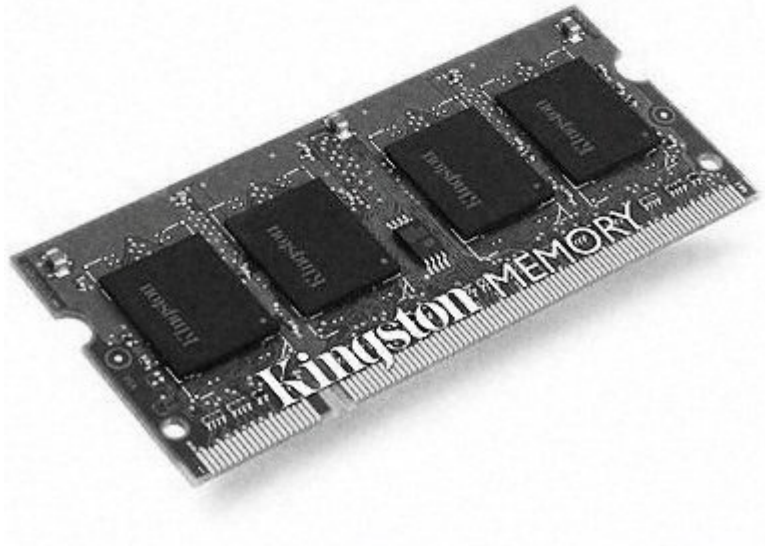
FURB



Ferramenta de gerenciamento de metadados para banco de dados Oracle

Acadêmico: Charles Ristow

Orientadora: Joyce Martins

FURB




Roteiro

- **Introdução**
- **Objetivos do trabalho**
- **Conceitos básicos**
- **Contexto atual**
- **Requisitos do problema**
- **Técnicas e ferramentas utilizadas na especificação**
- **Especificação da ferramenta**
- **Técnicas e ferramentas utilizadas na implementação**
- **Operacionalidade da implementação**
- **Resultados e discussão**
- **Conclusão**
- **Extensões**

Introdução

- **Mercado exige qualidade e velocidade na entrega dos serviços/produtos oferecidos pelas empresas de informática.**
- **A constante competição motiva as empresas a pesquisar e aperfeiçoar técnicas e ferramentas.**
- **Alterações no ciclo de desenvolvimento de software influencia na redução de custos e tempo.**
- **Soluções adotadas:**
  - **Automatização de tarefas;**
  - **Reutilização de código.**

Objetivos do trabalho

- **Disponibilizar uma ferramenta para criação e manutenção do metadados para banco de dados Oracle, gerando *scripts* SQL;**

- **Através da utilização de *templates*, permitir gerar classes de acesso ao banco de dados para a linguagem Object Pascal e Java;**

Conceitos básicos

- **banco de dados relacional;**
- **metadados;**
- ***templates*;**

- **motor de *templates*;**
- **geração de código ativa;**

- **geração de código passiva.**

Contexto Atual

- **Power Designer;**

- **Rational Rose;**

- **Oracle SQL Developer;**
- **Ferramenta para geração automática de código Java;**

- **Gerador de código baseado em dicionário de dados.**




Requisitos do problema

- **Requisitos funcionais:**
  - **permitir acesso ao banco de dados Oracle;**
  - **permitir identificação do metadados do banco acessado;**
  - **criar e/ou alterar metadados e gerar *scripts* SQL para realizar tal procedimento;**
  - **permitir configurar o banco de dados e arquivo de *template*;**
  - **indicar em qual diretório serão gerados os arquivos;**
  - **permitir selecionar para quais tabelas será gerado código;**
  - **permitir gerar código para camada de acesso a dados nas linguagens Object Pascal e Java.**

Requisitos do problema

- **Requisitos não-funcionais:**
  - **utilizar a ferramenta Borland Delphi 7 para desenvolvimento da interface;**
  - **interface deve ser de fácil compreensão permitindo acesso a todas as funcionalidades através de botões de atalho e menus;**
  - **as consultas ao banco devem ser rápidas. Tempo inferior a 30 segundos;**
  - **utilizar *templates* para formatação do código de saída.**

Técnicas e ferramentas utilizadas na especificação

- **Especificação da GLC do motor de *templates* através da notação BNF;**

- **Definição dos diagramas de casos de uso, de classes e de seqüência da UML através da ferramenta Enterprise Architect 4.5.**

Especificação da ferramenta

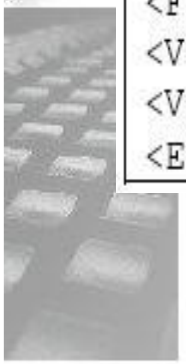
- **Linguagem de *templates*:**
  - ***tokens*;**
  - **Gramática e ações semânticas.**
- ***Template* e código de saída esperado para linguagem Object Pascal;**
- **Diagrama de casos de uso;**
- **Diagrama de classes;**
- **Diagrama de seqüência.**

DEFINIÇÕES REGULARES	*TOKENS*

char : [^<]
char2 : [a-z A-Z]
digit : [0-9]
	//palavras reservadas
	var : "<" {char2} ({char2} | {digit} | _)* ">"
	
	template = var : "<template>"
	endtemplate = var : "<endtemplate>"
	tablename = var : "<tablename>"
	fields = var : "<fields>"
	endfields = var : "<endfields>"
	fieldname = var : "<fieldname>"
	fieldtype = var : "<fieldtype>"
	
