1.3 Organização do Trabalho

Este trabalho está organizado em seis capítulos. O primeiro capítulo descreveu a motivação e o contexto no qual este trabalho está inserido, também abrangendo o objetivo geral e os objetivos específicos.

O capítulo 2 trata sobre a internacionalização de software, o avanço da Internet e as conseqüentes mudanças na ordem mundial com relação ao mercado de software, decorrentes da sua rápida expansão. Este mesmo capítulo traz ainda os principais objetivos da internacionalização de software.

No capítulo 3, será detalhada a evolução da internacionalização de software nas ultimas décadas, assim como os principais métodos utilizados até os dias atuais para internacionalização.

No capítulo 4, é apresentado o **Gerenciador MultiLinguas**, motivo deste trabalho de conclusão de curso. Todas as suas funcionalidades são demonstradas neste capítulo, assim como a classe importadora, que faz a inclusão do arquivo de línguas para o código fonte do usuário.

O capítulo 5 demonstra a aplicação da internacionalização a um software de usuário, no caso, o próprio **Gerenciador MultiLinguas**. Neste capítulo, são detalhados quais os passos que o programador deve seguir para implementar a internacionalização.

Por fim, o capítulo 6 traz as conclusões finais a respeito do trabalho, suas contribuições e indicações para trabalhos futuros.

Capítulo 2

Introdução

Atualmente as empresas de ponta do mercado internacional de software são provenientes de paises desenvolvidos, como EUA, Japão, ou da Europa como um todo. Os países em desenvolvimento, cujos softwares produzidos geralmente tem menor aceitação no mercado internacional, e que não utilizam o inglês como língua oficial, são os que necessitam maior atenção. Uma vez que estas empresas possuam o domínio de línguas estrangeiras convencionais nos softwares que produzem, seus produtos terão maior receptividade no exterior, tornando-se mais comercializáveis e competitivos perante gigantes do setor.

2.1 O Brasil e a Internacionalização

Em função da globalização e da conseqüente maior interação entre mercados a nível mundial, a internacionalização de software passou a ocupar lugar de destaque na economia mundial.

No âmbito dos paises em desenvolvimento, o Brasil se encontra numa posição privilegiada mundialmente, mas precisa dar maior importância à internacionalização do software: O país é considerado um dos 7 melhores mercados de software do mundo, tendo movimentado aproximadamente 1,9 bilhões de dólares em 2001 (Softex e MIC). Entretanto gera um grande déficit nas contas externas brasileiras, que era de 5% em 2001 e está sempre crescendo em função das demandas do processo de modernização industrial e empresarial. Prevê-se que, em 3 anos, as importações de software podem chegar a 5 bilhões de dólares [DE LUCCA03].

Este exemplo demonstra a necessidade de centrar esforços na exportação ao mesmo tempo em que se implanta uma política de substituição inteligente de importações (em especial de alta tecnologia pelo alto valor agregado que representam). Nesta atividade, portanto, desenvolver-se-ão ações no sentido de difundir junto aos empreendedores (empresas incubadas ou não) a consciência da necessidade de implantar uma estratégia para internacionalizar seus produtos e apoiá-los em seus esforços para tal. Os países do MERCOSUL serão o primeiro objetivo destes esforços pela proximidade física e pelo interessante mercado que representam. Como efeito colateral, atua-se em prol do fortalecimento do bloco de países, que representa um papel muito importante no equilíbrio econômico global[DE LUCCA04].

O país e as empresas precisam definir estratégias para atingir mercados internacionais. O primeiro passo para isso é melhorar a qualidade dos produtos e a produtividade dos processos de desenvolvimento, para corresponder à competitividade internacional. Além disso, as empresas precisam internacionalizar seus produtos (adequá-los às legislações locais, providenciar documentação multilingue, etc) e preparar sua infra-estrutura interna para uma nova realidade [DE LUCCA03].

2.2 Internacionalização de Software

Internacionalização pode ser conceituada como sendo a criação ou modificação de algum produto de modo a facilitar o uso em diferentes países e idiomas.

A internacionalização faz com que os softwares, manuais, tutoriais e demais documentação eletrônica, aceitem diversos formatos multilingues. Ou seja, o produto em questão deve se adaptar a exigências locais, incluindo: textos literais traduzidos, formatos de moedas e outras adaptações necessárias para atender aos requisitos culturais, legais e técnicos de mercados regionais.

A seguir, estão listados os principais objetivos da internacionalização de software, assim como a maneira a qual dever sem implementados:

- Garantir total funcionalidade em qualquer idioma: a aplicação tem seus textos literais originais traduzidos, formatos de moeda adaptados, entre outros, de acordo com cada um dos idiomas suportados;
- Total transparência para o usuário na mudança de idiomas de um aplicativo em tempo de execução;
- Centralizar o gerenciamento de textos literais fora da aplicação, de maneira que não seja necessário ter acesso ao código fonte da aplicação para que a mesma possa, por exemplo, alterar um texto literal. Isto, pois estas informações são redundantes para o desenvolvimento do código da aplicação;
- Reduzir os esforços e os custos de desenvolvimento e, principalmente, manutenção. A internacionalização, aplicada corretamente, deixa o código fonte mais robusto e livre de erros.
- Facilitar o trabalho dos tradutores, de maneira que eles possam ter acesso mais facilmente aos textos literais do programa;
- Oferecer um “paradigma de programação para software internacional”, que sirva de base para o desenvolvedor fazer um bom uso da internacionalização na sua plenitude, tendo como resultado um código robusto, eficaz e elegante, assim como com relação aos demais processos que envolvem uma empresa que deseja se tornar internacional.

Capítulo 3

Introdução

Neste capítulo será apresentado um breve histórico sobre evolução da Internet e da internacionalização nos últimos anos e do final do século passado, assim como as ferramentas mais utilizadas.

3.1 Uma Análise da Evolução da Internacionalização

Nas próximas sessões, será feito um comparativo entre o desenvolvimento da Internet e mercado mundial de software das décadas a partir do surgimento da Internet, até os dias atuais. Tentar-se-á demonstrar com isto o porque do atual momento vivido pela internacionalização de software, que há poucos anos veio a ter importância crescente e crucial para as empresas da área de TIC (Tecnologias de Informação e Comunicações).

3.1.1 Décadas do Século XX

O suporte à internacionalização nas décadas passadas praticamente não existiu, ou, quando houve, era implementado apenas em softwares focados ao uso técnico. Esta realidade perdurou até meados dos anos 90, por alguns motivos:

- A Internet era restrita ao uso acadêmico, militar ou a usuários que tivessem um maior poder aquisitivo;
- Eram poucos os softwares de usuário que contavam com interface gráfica. Os próprios sistemas operacionais com interfaceamento gráfico, que por sua

vez sofriam pela deficiência de performance dos micro-computadores da época, ainda estavam em formação;

- Havia uma falta muito grande de mecanismos padronizados para escrita e representação de outros idiomas;
- O mercado de software para usuário final que estava recém surgindo, apoiado nos outros motivos supracitados;
- Suporte a internacionalização em linguagens de programação limitado, disponível somente em C.

3.1.2 Ultima década do Século XX e Inicio do Século XXI

Como citado anteriormente, os últimos anos, muito mais acentuadamente desde o começo do século XXI, impôs a abertura dos mercados internos mundiais para a globalização. Buscando ter qualidade e competitividade nos seus produtos e tornar as exportações viáveis financeiramente, as empresas de software tiveram que se adaptar a uma nova exigência: a internacionalização dos seus produtos.

A internacionalização surgiu com força maior nos últimos anos, levada pelos seguintes fatores:

- A Internet, hoje mundialmente difundida, é usada como sinônimo para inclusão digital de povos de todo o planeta, é de fácil acesso;
- A enorme velocidade com que os softwares para usuário final tomaram a Internet. Hoje, podemos encontrar software-solução para praticamente qualquer tipo de problema;
- A explosão das interfaces gráficas, que agora tem bases sólidas nos sistemas operacionais e micro-computadores robustos e de altíssima performance;
- Surgimento de mecanismos padronizados de escrita e representação de idiomas diversos;
- Suporte em praticamente todas as linguagens de programação;

Assim, podemos concluir que os últimos anos do século passado, e os poucos vivenciados do século atual, foram marcados por uma revolução digital, buscando a inclusão digital, rápida criação, adaptação e distribuição de softwares pelo planeta. Desta maneira, melhores ferramentas para a internacionalização se fazem extremamente necessárias neste momento.

3.2 As Ferramentas Mais Utilizadas Atualmente

Atualmente, as linguagens de programação mais utilizadas, têm suporte nativo, de uma maneira ou de outra, à internacionalização. Porém, este tópico não será aqui discutido, por não ser objetivo deste trabalho demonstrar todos os suportes à internacionalização existentes em todas estas linguagens. Quer-se aqui, apenas passar uma noção geral de como a internacionalização de software foi realizada nos últimos anos, e como ela é implementada atualmente.

Nas próximas sessões, serão apresentadas as ferramentas mais utilizadas para internacionalização, desde as primeiras funções em bibliotecas C padronizadas pelo POSIX, até pacotes para criação, uso e manutenção de catálogos de mensagens para software livre.

Algumas destas ferramentas estão relacionadas a seguir:

- **Biblioteca padrão de C;**
- **Compilador C++ Builder: Arquivos *Resource*;**
- **Gettext: Software Livre;**

3.2.1 Biblioteca padrão de C

Primeira ferramenta utilizada para internacionalização, presente na biblioteca padrão do compilador C. Esta biblioteca é padronizada pelo POSIX para definir *locales*, que atuarão na tradução de mensagens e conversão de formatos: *strings,* classificação de caracteres, valores numéricos e monetários.

O uso de *locales* fará com que o aplicativo mantenha um mesmo tipo de representação canônica de informações e dados, ou seja, todas as operações sobre textos literais, valores numéricos e monetários, entre outros, serão sempre afetadas pelo *locale* selecionado.

A especificação de *locales* é simples: uma chamada ao método *setlocale*() da biblioteca C, passando-se por parâmetro a categoria que deseja-se mudar o *locale* atual, e qual *locale* usar.

Assim, teríamos algo como o exemplo a seguir:

`Setlocale(LC_CTYPE,C)`, onde **LC_TYPE** é a categoria a qual se deseja modificar o *locale*, e **C** é *locale* padrão do compilador C++ Builder.

Outras categorias de *locale* são:

- **LC_COLLATE:** referente ao modo como *strings* são ordenados;
- **LC_CTYPE:** refere-se ao modo como é feita a conversão e a classificação de caracteres;
- **LC_MESSAGES:** utiliza mensagens traduzidas, com base num catálogo de mensagens previamente construído;
- **LC_MONETARY:** afeta o formato de valores monetários;
- **LC_NUMERIC:** afeta o formato como valores numéricos são interpretados;
- **LC_TIME:** refere-se à formatação de data e hora.

3.2.2 Compilador C++ Builder: Arquivos *Resource*

A tradução de mensagens é feita separadamente do uso do *locale*, utilizando-se de um recurso disponível no compilador C++ Builder: os arquivos *resource*. Estes arquivos podem ser utilizados da seguinte maneira:

- Cria-se um arquivo *resource* para cada idioma a ser disponibilizado no software. Insere-se manualmente os textos literais neste arquivo, traduzidos para o idioma ao qual o arquivo representa, atribuindo-os a identificadores, que são os mesmos identificadores utilizados nos arquivos dos outros idiomas.

Exemplo da criação de dois arquivos de recurso: o primeiro em português, o segundo em inglês:

```
STRINGTABLE {
// strings da Aplicacao - AppAM*
   STR_APPAM_STRING, "TEXTO DO STRING"
/// stings ....
......
}

STRINGTABLE {
// strings da Aplicacao - AppAM*
   STR_APPAM_STRING, "STRING TEXT"
/// stings ....
......
}
```

- Faz-se a inclusão destes arquivos *resource* no código fonte da aplicação.

  Exemplo da inclusão de arquivos de recurso de idiomas ao projeto:

```
#ifdef LINGUA_PORTUGUES
#include "Portugues.rc"
#endif

#ifdef LINGUA_INGLES
#include "Ingles.rc"
#endif

#ifdef OUTRALINGUA ...
```

- No momento da compilação do aplicativo, o idioma ao qual deseja-se ter o mesmo compilado é fornecido ao compilador. O resultado final é o aplicativo compilado no idioma escolhido.

  Exemplo:

```
#define LINGUA_PORTUGUES
```

Assim, o software estará internacionalizado, porém, gerando uma enorme carga de trabalho sobre o desenvolvedor no gerenciamento dos textos literais e identificadores, principalmente em aplicações onde o número de identificadores pode chegar aos milhares.

Este método não prevê a mudança de idioma do aplicativo em tempo de execução.

3.2.3 Gettext: Software Livre

Os últimos anos também vêm sido acompanhados por uma outra revolução no desenvolvimento de softwares: os softwares de código fonte aberto, ou Software Livre. Estes softwares são geralmente também implementados em plataformas livres, representadas principalmente pelo Linux.

O pacote Gettext, de código fonte aberto, utilizado para criação, uso e manutenção de catálogos de textos literais, é disponível para o Linux. Ele age em conjunto com a GLIBC, um dos pacotes que geralmente vem incluído ao pacote do sistema operacional, ou também pode ser distribuído como uma biblioteca individual.

O seu uso é baseado na associação de catálogos de textos literais à categoria **LC_MESSAGES** do *locale*, detalhada na sessão 3.2.1. Com isto, ao chamar-se o método *gettext*(), passando por parâmetro um identificador para o texto literal desejado, o mesmo devolverá o texto literal no idioma especificado. Caso o catálogo ou mesmo a tradução para o idioma não seja encontrado, o próprio parâmetro passado para o método é retornado como resposta.

Assim, ao invés de se utilizar o trecho de código a seguir para imprimir uma mensagem na tela:

```
Printf("Isto é um teste");
```

Teríamos isto:

```
Printf(gettext("Isto é um teste"));
```

Onde o valor impresso na tela seria a mensagem “Isto é um teste”, traduzida para o idioma escolhido previamente.

O modo como *gettext* opera é, num contexto geral, a alma do funcionamento da aplicação que será proposta neste trabalho nos próximos capítulos.

Capítulo 4

Introdução

O **Gerenciador MultiLinguas** é um aplicativo de cunho administrativo, como o próprio nome sugere.

Através dele, será feita o gerenciamento dos textos literais das aplicações, sendo salvo o seu conteúdo em um arquivo texto codificado. Para ilustrar melhor, este arquivo pode ser considerado como sendo a base de dados de textos multilingue de uma determinada aplicação, onde cada texto literal terá sua tradução para o seu respectivo idioma. Assim sendo, este arquivo será usado como fonte para busca de todos os textos literais necessários ao código fonte, no idioma desejado, pelo aplicativo a que se deseja dar suporte multilingue.

Somente através do **Gerenciador MultiLinguas** poderá ser feita qualquer alteração neste arquivo, seja nos idiomas, ou nos textos literais. Assim, o programa centraliza em si o poder de manuseio dos dados, garantindo a coesão dos mesmos. A coerência só não pode ser assumida, na medida em que o programa não tem a função de um tradutor, ou seja, ele não realiza traduções automáticas dos dados inseridos. Isto fica ao encargo do usuário, para que o mesmo possa fazer as traduções a seu critério.

4.1 Os Tipos de Dados

Existem três tipos básicos de dados no programa:

I. **Identificadores (**ID**):** É uma sequência **única** de caracteres alfanuméricos (*string*) que designa, como o próprio nome sugere, um identificador para um determinado TXT da aplicação.

Exemplo de aplicação:
ID : STR_FMAIN_CAPTION

II. Tipos (TP**):** Nada mais é que um agrupamento de IDs. O motivo de sua existência é simples: dar ao usuário a ferramenta necessária para que ele possa agrupar IDs a seu critério, visando facilitar a sua futura manutenção. Também recebe uma sequência **única** de caracteres alfanuméricos (*string*) como nome, para diferenciá-los de outros TPs, inclusive IDs;

Exemplo de aplicação:
TP : STRS_FMAIN
IDs pertencentes a este tipo: STR_FMAIN_CAPTION
STR_FMAIN_HINT

III. Textos Literais (TXT**):** as TXTs são também sequências de caracteres alfanuméricos (*string*), que correspondem aos textos literais da aplicação do usuário, aos quais este último deseja tornar internacionalizável. Este tipo de dado é atribuído a um ID, tendo uma versão em cada idioma disponível.

Exemplo de aplicação:
ID : STR_FMAIN_CAPTION
TXT do ID na língua Portuguesa: "Formulário Principal"
TXT do ID na língua Inglesa: "Main Form"

4.2 O Formato de Arquivo MultiLinguas: Integridade e Coesão de Dados

Para garantir o correto funcionamento do programa, coibir ações indesejadas do usuário para com os arquivos e, assim, garantir a manutenção da integridade e coesão dos dados, os arquivos no formato MultiLinguas (*.aml) são cifrados.

Desta maneira, quando necessitar alterar algum dado no arquivo, o usuário não tem outra alternativa além de utilizar o Gerenciador MultiLinguas, que garante as duas condições, em condições normais de uso.

4.3 Janela Principal do Programa

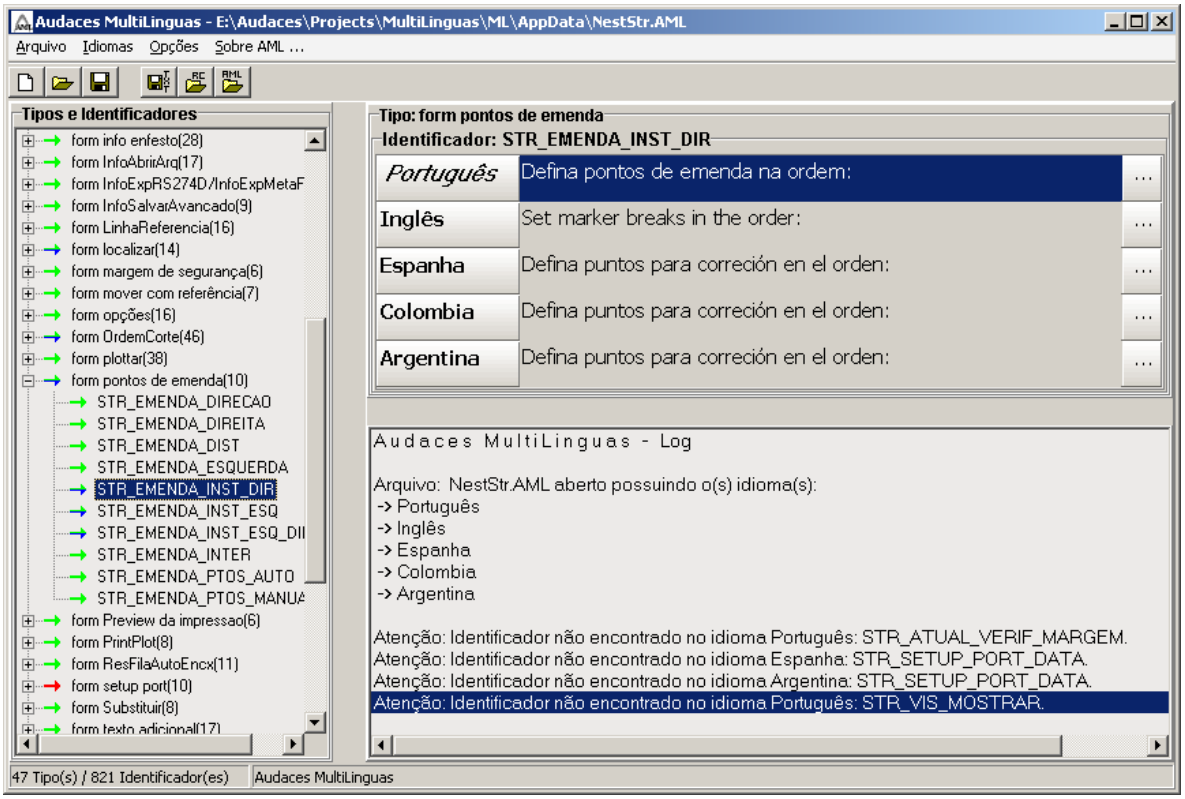


Figura 1: Janela Principal do Programa

Acima, temos a interface principal do programa. Podemos facilmente visualizar 3 importantes componentes:

I. ***Log* de Operação do Programa** (LOP): sua função é informar as ações mais importantes que foram realizadas, seja pelo usuário ou pelo sistema, assim como servir de guia para que o usuário corrija erros em potencial. Em

algumas situações, pode ser usado para localizar elementos na árvore de elementos, agilizando o processo de resolução.

II. **Árvore de Elementos (**AE**)**: é o principal componente do programa, que tem uma simples função: Mostrar ao usuário todos os TPs , com seus respectivos IDs. Fornece, desta maneira, a interface para que sejam feitas novas inserções, atualizações, exclusões, consultas, e outras operações sobre os dados do arquivo sendo atualmente exibido.

III. ***Grid*** **de Edição de TXTs (**GRIDEDIT**):** não menos importante do que a árvore de elementos, este componente é utilizado pelo usuário como ferramenta para edição de TXTs.

Conhecendo os componentes da interface principal do programa, serão apresentadas nas sessões a seguir cada uma das suas funcionalidades.

4.4 As Funcionalidades

Podemos separar as funcionalidades do programa em 3 grandes grupos, que serão mais bem detalhados nas próximas sessões:

- Funcionalidades de Edição Geral;
- Funcionalidades Sobre Idiomas;
- Funcionalidades Sobre Arquivo.

4.4.1 Funcionalidades de Edição Geral

Define as ferramentas utilizadas para edição como um todo de TPs, IDs e TXTs. São elas:

- Criação de Novos TPs e IDs;
- Procura (Localizar) de TPs, IDs e TXTs;
- Substituição de TPs, IDs e TXTs;

- Renomear TPs e IDs;
- Excluir TPs e IDs;
- Copiar/Recortar/Colar TPs e IDs.

4.4.1.1 Criação de Novos TPs e IDs

Constitui de uma das principais ferramentas do programa, sendo sua ação muito simples e direta:

- **Criação de TP**: um novo TIPO será criado, para que nele, novos IDs possam ser agrupados;
- **Criação de ID**: um novo ID será criado. Assim, um dado TXT, com versões em todos os idiomas disponíveis, pode ser designado para este ID.

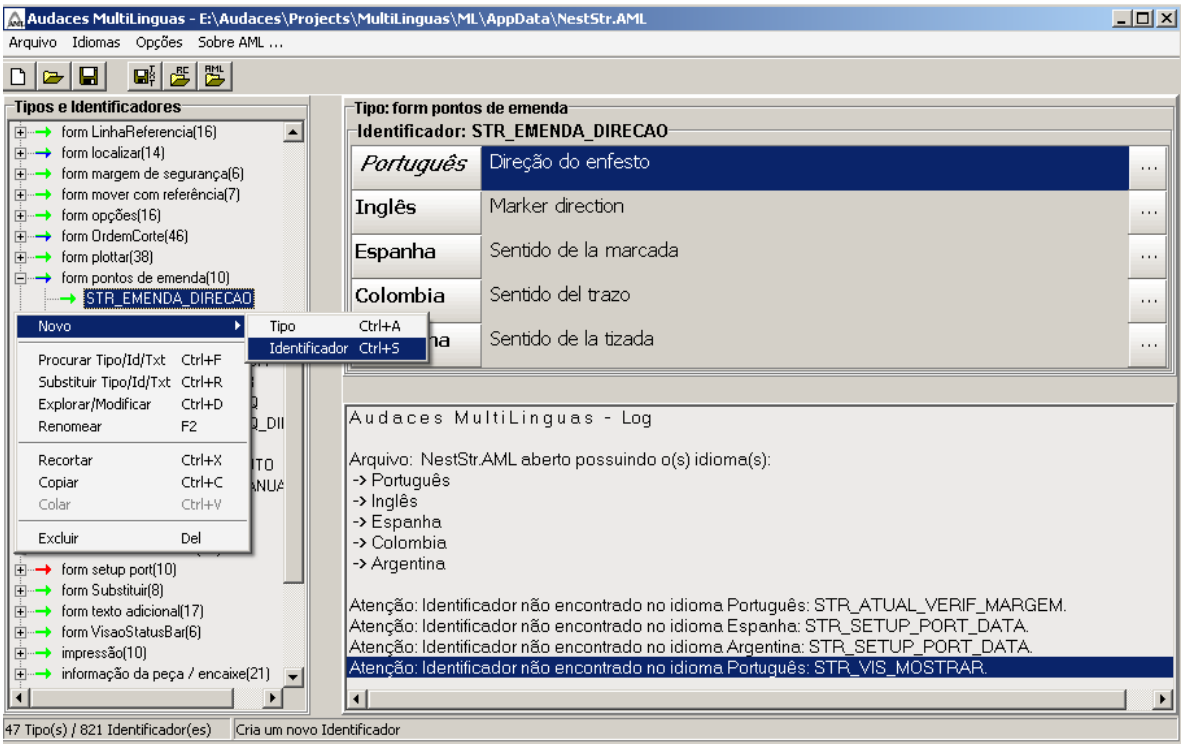


Figura 2: Criando um ID

4.4.1.2 Procura (Localizar) TPs, IDs e TXTs

Ferramenta esta indispensável em qualquer software que contenha um grande número de dados, é usada para facilitar a localização de:

- **TIPOs e IDs na AE**: localiza o elemento na AE que possua o conjunto de caracteres fornecido no campo de procura, selecionando-o e mostrando o seu respectivo TXT na GRIDEDIT, sendo que este TXT possui versões em cada idioma disponível;
- **TXTs na GRIDEDIT**: localiza e mostra TXTs que possuam o conjunto de caracteres fornecido no campo de procura, selecionando na AE o ID ao qual ela pertence.