	//identificadores
	identifier : {char}+
	
	//símbolos especiais
	"<"

Tokens






```
<Template> ::= template #1 <Body> endtemplate #2
<Body>     ::= <Text> <Body_>
<Body_>    ::= î | <Body>
<Text>     ::= identifier #3 | <TableName> | <Fields> | "<" #3
<TableName>::= tablename #4
<Fields>   ::= fields #5 <Value> endfields #6
<Value>    ::=  <Element> <Value_>
<Value_>   ::= î | <Value>
<Element>  ::= identifier #7 | fieldname #7 | fieldtype #7 | <TableName> | "<" #7
```

Gramática e ações semânticas da linguagem

```
<template>

unit uclaCad<tablename>;

interface
  uses  SqlExpr, SysUtils;

type
  TCad<tablename> = class
  private
    <fields> f<fieldname> : <fieldtype> ; <endfields>
  public
    Constructor Create(const coSqlConection : TSQLConnection); virtual; abstract;
    Destructor Destroy; virtual; abstract;
    <fields> property <fieldname> : <fieldtype> read f<fieldname>;
  end;

implementation
end.

<endtemplate>
```

***Template* para linguagem Object Pascal**

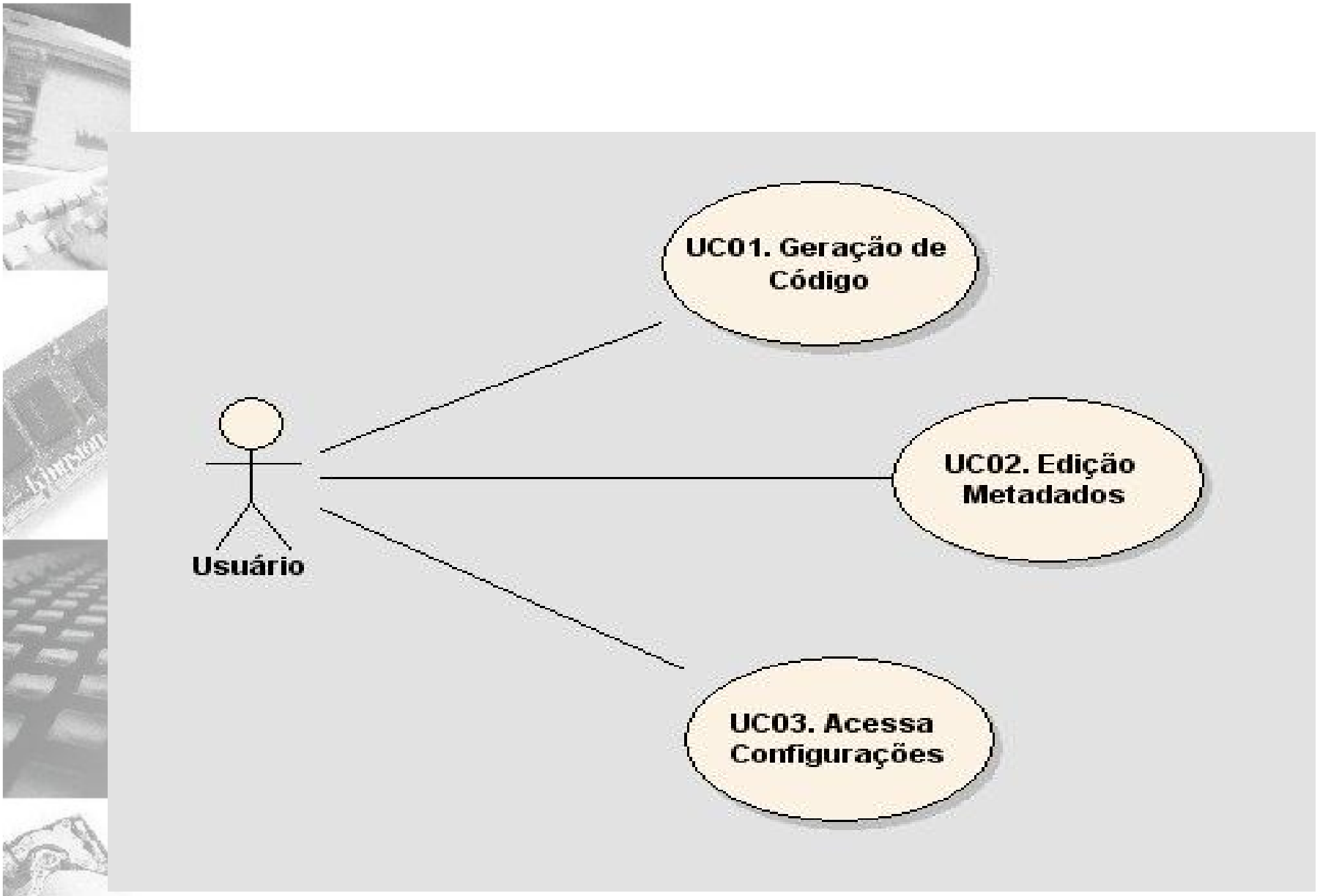


Diagrama de casos de uso

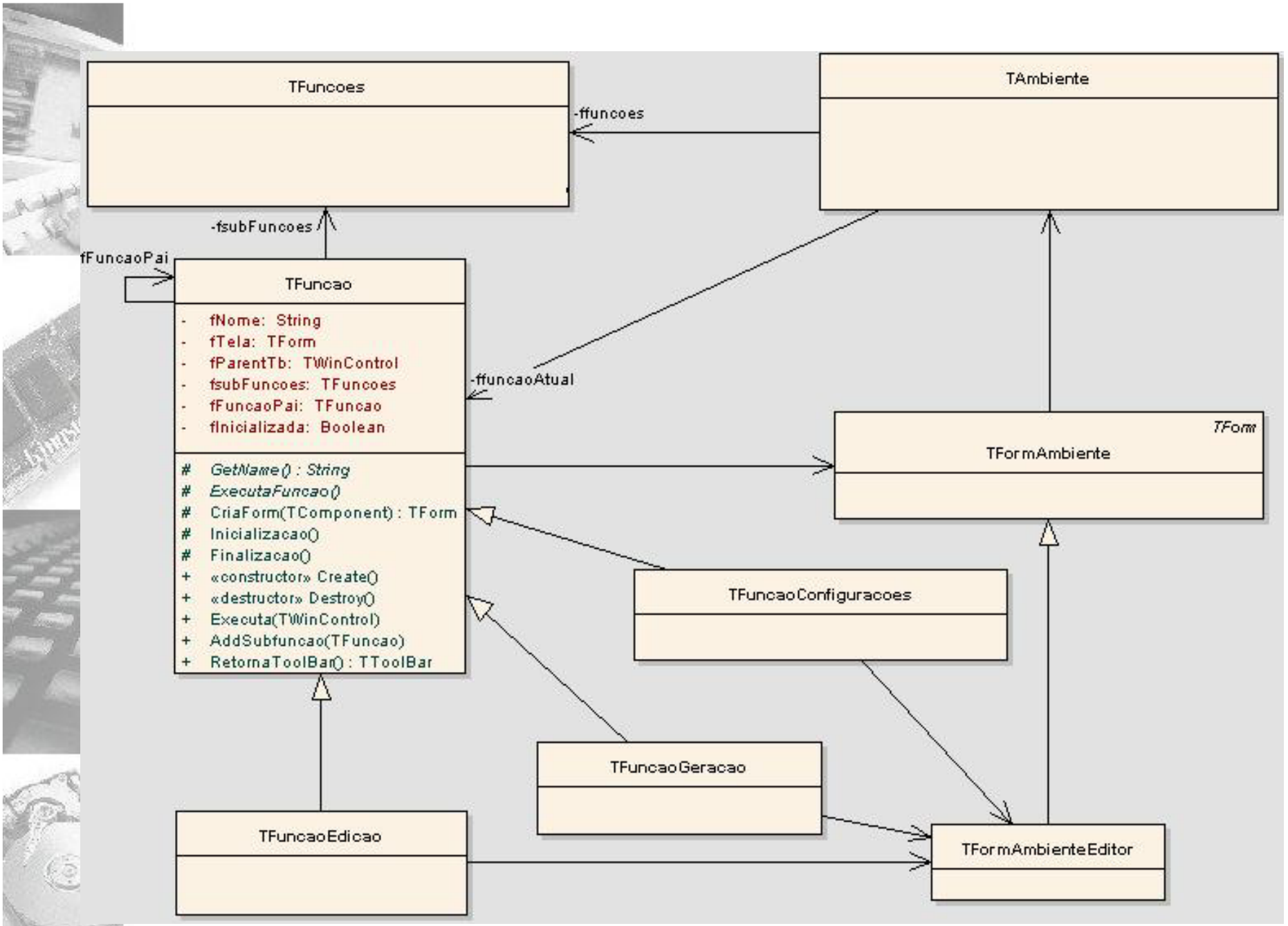


Diagrama de classes: controle das funcionalidades

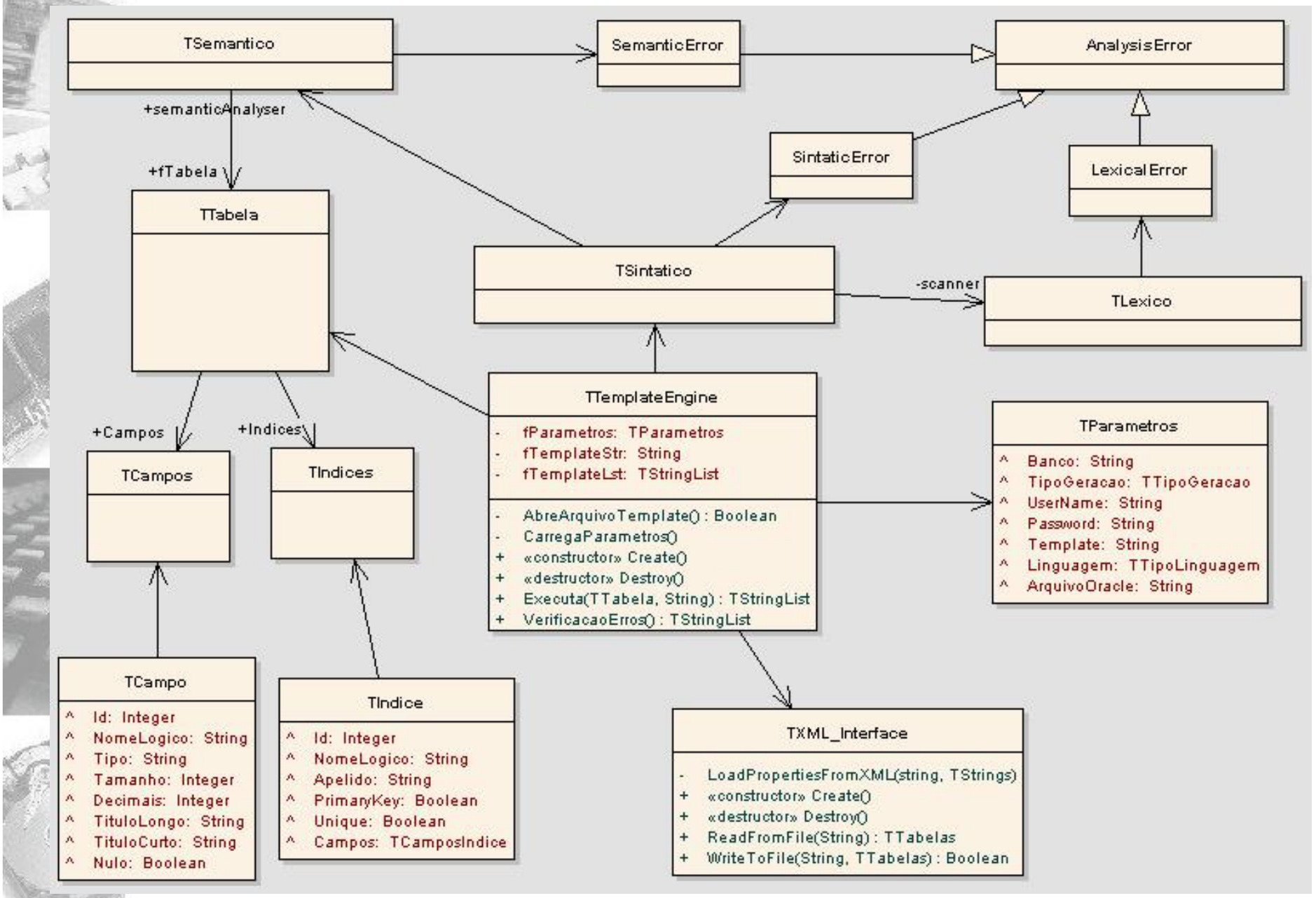