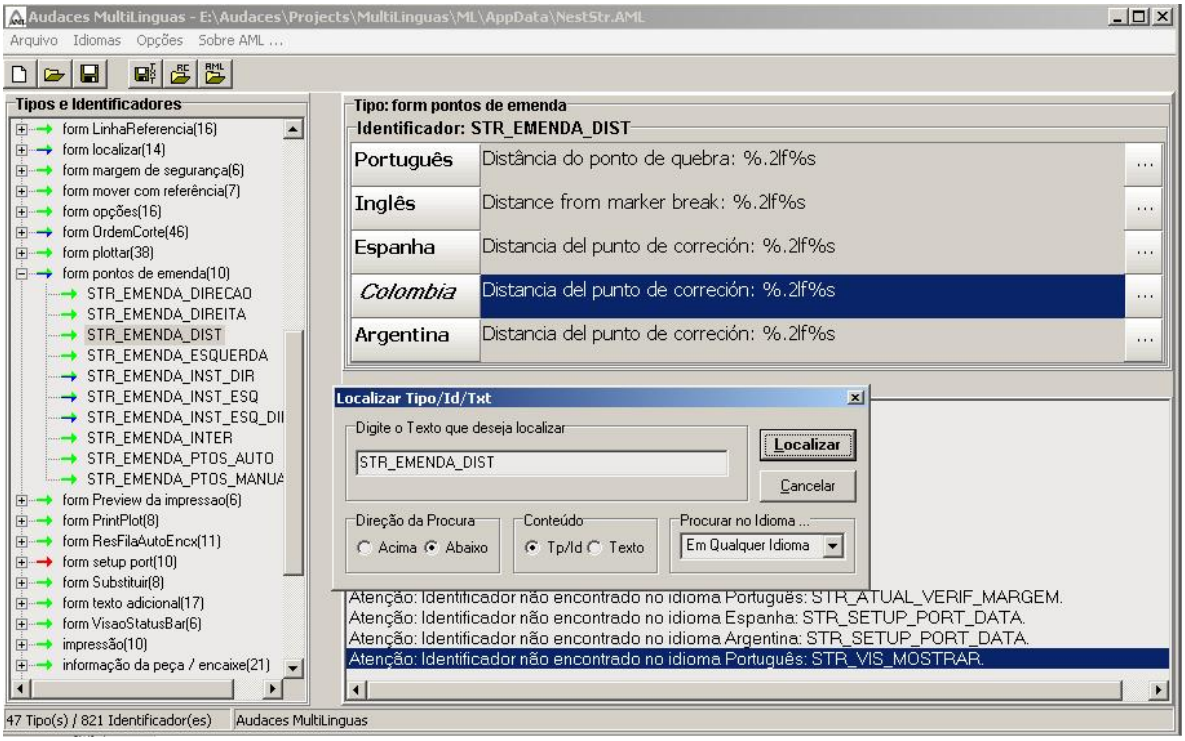


Figura 3: Procurando por um TP ou ID na AE

4.4.1.3 Substituição de TPs, IDs e TXTs

Ferramenta não tão indispensável quanto a anterior, mas que também geralmente consta em softwares que lidam com grande número de dados. É usada para facilitar a localização e subsequente substituição de:

- **TIPOs e IDs na AE**: localiza o elemento na AE que possua o conjunto de caracteres fornecido no campo de procura, selecionando e substituindo-o pela sequência de caracteres especificada no campo de substituição.
- **TXTs na GRIDEDIT**: localiza TXTs que possuam o conjunto de caracteres fornecido no campo de procura, selecionando e substituindo-os pela sequência de caracteres especificada no campo de substituição.

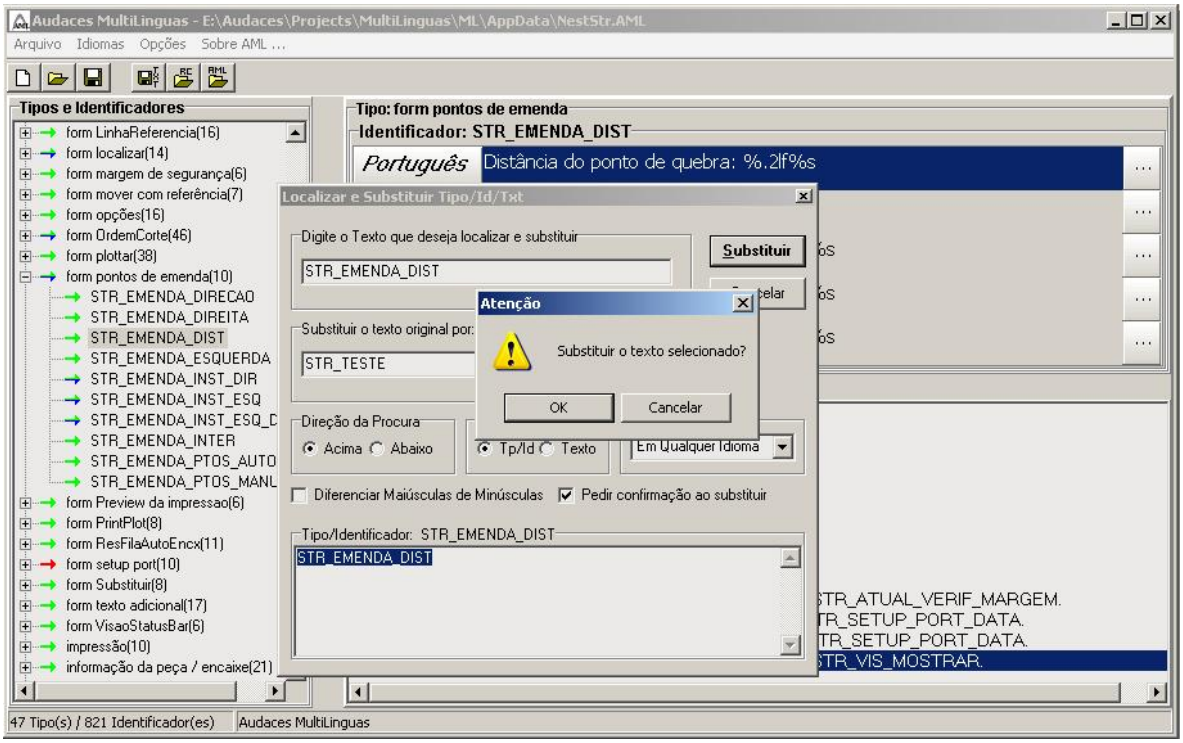


Figura 4: Substituindo o nome de um TP ou ID

4.4.1.4 Renomear TPs e IDs

Esta ferramenta é utilizada para atribuir uma nova sequência de caracteres alfanuméricos a um TP ou ID, lembrando *que* os mesmos tem nomes **únicos**, e não devem coincidir com o nome de outro TP ou ID.

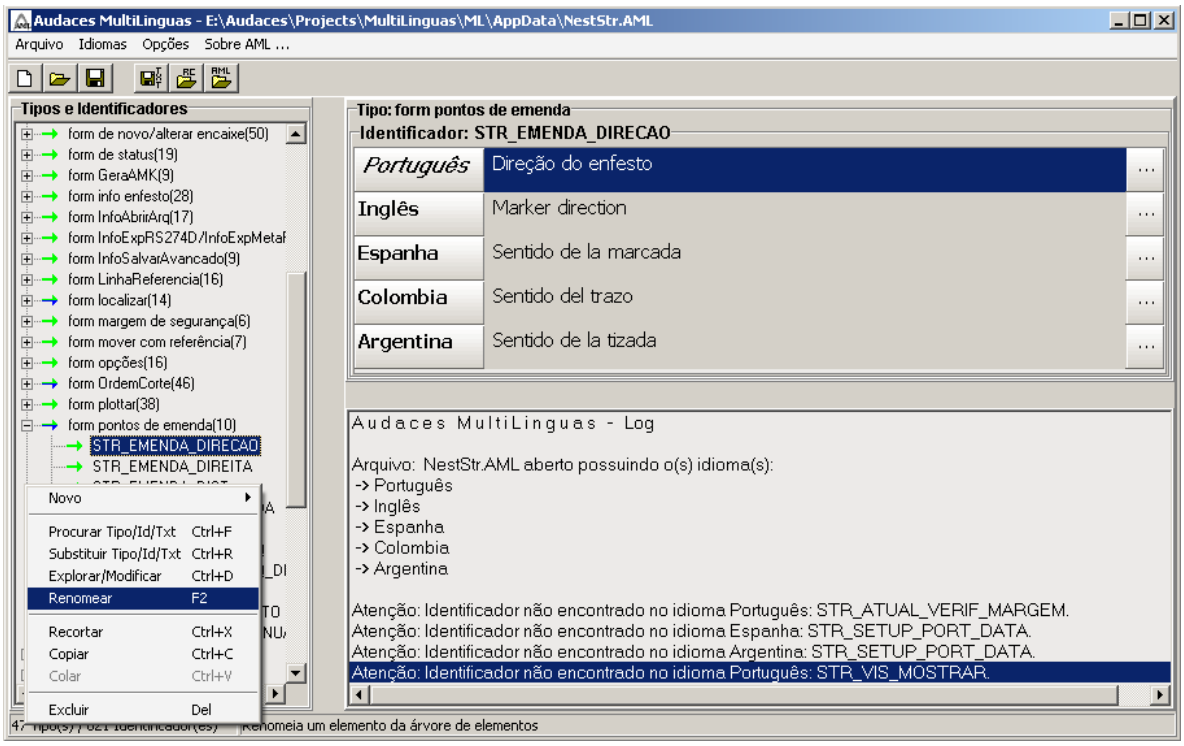


Figura 5: Renomeando um ID

4.4.1.5 Excluir TPs e IDs

Com o uso desta ferramenta, é possível excluir permanentemente TPs ou IDs da AE, a critério do usuário.

Importante: Na tentativa de excluir um TP que possua um ou mais IDs, todos os seus IDs serão também permanentemente excluídos.