Diagrama de classes: geração de código

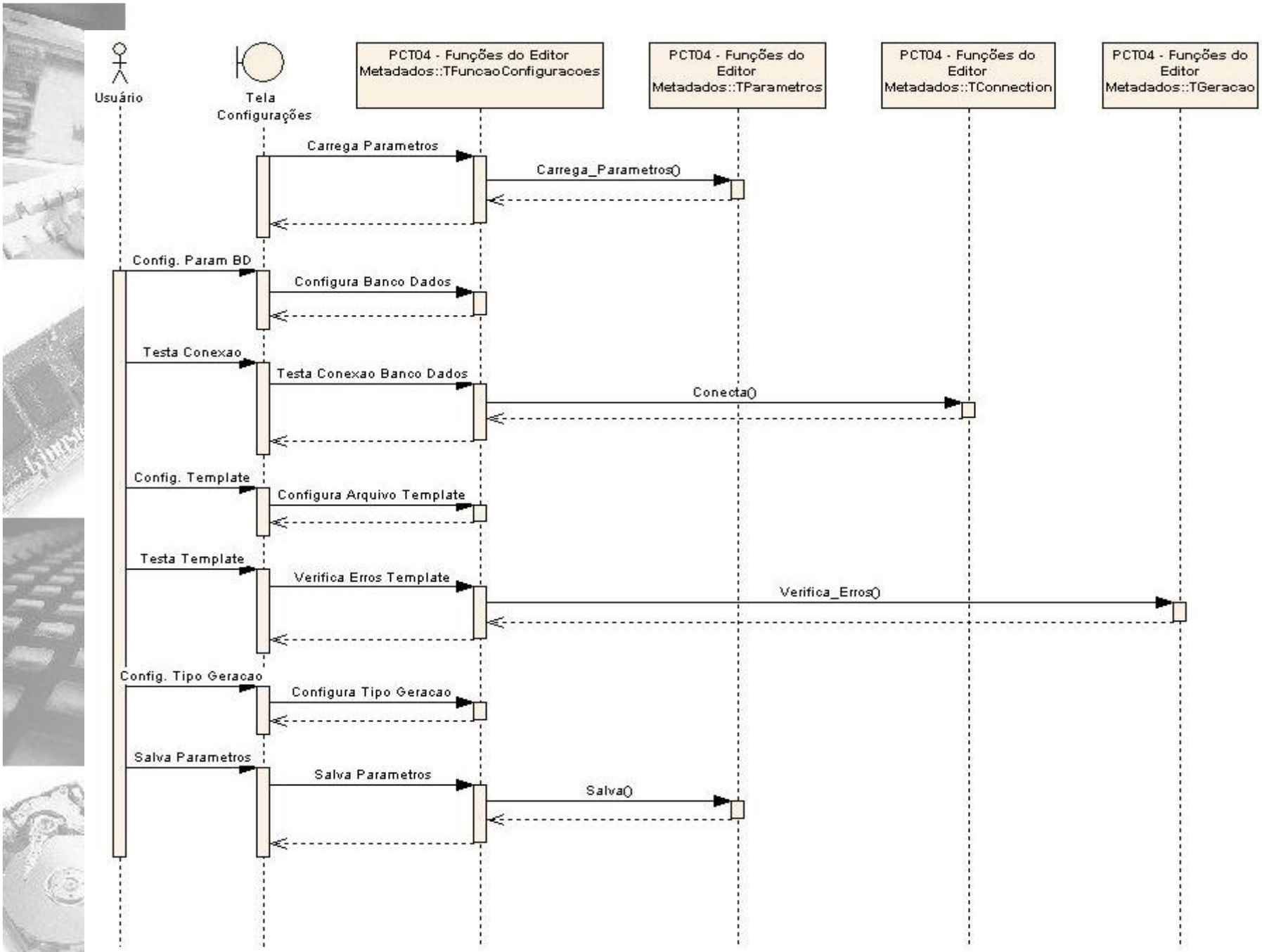


Diagrama de seqüência: configuração parâmetros

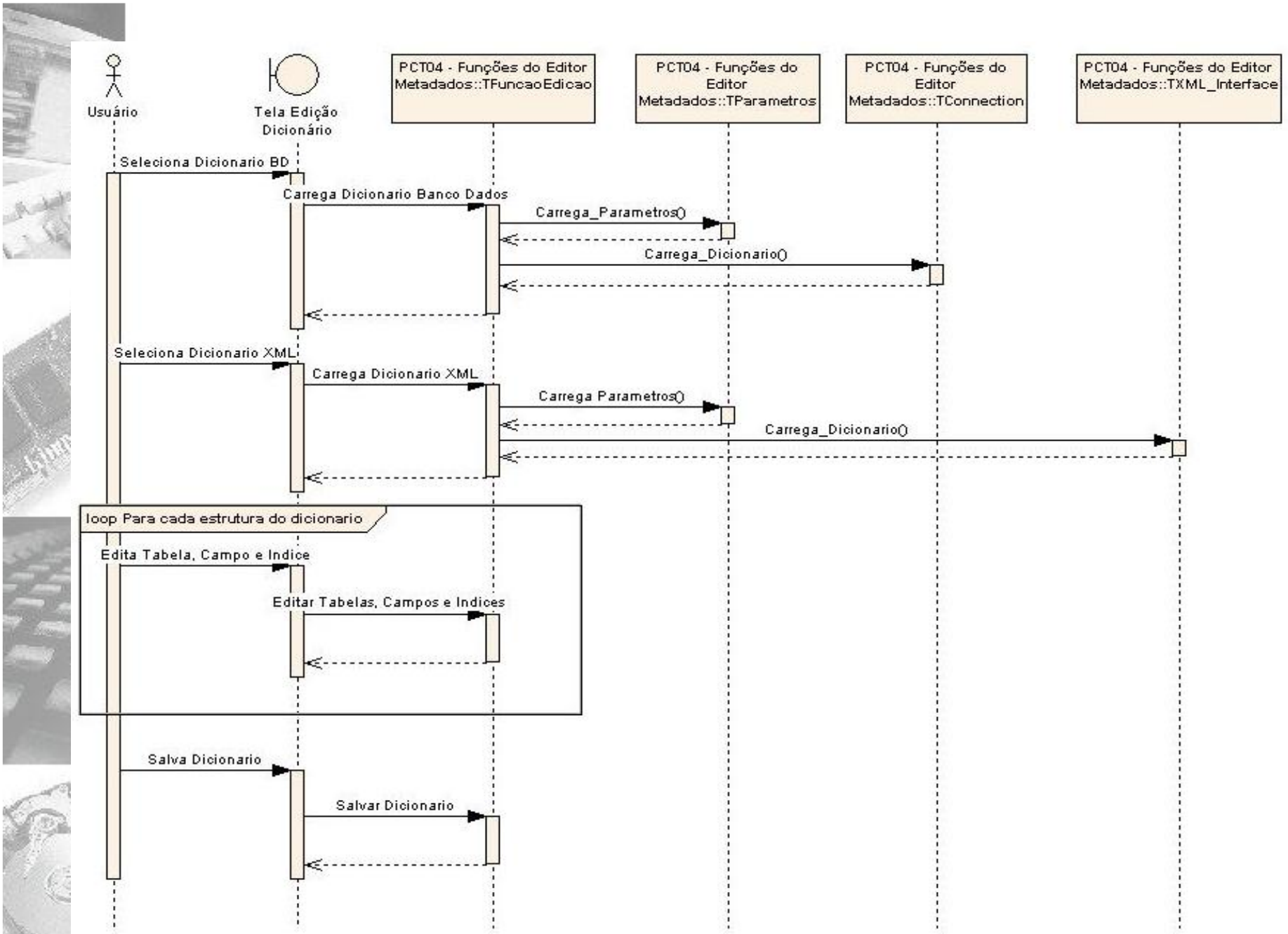


Diagrama de seqüência: criação/edição metadados

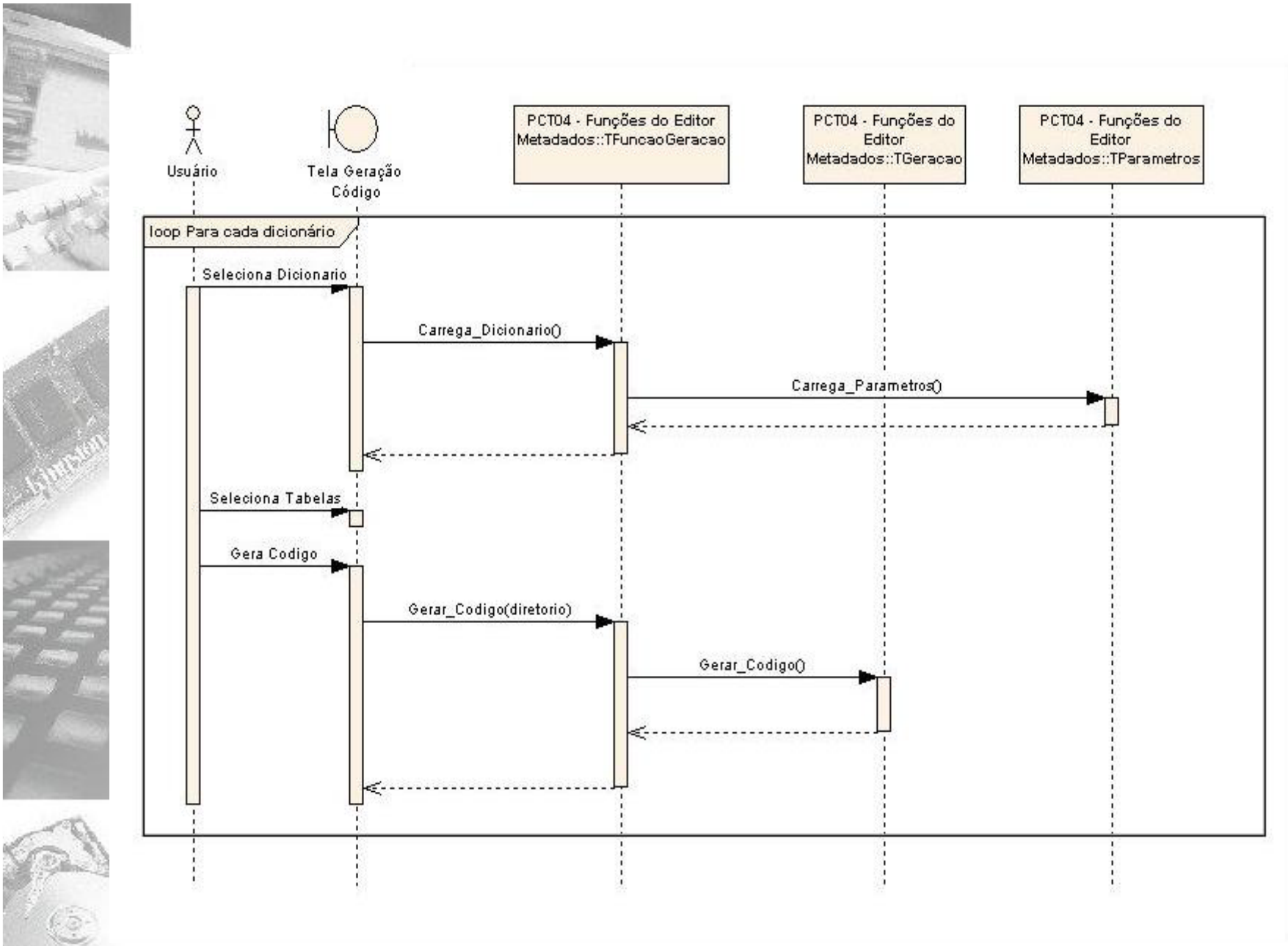


Diagrama de seqüência: geração de código