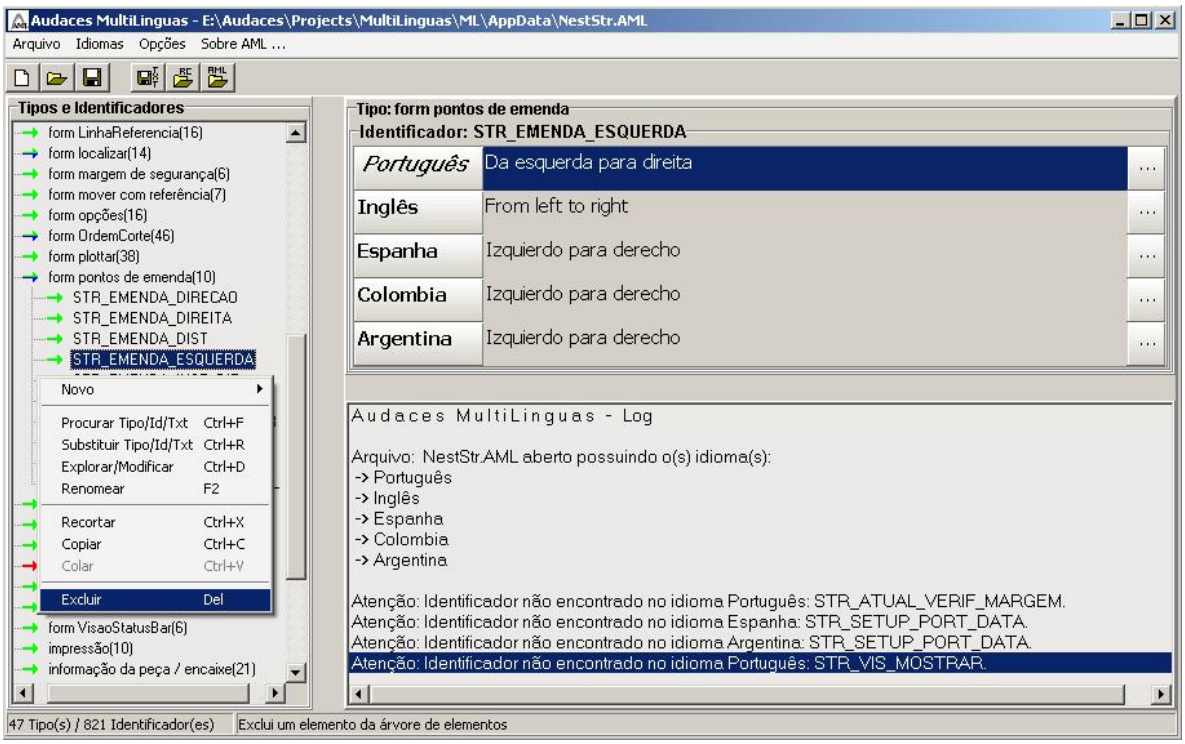


Figura 6: Exclusão de um ID

4.4.1.6 Copiar/Recortar/Colar TPs e IDs

Ferramenta esta concebida para poupar trabalho ao usuário em situações bastante restritas.

- **Copiar e Colar TPs:** usando estes dois procedimentos em sequência, será criado um novo TP na AE, cujo nome será o mesmo do TP copiado, acrescentado de um índice ao final do seu nome. O mesmo é aplicado aos respectivos IDs do TP copiado. Pode ser usado para, por exemplo, criar uma cópia temporária de um TP na AE;
- **Recortar e Colar TPs:** esta operação não pode ser realizada, pois todos os TPs da AE são ordenados por ordem alfabética e, portanto, não tem outro lugar na AE além do lugar o qual eles ocupam;
- **Copiar e Colar IDs:** o uso destes dois procedimentos, também em sequência, criará um ID de mesmo nome do ID que foi copiado, com o acréscimo de um índice ao final do seu nome;

- **Recortar e Colar IDs:** esta operação é útil quando deseja-se transferir um ID de um TP para outro. A realização desta operação em um mesmo TP, não tem efeito algum.

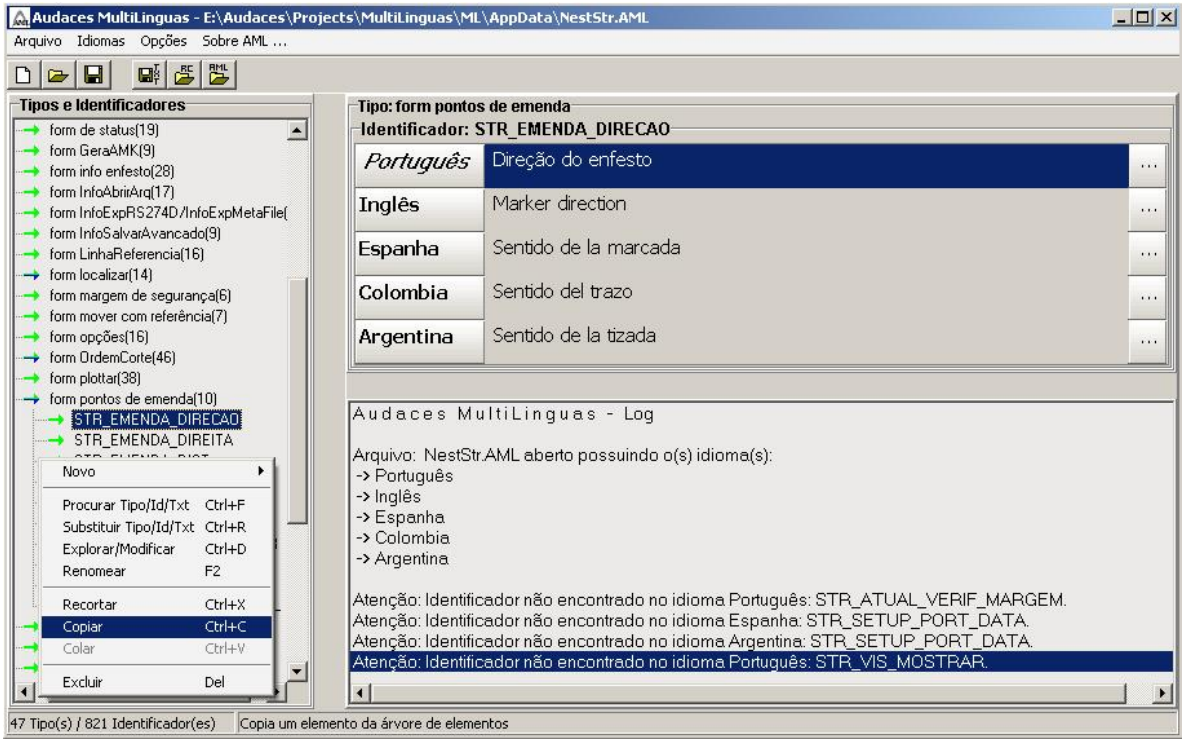


Figura 7: Copiando um ID

4.4.2 Funcionalidades Sobre Idiomas

Define as ferramentas necessárias para manuseio de idiomas.

- Criação e Exclusão de Idiomas;
- Criar Cópia de Idiomas;
- Mudança de Nome de Idiomas.

4.4.2.1 Criação e Exclusão de Idiomas

Esta é a ferramenta utilizada para:

- **Inserir um Novo Idioma ao Arquivo Atual:** Tal operação é necessária quando, por exemplo, se deseja dar suporte a um novo idioma as TXTs existentes, fazendo com que mais um campo, contendo o nome do idioma adicionado, seja inserido na GRIDEDIT.
- **Excluir um Idioma do Arquivo Atual:** Operação sobre os idiomas do arquivo atual, realizada quando se deseja remover o suporte dos IDs a um determinado idioma. Assim sendo, para qualquer ID da AE, o campo na GRIDEDIT que designava TXTs do idioma excluído, não mais existirá.

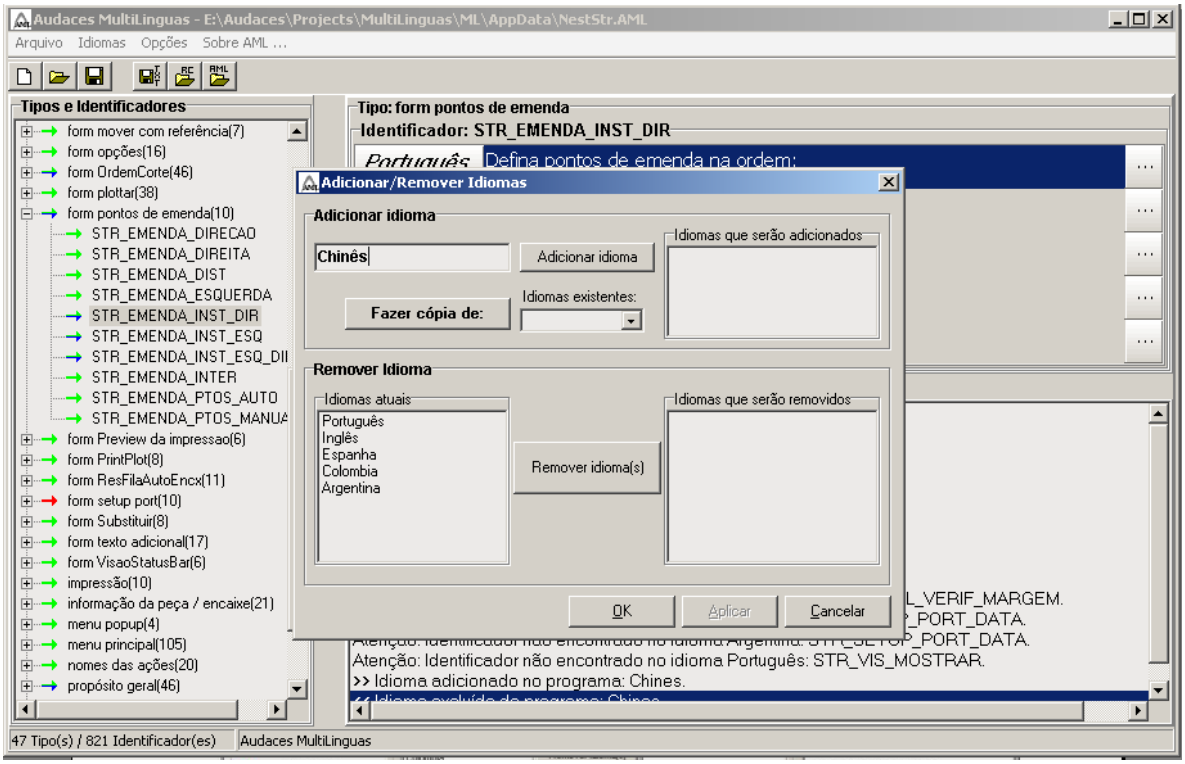


Figura 8: Adicionando um novo idioma

4.4.2.2 Criar Cópia de Idiomas

Esta ferramenta é utilizada para adicionar um idioma ao arquivo atual, mas não apenas isso: todos os seus TXTs serão copiados de um idioma já existente.

A ferramenta torna-se de grande valia quando o usuário deseja dar suporte, por exemplo, a um idioma de um país que seja o mesmo de outro país, cujo idioma já existe no arquivo atual, mas que possua apenas algumas diferenças gramaticais na construção de suas frases entre os dois. Assim, bastarão algumas alterações manuais nos TXTs do novo idioma adicionado, ao invés de inserir um a um, que seria o procedimento padrão realizado quando se adiciona um novo idioma ao arquivo atual, explicado na sessão anterior.

Exemplo prático:

Idioma Português no Brasil, e idioma Português em Portugal.

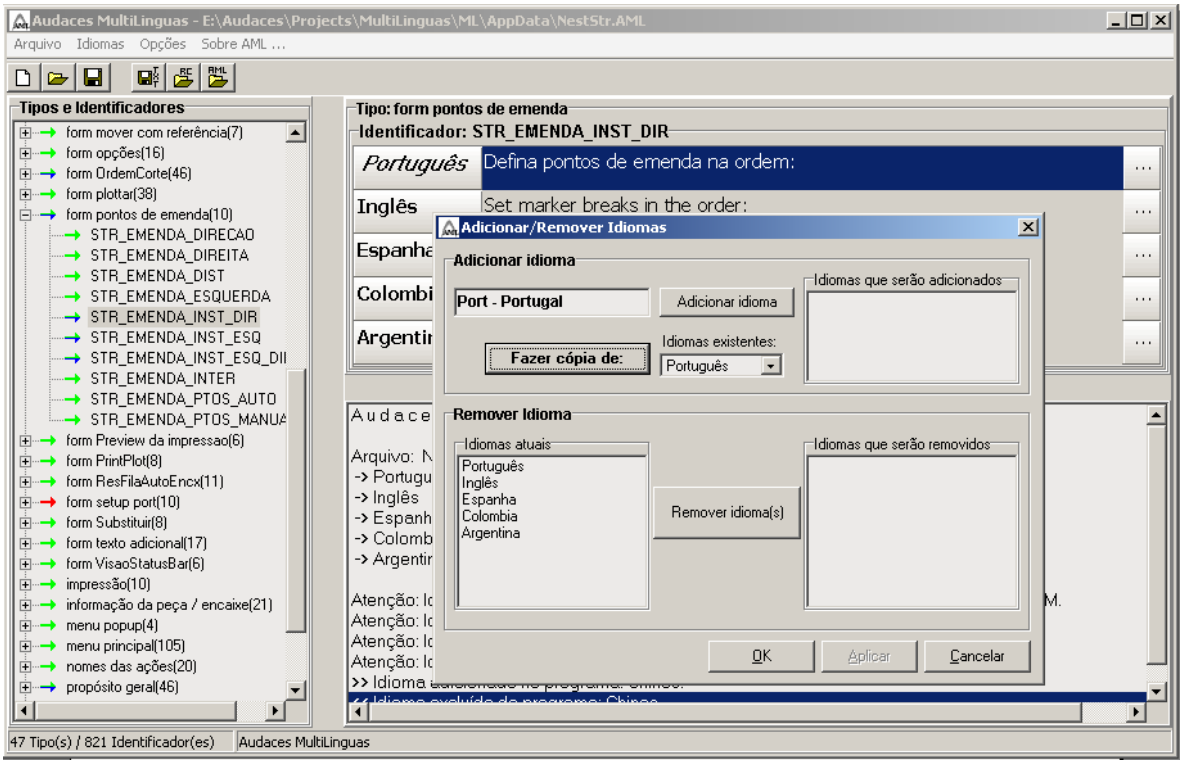


Figura 9: Fazendo cópia de idioma

4.4.2.3 Mudança de Nome de Idiomas

Esta funcionalidade é simples e direta: utilizada para mudar o nome de um idioma para outro nome, porém, que não coincida com o nome de algum outro idioma já existente.

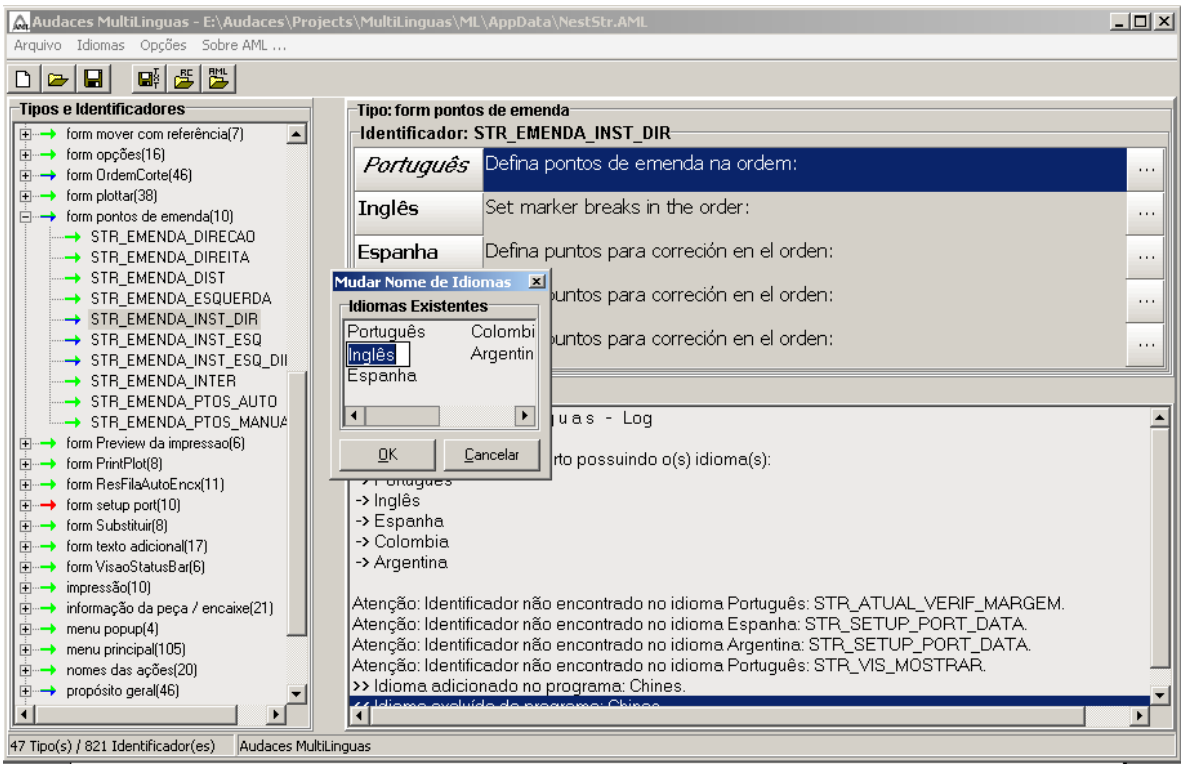


Figura 10: Mudando o nome de um idioma

4.4.3 Funcionalidades Sobre Arquivos

Define ferramentas genéricas sobre arquivos:

- Ações Genéricas: Novo, Abrir e Salvar Arquivos;
- Importar/Mesclar Arquivo Resource;
- Mesclar Arquivo AML.

4.4.3.1 Ações Genéricas: Novo, Abrir e Salvar Arquivos

Esta sessão define as ferramentas que qualquer software em geral que lida com dados voláteis necessita:

- **Novo Arquivo:** Reinicializa a interface principal do programa, fechando o arquivo atual;
- **Abrir Arquivo:** Abre um arquivo no formato MultiLinguas (*.aml);

- **Salvar e Salvar Como:** Salva o conteúdo atual do programa em um arquivo no formato MultiLinguas (*.aml);
- **Salvar Arquivo TXT:** Salva o conteúdo atual do programa em um arquivo formato Texto (*.txt).

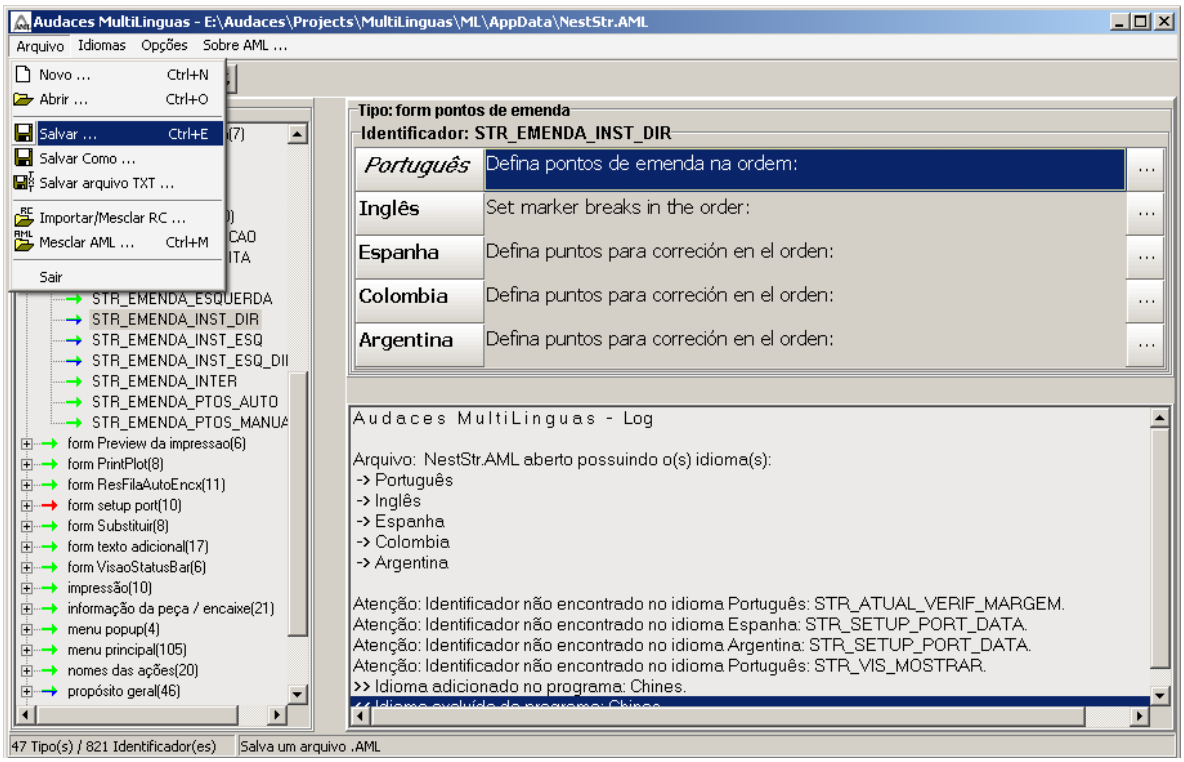


Figura 11: Salvar um arquivo AML

4.4.3.2 Importar/Mesclar Arquivo Resource

Esta ferramenta faz a importação ou mesclagem de um arquivo *Resource* para o programa, formato este detalhado anteriormente. Este tipo de arquivo apresenta-se ao usuário no formato texto, e pode ser editado abertamente, o que poderia gerar uma série de erros na importação ou mesclagem. Para evitar estes problemas, o arquivo passa inicialmente por um *parser*, corrigindo possíveis erros como, por exemplo, renomear TPs e IDs com nomes repetidos.

Após a análise prévia do arquivo, a decisão entre importação/mesclagem é feita pelo próprio programa. Assim sendo, a mesclagem ocorre se, no momento em que a ferramenta

foi requisitada, já existia conteúdo inicializado no programa, como por exemplo, um arquivo que estava aberto na interface. Caso contrario, o conteúdo do arquivo é importado para o programa, como se um arquivo estivesse sendo aberto. Em ambas as situações, o usuário deve designar um novo idioma para o conteúdo do arquivo sendo importado/mesclado, pois, por decisões de projeto, não é possível a mescla de dois arquivos *Resource* num mesmo idioma.

Como o processo de mesclagem pode ainda vir a encontrar inconsistências, este deve receber uma atenção maior. Neste processo, tentar-se-á mesclar o conteúdo atual do programa, com o conteúdo do arquivo sendo mesclado e, caso haja diferenças cruciais entre os referidos, um formulário de mesclagem será instanciado, dando opções ao usuário para decidir quais atitudes tomar com relação às inconsistências encontradas.

Estando uma vez neste formulário de mesclagem, o usuário tem acesso visual imediato a três principais componentes:

- Uma AE à esquerda, contendo todos os elementos já presentes anteriormente no programa;
- Uma AE à direita, contendo todos os TPs e IDs encontrados no arquivo sendo mesclado, e que não foram inseridos ao programa por algum motivo;
- Um LOP, que descreve o motivo pelo qual cada um dos elementos da AE da direita não foram inseridos automaticamente no programa, entre outros. Este componente tem a função de conduzir o usuário, para que ele, usando as ferramentas disponibilizadas em menus de acesso rápido sobre as AEs, possa corrigir as inconsistências encontradas.

Fazendo uso dos três componentes visuais, assim como menus de acesso rápido contendo as mesmas ferramentas de edição geral (descritas na sessão 3.3.1), o usuário tem condições de decidir com relação a cada um dos problemas encontrados na mesclagem. O usuário tem também a opção de ignorar o tratamento destes problemas, descartando automaticamente todos os elementos que não foram inseridos na mesclagem e voltando à janela principal do programa.