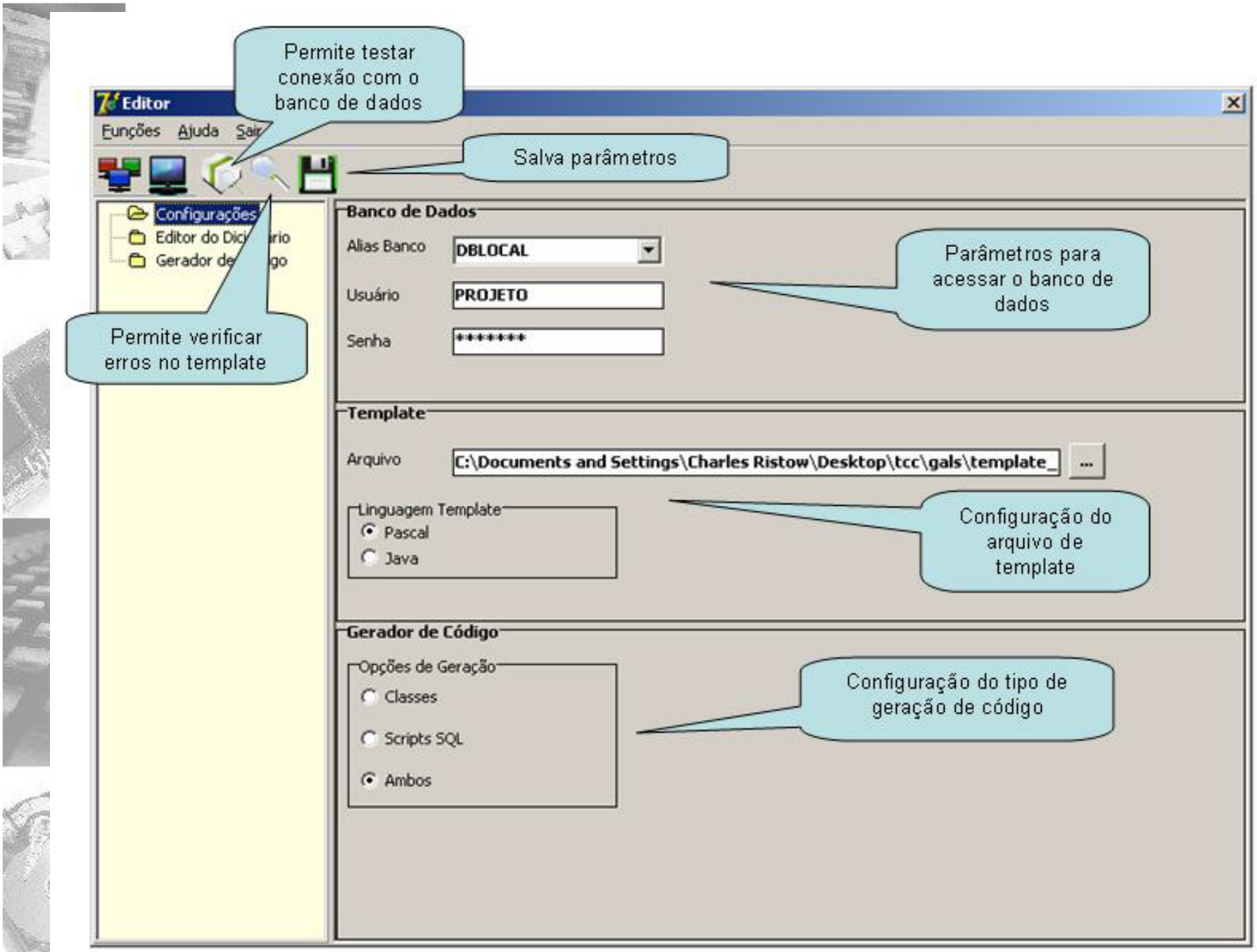
Técnicas e ferramentas utilizadas na implementação

- **Técnica de programação: Orientação a objetos com a linguagem Object Pascal;**

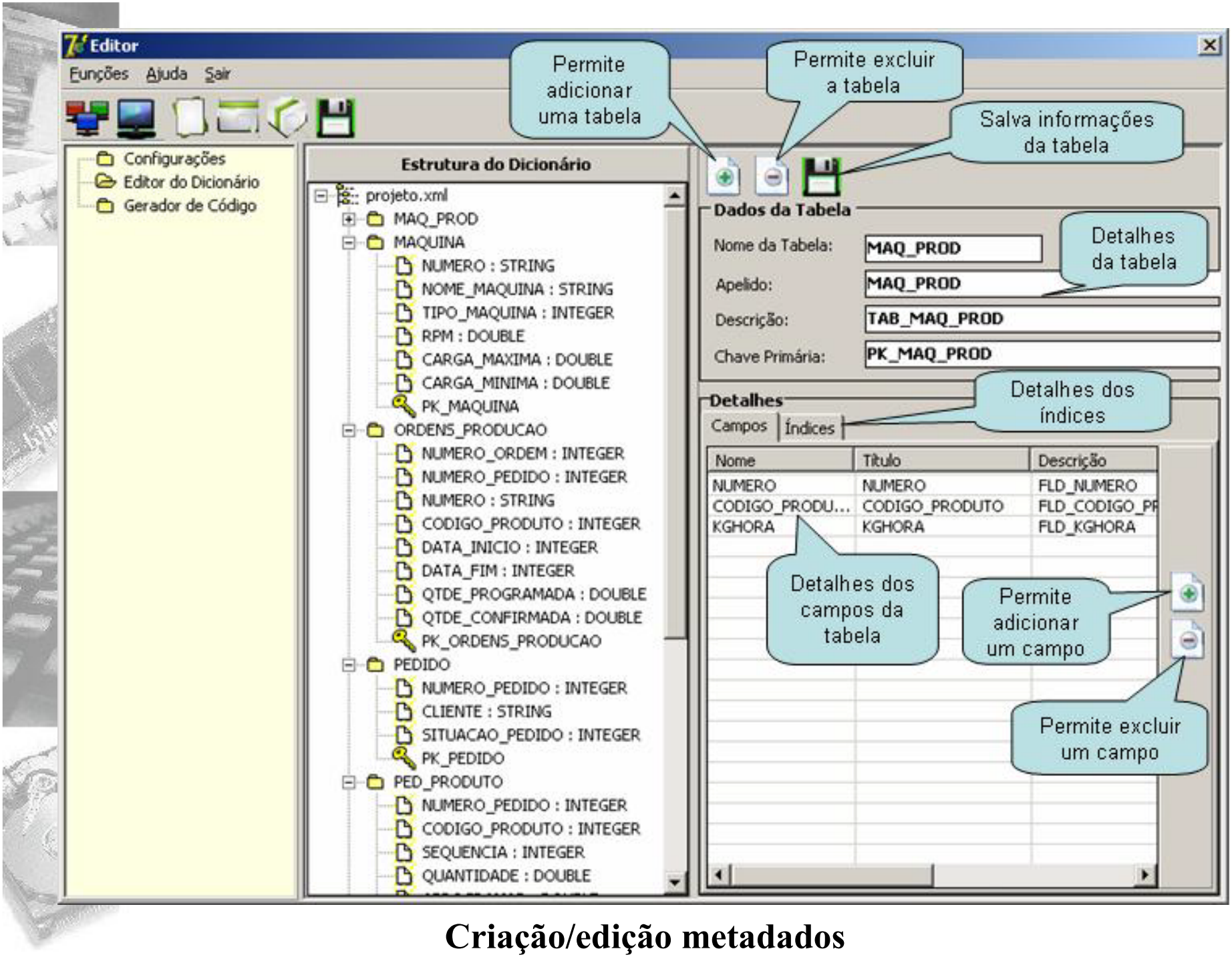
- **Ferramentas utilizadas:**
  - **Borland Delphi 7**
  - **GALS**