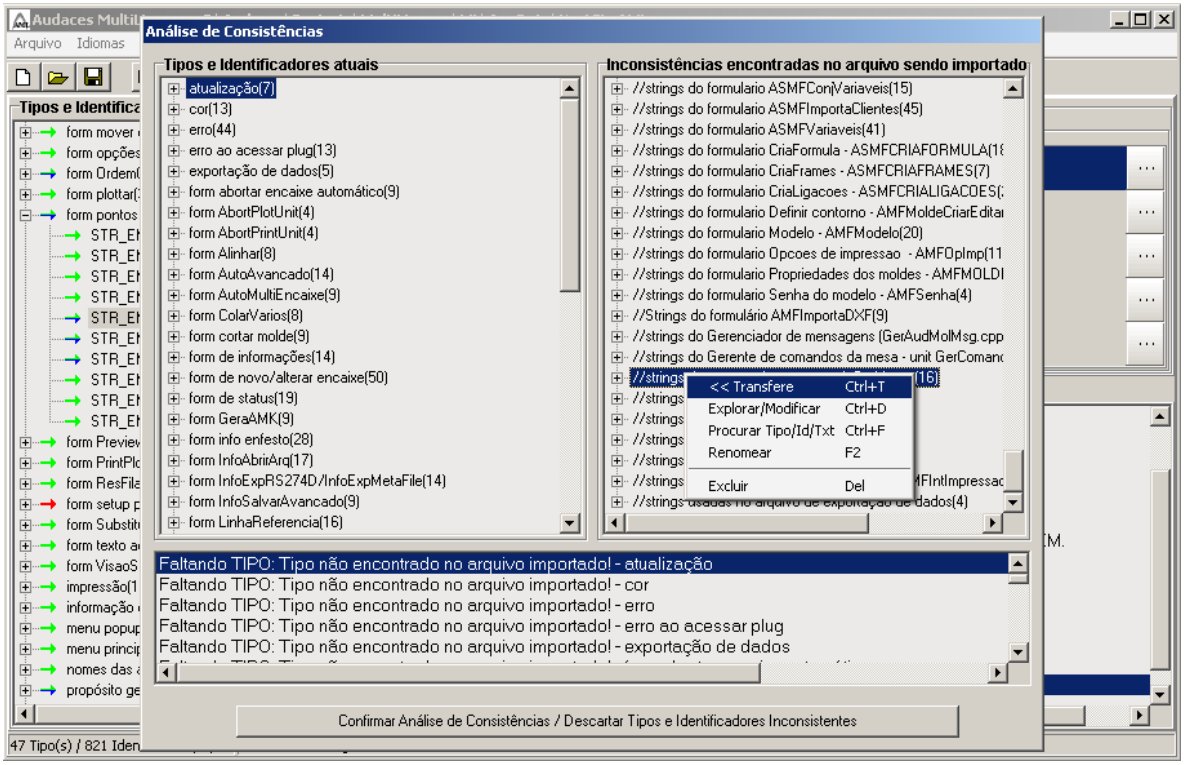


Figura 12: Mesclando um arquivo ao programa

4.4.3.3 Mesclar Arquivo AML

Esta ferramenta funciona de forma muito semelhante à importação/mesclagem de arquivos *Resource*, descrita na sessão anterior. Porém, por se tratar de mesclagem de arquivos fechados (cifrados) que, portanto, não podem ser editados fora no Gerenciador MultiLinguas, tem sua integridade e consistência garantidas em condições normais de uso.

Partindo desta premissa, a mesclagem de arquivos AML é mais simples: não existe *parser* para prévia análise do conteúdo do arquivo. Ao requisitar o uso da ferramenta, o programa tentará mesclar automaticamente o conteúdo do arquivo com o conteúdo atual do programa, adicionando novos idiomas, caso forem encontrados.

Novamente, da mesma maneira como na importação/mesclagem de arquivos *Resource*, faz-se uso da mesma interface de mesclagem, que é instanciada caso alguma inconsistência seja encontrada, para resolver problemas pendentes. A descrição desta interface se encontra na sessão anterior, e funciona exatamente da mesma maneira em ambas as situações.

4.5 A Classe Importadora

Como detalhado na sessão 4.2, os arquivos no formato MultiLinguas (*.aml) são cifrados. Portanto, não seria possível a leitura deste no código fonte do usuário, caso não existisse um meio que faça o interfaceamento para o arquivo AML, para que o seu conteúdo possa ser acessado para obter os textos literais. É com este objetivo que foi desenvolvida a classe importadora.

A classe importadora faz o processo inverso da cifragem, decifrando o conteúdo do arquivo para que a leitura possa ser feita e, assim, ter acesso aos textos literais contidos no arquivo.

O seu funcionamento é bastante simples: no construtor da classe, passa-se por parâmetro o idioma ao qual deseja-se ter acesso aos textos literais, e o caminho do arquivo AML. Feita a inicialização da classe, basta chamar o método que retorna textos literais, passando para ele, por parâmetro, o ID ao qual se deseja obter o texto literal.

Exemplificando, segue este trecho de código:

```
ImportadorAML *importadorAml= new ImportardorAML(idioma, caminhoArqAML);
importadorAml->PegaTxtId(ID);
```

Capítulo 5

Introdução

Este capítulo apresenta um exemplo de aplicação da internacionalização, como ela foi proposta neste trabalho.

Como será visto a seguir, os fontes são do próprio Gerenciador MultiLinguas, responsável pelo gerenciamento dos textos literais, que foi internacionalizado para demonstrar o funcionamento de um aplicativo internacionalizado.

Inicialmente, o gerenciador dava suporte apenas à língua portuguesa. Com a sua internacionalização, o suporte aos idiomas inglês e espanhol foi incluído, podendo ainda ser expandido quando for necessário. Portanto, este exemplo trata-se da conversão de um aplicativo comum, para um aplicativo internacionalizado.

5.1 Exemplo de Software Internacionalizado: O Gerenciador MultiLinguas

O Gerenciador MultiLinguas, apresentado no capítulo anterior, serviu também como programa de teste para demonstrar o funcionamento de um software internacionalizado.

A demonstração da aplicação internacionalizada será feita em 3 passos:

- **Primeiro Passo:** Geração de um arquivo (*.aml) que terá os textos literais da aplicação traduzidos em cada um dos idiomas suportados;
- **Segundo Passo:** Criação de uma classe que fará uso da classe importadora, também detalhada no capitulo anterior. Esta classe terá acesso ao arquivo MultiLinguas previamente gerado no **Primeiro Passo**;

- **Terceiro Passo:** Chamadas à classe criada no **Segundo Passo**, para que se obtenha um texto literal desejado, no idioma escolhido.

Para ilustrar o desenvolvimento de um aplicativo ao qual deseja-se obter suporte a internacionalização, três classes do projeto do gerenciador serão analisadas ao decorrer da descrição dos três passos nas próximas sessões deste capítulo:

- **MLGerMultiLinguas:** esta classe implementa um **gerente de línguas**, cujo qual é utilizado por toda a aplicação para, entre outros, selecionar o idioma corrente e ter acesso à classe que manipula (lê) os textos literais contidos num arquivo (*.aml) gerado previamente;
- **MLFMain:** esta classe é a implementação do **formulário principal** do aplicativo. Será analisada com o objetivo de mostrar um exemplo de aplicação em um formulário;
- **MLComdIOAml:** classes que implementam os **dispositivos de abertura e salvamento de arquivos no formato** (*.aml), demonstrará a utilização do gerente de línguas, que funciona de maneira similar ao formulário principal.

5.1.1 Primeiro Passo: Geração de um Arquivo AML

Este não é necessariamente o primeiro passo ao implementar-se um aplicativo internacionalizável, e muito dificilmente o será, pois geralmente não sabemos com antecedência quais os textos literais que estarão presentes no programa, assim como seus respectivos escopos, para que possam ser agrupados corretamente em TPs no arquivo. Foi assim escolhido para ser o primeiro passo, apenas para facilitar a compreensão da internacionalização de um software, demonstrando que os IDs, com seu respectivo texto literal no idioma original, e as suas traduções nos outros idiomas, devem estar disponíveis.

O arquivo sendo mostrado na figura abaixo, aberto no gerenciador, é o arquivo AML que contem todos os textos literais usados pelo aplicativo. O modo de criação de TPs e IDs foi também demonstrado no capitulo anterior e, portanto, não será comentado novamente.

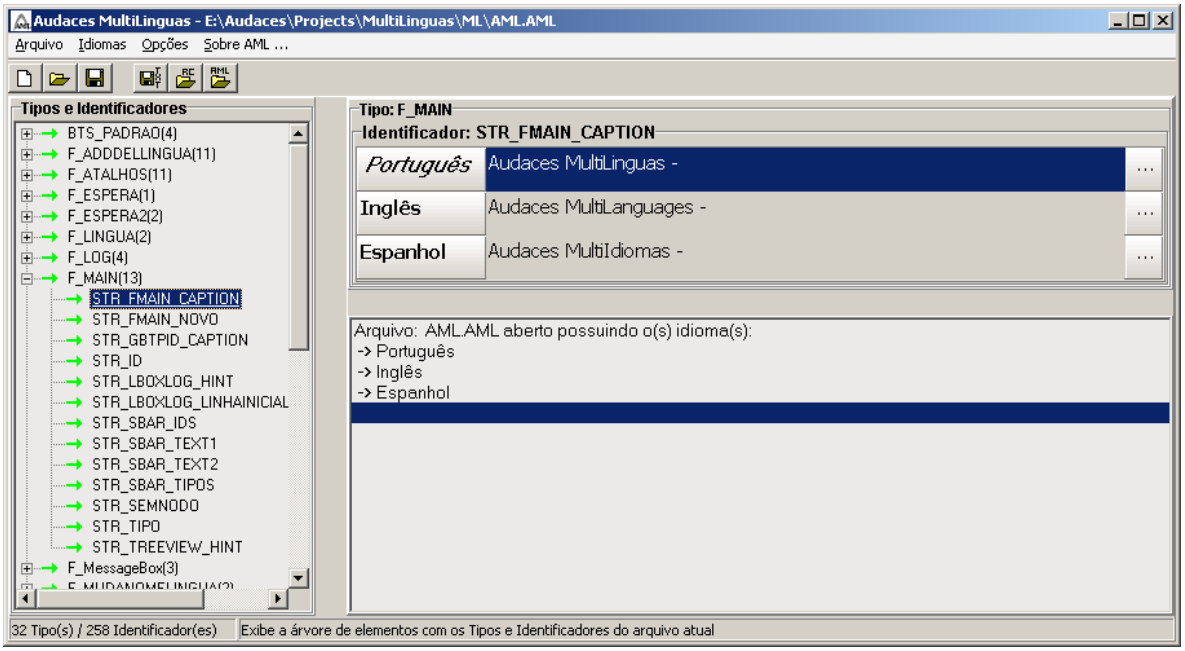


Figura 13: o arquivo AML utilizado pelo próprio gerenciador

Tendo cada texto literal sendo atribuído a um ID, e cada um destes IDs devidamente agrupados dentro de TPs, o arquivo está pronto para ser utilizado pela aplicação.

5.1.2 Segundo Passo: A Classe Importadora

Como detalhado no capítulo anterior, para fazer uso do arquivo AML gerado, é necessária a inclusão ao projeto de uma classe importadora, que acessará os dados contidos no arquivo para a aplicação em questão. O uso da classe importadora se faz necessário, pois, como também citado em capítulos anteriores, o arquivo AML é cifrado, ou seja, sem ela, não seria possível a correta leitura do arquivo.

Abaixo está exemplificado a inicialização da classe importadora no código fonte do Gerenciador MultiLinguas:

```
void MLGerMultiLinguas::Inicializese()
{
   /*tenta abrir o arquivo AML*/
   if (!FileExists(pathArqAML)) {
         MLGI::MsgAlerta("AML file not found, aborting!");
         /*repare que na mensagem de erro acima, a classe importadora não
         é usada, pois o arquivo AML contendo os textos literais não pode
```

```
            ser aberto! Sendo assim, foi designada uma mensagem em inglês
            para alertar o usuário*/
            exit(EXIT_SUCCESS);
    }
    /*lê arquivo de configuração para saber qual idioma usar*/
    idioma= LeArqInit();

    switch (idioma) {
       case 0: strcpy(idiomaAtual, "PORTUGUES"); break;
       case 1: strcpy(idiomaAtual, "INGLES"); break;
       case 2: strcpy(idiomaAtual, "ESPANHOL"); break;
       default: strcpy(idiomaAtual, "INGLES");
    }

    /*instancia o importador AML com o arquivo e o idioma selecionado*/
    importador= new ImportarAml(idiomaAtual, pathArqAML);
}
```

Ou seja, o importador é inicializado com o idioma que será utilizado na aplicação, deixando-o preparado para retornar os textos literais dos IDs solicitados.

Abaixo está o código do método da classe que acessa o método da classe importadora, retornando o texto literal de um determinado ID:

```
char *MLGerMultiLinguas::LeTxtAML(char *idTxt)
{
      return importador->PegaTxtId(idTxt);
}
```

O método é tão simples quanto parece: acessando PegaTxtId(ID) da classe importadora, o valor de retorno é o texto literal referente aquele ID no idioma previamente escolhido.

Uma questão aqui pertinente seria: Não é possível alternar o idioma durante a execução do aplicativo? A resposta a esta pergunta é simples: por motivos de decisão de projeto, isto não foi inicialmente planejado. Porém, para que a classe importadora passe a funcionar de maneira "dinâmica", bastam pequenas alterações na sua estrutura. O maior trabalho estaria em gerenciar o correto funcionamento da interface do aplicativo, quando um usuário alternasse entre idiomas durante a execução do mesmo.

5.1.3 Terceiro Passo: Utilizando a Classe Importadora

O terceiro e último passo é o mais simples de todos: as chamadas à classe importadora, requisitando um texto literal referente a um ID, no idioma escolhido previamente.

No formulário principal, teríamos chamadas neste teor:

```
void __fastcall TMLFMain::FormCreate(TObject *Sender)
{
   miArq->Caption= MLGerMultiLinguas::LeTxtAML("STR_ARQUIVO_CAPTION');
   miArq>Caption=
     MLGerMultiLinguas::LeTxtAML("STR_ARQUIVO_NOVO_CAPTION');
   miArqAbrir->Caption=
     MLGerMultiLinguas::LeTxtAML("STR_ARQUIVO_ABRIR_CAPTION');
}
```

Com a chamada a *MLGerMultiLinguas::LeTxtAML(ID),* estamos requisitando a classe gerente de línguas que retorne o texto literal referente ao ID sendo passado por parâmetro, novamente, no idioma previamente selecionado.

O procedimento seria o mesmo numa classe interna da aplicação qualquer, como segue o exemplo de uso na classe MLComdIOAML:

```
bool MLComdAbrirAML::LeArq(char *nomeArq)
{
   amlArq= new AmlArqIn(nomeArq);
   if (!amlArq->TestaCRC()) {  //aborta, erro de abertura
     MLGI::MsgAlerta(MLGerMultiLinguas::LeTxtAML("STR_ABREAML_CRC_ERRO'));
     return false;
   }
   if(nVer != VERSAO) { //aborta, erro de abertura
     MLGI::MsgAlert(MLGerMultiLinguas::LeTxtAML("STR_ABREAML_VER_ERRO'));
    return false;
   }
}
```

Seguindo estes três passos, o usuário será capaz de tornar o seu software internacionalizável, no que se refere à tradução de textos literais para os idiomas aos quais ele deseja que sua aplicação suporte.

Capítulo 6

Introdução

Este capítulo apresenta as conclusões deste trabalho, detalhando suas contribuições e prevendo perspectivas futuras com relação ao mesmo.

6.1 Revisão das motivações e objetivos

O objetivo principal deste trabalho foi demonstrar a utilização de um software-aplicativo na sua plenitude, o **Gerenciador MultiLinguas**, desenvolvido com o objetivo de suprir uma necessidade das empresas de tecnologia de informação e comunicação, as maiores beneficiadas pelo uso do mesmo, com relação a internacionalização.

O trabalho vem como uma importante referência ao desenvolvimento de software internacionalizável, num momento em que as empresas nacionais e internacionais buscam maior qualidade em seus produtos, com o objetivo de atingir mercados internacionais. O mesmo realiza as tarefas de gerenciamento e agrupamento de informações de textos literais de aplicativos, desonerando o código fonte de variações redundantes causadas por traduções para cada idioma aos quais um determinado aplicativo suporta.

6.2 Visão geral do trabalho

O trabalho apresentou inicialmente uma análise do mercado internacional de softwares, suas atuais condições e exigências para a aceitação de um software a nível mundial. Foi também discutida a boa posição que o Brasil tem com relação aos demais

mercados de software de paises em desenvolvimento, mas que ainda sofre de um grave déficit na sua balança comercial, que precisa ter na melhoria da qualidade dos softwares produzidos internamente, o principal elemento para virar este quadro a seu favor.

Seguiu-se de uma pequena retrospectiva sobre a internacionalização de software desde o surgimento da Internet, e as conseqüentes mudanças na ordem mundial com relação ao mercado de software, decorrentes da sua rápida expansão. Vimos que surgiram condições cada vez maiores para que empresas de software se tornem conhecidas e respeitadas mundialmente, a partir do momento que dêem a merecida importância ao tema deste trabalho.

Foram também apresentados alguns dos principais métodos usados desde os primórdios da historia da computação moderna para a internacionalização de software. Alguns se mostraram bastante eficientes no contexto geral, já outros, nem tanto, por não exonerarem do programador e do seu código fonte a responsabilidade pelo manuseio das informações e formatos de dados referentes a diferentes idiomas, entre outros fatores.

Finalizando este trabalho, foi detalhado o funcionamento do **Gerenciador MultiLinguas**, o seu correto manuseio e da classe responsável pela utilização do arquivo de línguas pelo desenvolvedor. No penúltimo capítulo, um exemplo de sua aplicação no código fonte do próprio software, onde foi dado um exemplo de software internacionalizado.

6.3 Contribuições e escopo do trabalho

Considerando os objetivos iniciais, algumas contribuições deste trabalho podem ser citadas:

- Apresentação dos principais métodos utilizados para a internacionalização de software;
- Implementação de um software responsável pelo gerenciamento dos textos literais das aplicações ao qual é aplicado;

- Demonstração das qualidades de um software corretamente internacionalizado, aplicando a internacionalização ao código fonte do próprio gerenciador proposto.

6.4 Perspectivas futuras

O trabalho ainda pode ser expandido de maneira a tornar-lhe uma ferramenta ainda mais inteligente e automatizada, simplificando ainda mais o processo de internacionalização de software. Alguns dos pontos que poderiam ser explorados são:

- Implementação de uma ferramenta que fizesse a tradução automática dos textos literais inseridos no gerenciador, para todas os idiomas disponíveis;
- Implementação de uma ferramenta que recebesse diretamente do ambiente do compilador os textos literais, criando novos identificadores a cada novo texto necessário no código fonte, de maneira que o programador fosse conduzido a confirmá-lo no momento em que são criados.
- Melhorias no formulário de mesclagem de arquivos, de maneira a dar maior poder de usabilidade ao usuário;
- Outras melhorias em geral.

Referências Bibliográficas

[DELUCCA03] DE LUCCA, José Eduardo.
SC Terá o Primeiro Centro de Internacionalização de Software do País, 19/Novembro, 2003. Geness. http://www.geness.ufsc.br/article.php?sid=96 .

[DELUCCA04] DE LUCCA, José Eduardo.

Centro de Excelência em Globalização, Internacionalização e Localização de Software (CEGIL), **Resumo Executivo**

Gomensoro Malheiros, Marcelo de.

I Seminário de Desenvolvimento de Software Livre, **Internacionalização de Projetos em Software Livre,** Maio 2004

Software Internationalization Capabilities, Maio 2004

http://www.lionbridge.com/globalization/capabilities/software-internationalization.liox?intLangID=10

The GNU C Library, Maio 2004
http://www.gnu.org/software/libc/manual/

User Influence on gettext, Maio 2004

http://www.gnu.org/software/libc/manual/html_node/Using-gettextized-software.html#Using%20gettextized%20software

FERRAMENTA DE AUXILIO AO DESENVOLVIMENTO DE SOFTWARE MULTILINGUAS

Michel Jean Grando
Curso de Graduação em Ciência da Computação (CPGCC)
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476, CEP 88040-970 – Florianópolis – SC – Brasil

grando@inf.ufsc.br

Resumo. Este artigo apresenta uma alternativa e um aplicativo para a internacionalização de software, seja ela aplicada a um software, assim como para seus manuais, tutoriais, ou outro tipo de documentação eletrônica. O aplicativo em questão será capaz de fazer o completo gerenciamento dos textos literais, para qualquer idioma ao qual deseja-se dar suporte, desonerando do código fonte de aplicação do usuário qualquer tipo de texto literal utilizado pela interface como um todo. Isto trará elegância e abrangência idiomática ao código, ao passo que a internacionalização de um software aumenta a manutenibilidade e, principalmente, abre portas para novos mercados, gerando dividendos.

Palavras-chave: Internacionalização de Software, Software Multilingue

1 INTRODUÇÃO

Um software que dê suporte multilingue é de fundamental importância para as empresas que pretendem exportar e conquistar mercados internacionais. Tal condição vem tendo uma importância crescente nos últimos anos, exigindo atenção da comunidade internacional para legislar e criar procedimentos que facilitem o processo de desenvolvimento de software internacionalizável.

O que é o software multilingue? Até onde a internacionalização de software dentro de uma empresa deve interferir no processo de produção?

Entende-se por software multilingue como sendo aquele que dá suporte a mais de um idioma, possibilitando a sua utilização em uma das línguas incorporadas pelo mesmo. Assim sendo, um software produzido no Brasil, na língua portuguesa, deverá ter também ter uma versão em outro idioma qualquer, de acordo com o país ou localidade ao qual o software se destina.

A proposta deste trabalho é um aplicativo, capaz de realizar um serviço redundante ao programador durante o desenvolvimento de uma ferramenta qualquer. Este aplicativo é capaz de reunir toda a informação textual de uma aplicação em um único arquivo seguro, para ser utilizado pelo desenvolvedor no seu código fonte, tornando-o mais elegante e fácil de manter.

Assim como a utilização deste aplicativo proposto, deve também existir todo um processo de produção e gerenciamento inteligente e organizado de software multilingue, envolvendo etapas no processo de produção do software, de maneira que possibilite e facilite a manutenção do suporte a idiomas estrangeiros. O mesmo conceito pode ser aplicado à criação de documentação, manuais, tutorias e outros.

Os softwares de médio e grande porte, onde a quantidade de informações nos mais variados idiomas cresce continuamente, tornando-se vasta e difícil de gerenciar, são os que farão melhor uso desta ferramenta.

2 INTERNACIONALIZAÇÃO DE SOFTWARE

A seguir, serão discutidas breves noções sobre a internacionalização nos dias atuais, assim como os seus objetivos.

2.1 O Brasil e a Internacionalização

No âmbito dos paises em desenvolvimento, o Brasil se encontra numa posição privilegiada mundialmente, mas precisa dar maior importância à internacionalização do software: O país é considerado um dos 7 melhores mercados de software do mundo, tendo movimentado aproximadamente 1,9 bilhões de dólares em 2001 (Softex e MIC). Entretanto gera um grande déficit nas contas externas brasileiras, que era de 5% em 2001 e está sempre crescendo em função das demandas do processo de modernização industrial e empresarial. Prevê-se que, em 3 anos, as importações de software podem chegar a 5 bilhões de dólares [DE LUCCA03].

O país e as empresas precisam definir estratégias para atingir mercados internacionais. O primeiro passo para isso é melhorar a qualidade dos produtos e a produtividade dos processos de desenvolvimento, para corresponder à competitividade internacional. Além disso, as empresas precisam internacionalizar seus produtos (adequá-los às legislações locais, providenciar documentação multilingue, etc) e preparar sua infra-estrutura interna para uma nova realidade [DE LUCCA03].

2.2 Internacionalização de Software

A internacionalização faz com que os softwares, manuais, tutoriais e demais documentação eletrônica, aceitem diversos formatos multilingues. Ou seja, o produto em questão deve se adaptar a exigências locais, incluindo: textos literais traduzidos, formatos de moedas e outras adaptações necessárias para atender aos requisitos culturais, legais e técnicos de mercados regionais.

São objetivos que se quer alcançar com a internacionalização:

- Garantir total funcionalidade em qualquer idioma: a aplicação tem seus textos literais originais traduzidos, formatos de moeda adaptados, entre outros, de acordo com cada um dos idiomas suportados;
- Total transparência para o usuário na mudança de idiomas de um aplicativo em tempo de execução;
- Centralizar o gerenciamento de textos literais fora da aplicação, de maneira que não seja necessário ter acesso ao código fonte da aplicação para que a mesma possa, por exemplo, alterar um texto literal. Isto, pois estas informações são redundantes para o desenvolvimento do código da aplicação;
- Reduzir os esforços e os custos de desenvolvimento e, principalmente, manutenção. A internacionalização, aplicada corretamente, deixa o código fonte mais robusto e livre de erros.
- Facilitar o trabalho dos tradutores, de maneira que eles possam ter acesso mais facilmente aos textos literais do programa;
- Oferecer um “paradigma de programação para software internacional”, que sirva de base para o

desenvolvedor fazer um bom uso da internacionalização na sua plenitude, tendo como resultado um código robusto, eficaz e elegante, assim como com relação aos demais processos que envolvem uma empresa que deseja se tornar internacional.

3 FERRAMENTAS ATUAIS

A seguir, serão comentados alguns dos principais métodos e ferramentas utilizadas para internacionalização até a presente data.

3.1 Biblioteca Padrão C

Primeira ferramenta utilizada para internacionalização, presente na biblioteca padrão do compilador C. Esta biblioteca é padronizada pelo POSIX para definir *locales*, que atuarão na tradução de mensagens e conversão de formatos: *strings,* classificação de caracteres, valores numéricos e monetários.

A especificação de *locales* é simples: uma chamada ao método *setlocale*() da biblioteca C, passando-se por parâmetro a categoria que deseja-se mudar o *locale* atual, e qual *locale* usar.

Assim, teríamos algo como o exemplo a seguir:
`Setlocale(LC_CTYPE,C)`, onde **LC_TYPE** é a categoria a qual se deseja modificar o *locale*, e **C** é *locale* padrão do compilador C++ Builder

3.2 Compilador C++ Builder: Arquivos *Resource*

A tradução de mensagens é feita separadamente do uso do *locale*, utilizando-se de um recurso disponível no compilador C++ Builder: os arquivos *resource*. Estes arquivos podem ser utilizados da seguinte maneira: cria-se um arquivo *resource* para cada idioma a ser disponibilizado no software. Insere-se manualmente os textos literais neste arquivo, traduzidos para o idioma ao qual o arquivo representa, atribuindo-os a identificadores, que são os mesmos identificadores utilizados nos arquivos dos outros idiomas.

Exemplo da criação de dois arquivos de recurso: o primeiro para o idioma português, o segundo para o idioma inglês:

```
STRINGTABLE {
   STR_APPAM_STRING, "TEXTO DO
STRING"
......
}
STRINGTABLE {
   STR_APPAM_STRING, "STRING
TEXT"
......
}
```

3.3 Gettext: Software Livre

O pacote Gettext, de código fonte aberto, utilizado para criação, uso e manutenção de catálogos de textos literais, é disponível para o Linux. Ele age em conjunto com a GLIBC, um dos pacotes que geralmente vem incluído ao pacote do sistema operacional, ou também pode ser distribuído como uma biblioteca individual.

O seu uso é baseado na associação de catálogos de textos literais à categoria **LC_MESSAGES** do *locale*. Com isto, ao chamar-se o método *gettext*(), passando por parâmetro um identificador para o texto literal desejado, o mesmo devolverá o texto literal no idioma especificado. Caso o catálogo ou mesmo a tradução para o idioma não seja encontrado, o próprio parâmetro passado para o método é retornado como resposta.

Assim, ao invés de se utilizar o trecho de código a seguir para imprimir uma mensagem na tela:

```
Printf("Isto é um teste");
```

Teríamos isto:

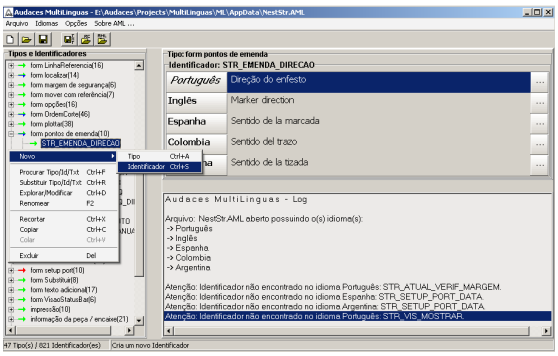
```
Printf(gettext("Isto é um teste"));
```

4 O APLICATIVO : O GERENCIADOR MULTILINGUAS

O **Gerenciador MultiLinguas** é um aplicativo de cunho administrativo, como o próprio nome sugere.

Através dele, será feita o gerenciamento dos textos literais das aplicações, sendo salvo o seu conteúdo em um arquivo texto codificado. Para ilustrar melhor, este arquivo pode ser considerado como sendo a base de dados de textos multilingue de uma determinada aplicação, onde cada texto literal terá sua tradução para o seu respectivo idioma. Assim sendo, este arquivo será usado como fonte para busca de todos os textos literais necessários ao código fonte, no idioma desejado, pelo aplicativo a que se deseja dar suporte multilingue.

Somente através do **Gerenciador MultiLinguas** poderá ser feita qualquer alteração neste arquivo, seja nos idiomas, ou nos textos literais. Assim, o programa centraliza em si o poder de manuseio dos dados, garantindo a coesão dos mesmos. A coerência só não pode ser assumida, na medida em que o programa não tem a função de um tradutor, ou seja, ele não realiza traduções automáticas dos dados inseridos. Isto fica ao encargo do usuário, para que o mesmo possa fazer as traduções a seu critério.



Em síntese, o funcionamento do aplicativo se faz com o uso de 3 classes de dados:

- IDENTIFICADOR (ID): É uma sequência única de caracteres alfanuméricos (*string*) que designa, como o próprio nome sugere, um identificador para um determinado TXT da aplicação;
- TIPO (TP): Nada mais é que um agrupamento de Ids. Possibilita ao usuário uma melhor organização das informações;
- TEXTO LITERAL (TXT): são também sequências de caracteres alfanuméricos (*string*), que correspondem aos textos literais da aplicação do usuário, aos quais este último deseja tornar internacionalizável. Este tipo de dado é atribuído a um ID, tendo uma versão em cada idioma disponível.

4.1 As funcionalidades do Gerenciador

Como todo aplicativo de cunho gerencial que comporta um grande número de dados, o Gerenciador MultiLinguas possui uma série de ferramentas de auxilio.

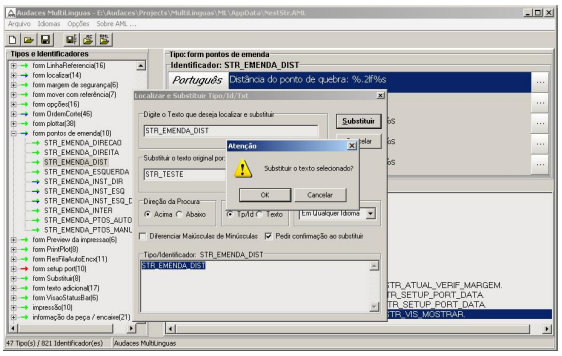
Podemos separar as funcionalidades do programa em 3 grandes grupos:

- Funcionalidades de Edição Geral;
- Funcionalidades Sobre Idiomas;
- Funcionalidades Sobre Arquivo.

4.1 Funcionalidades de Edição Geral

Funcionalidades estas que compreendem as ferramentas de uso geral, que serão demandadas no cotidiano do uso do aplicativo:

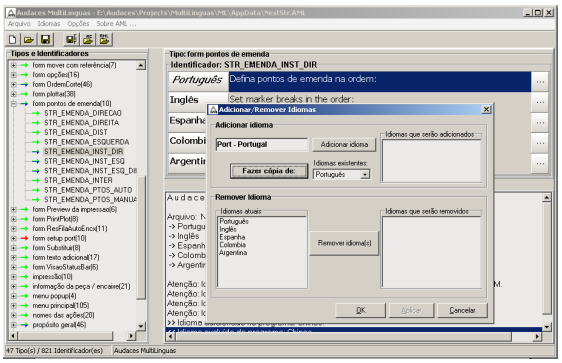
- Criação de Novos TPs e IDs;
- Procura (Localização) de TPs, IDs e TXTs;
- Substituição de TPs, IDs e TXTs;
- Renomear TPs e IDs;
- Excluir TPs e IDs;
- Copiar/Recortar/Colar TPs e IDs.



4.2 Funcionalidades Sobre Idiomas

Define as ferramentas necessárias para manuseio de idiomas. Com estas ferramentas, faz-se qualquer operação sobre os idiomas aos quais deseja-se dar suporte a um determinado software.

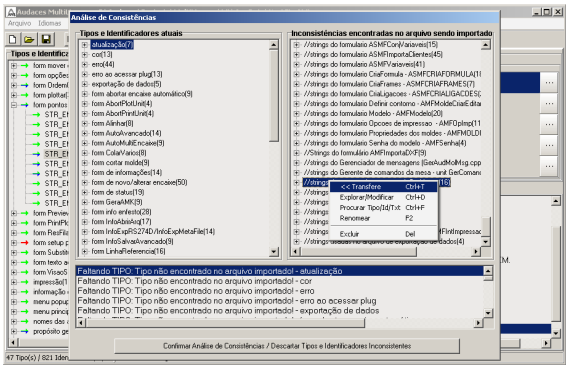
- Criação e Exclusão de Idiomas;
- Criar Cópia de Idiomas;
- Mudança de Nome de Idiomas.



4.3 Funcionalidades Sobre Arquivos

Define ferramentas sobre operações em arquivos. Além das operações básicas de abrir e salvar arquivos, é também oferecida uma ferramentas para migração do modo de internacionalização utilizando-se arquivos *Resource*, e uma ferramenta para mesclagem de arquivos no formato do gerenciador.

- Ações Genéricas: Novo, Abrir e Salvar Arquivos;
- Importar/Mesclar Arquivo Resource;
- Mesclar Arquivo AML.



4.4 A Classe Importadora

Uma classe ou *library* é provida juntamente com o Gerenciador MultiLinguas. Esta classe é necessária para utilização do arquivo gerado pelo gerenciador no código fonte do usuário.

Ela faz o processo inverso da cifragem do arquivo feito no momento em que um arquivo é salvo no gerenciador, decifrando-o para que seu conteúdo possa ser lido corretamente.

5 INTERNACIONALIZANDO UM SOFTWARE

Os procedimentos necessários a internacionalização de um software são simples e diretos.

Os passos a seguir, são aplicáveis a um software não internacionalizado, que já se encontra em processo de desenvolvimento. Mesmo assim, a sequência dos passos em softwares que estarão dando suporte multilingue desde a sua concepção, pouco mudará.

- **Primeiro Passo:** Geração de um arquivo (*.aml) utilizando o Gerenciador MultiLinguas, criando-se TPs, IDs e, por último, inserindo-se os textos literais da aplicação traduzidos para cada um dos idiomas suportados, colocando-os no seu devido ID;
- **Segundo Passo:** Criação de uma classe que fará uso da classe importadora, detalhada no capitulo anterior. Esta classe terá acesso ao arquivo MultiLinguas previamente gerado no **Primeiro Passo**;
- **Terceiro Passo:** Chamadas à classe criada no **Segundo Passo**, para que se obtenha um texto literal desejado, no idioma escolhido. Assim sendo, não mais teremos textos literais diretamente no código fonte da aplicação, mas apenas chamas a um método, requisitando que ele retorne um *string* para um dado identificador. Claramente, o programador deve ter conhecimento de quais IDs representam quais TXTs, para que não sejam obtidas *strings* incorretas.

6 CONCLUSÕES

O objetivo principal deste trabalho foi a apresentação de uma alternativa para as ferramentas de internacionalização presentes atualmente no mercado. Esta nova ferramenta busca extrair toda a parte textual apresentada na interface do código fonte dos aplicativos de usuário, de maneira a permitir a fácil internacionalização e alta manutenibilidade dos softwares onde for aplicado.

Considerando os objetivos iniciais, algumas contribuições deste trabalho podem ser citadas:

- Apresentação dos principais métodos utilizados para a internacionalização de software;
- Implementação de um software responsável pelo gerenciamento dos textos literais das aplicações ao qual é aplicado;

7 REFERÊNCIAS

[DELUCCA03] DE LUCCA, José Eduardo.
SC Terá o Primeiro Centro de Internacionalização de Software do País, 19/Novembro, 2003. Geness. http://www.geness.ufsc.br/article.php?sid=96 .

[DELUCCA04] DE UCCA, José Eduardo.
Centro de Excelência em Globalização, Internacionalização e Localização de Software (CEGIL), **Resumo Executivo**

Gomensoro Malheiros, Marcelo de.
I Seminário de Desenvolvimento de Software Livre, **Internacionalização de Projetos em Software Livre,** Maio 2004

Software Internationalization Capabilities, Maio 2004
http://www.lionbridge.com/globalization/capabilities/software-internationalization.liox?intLangID=10

The GNU C Library, Maio 2004
http://www.gnu.org/software/libc/manual/

User Influence on gettext, Maio 2004
http://www.gnu.org/software/libc/manual/html_node/Using-gettextized-software.html#Using%20gettextized%20software