Operacionalidade da implementação

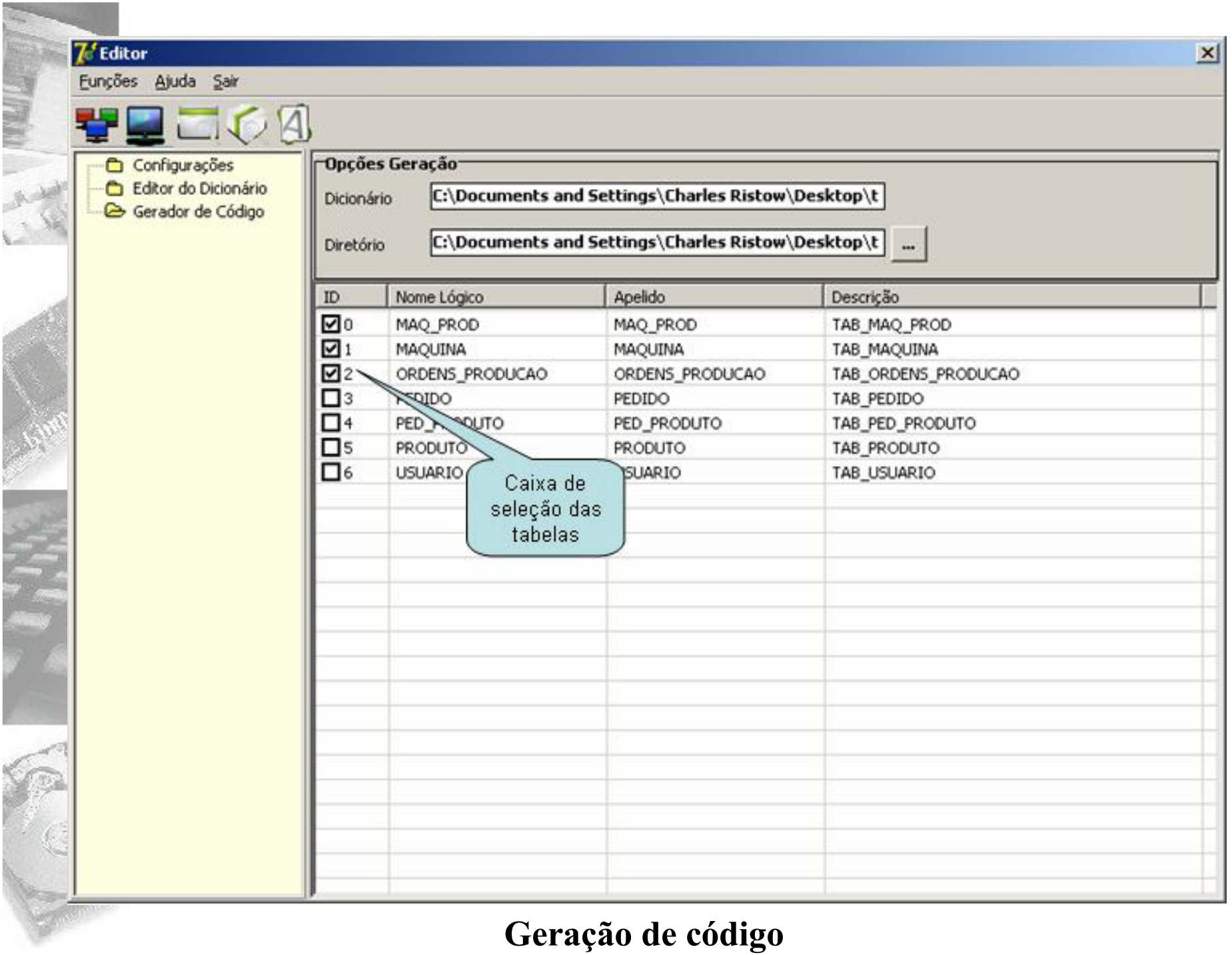
Visão geral do aplicativo



Configuração parâmetros



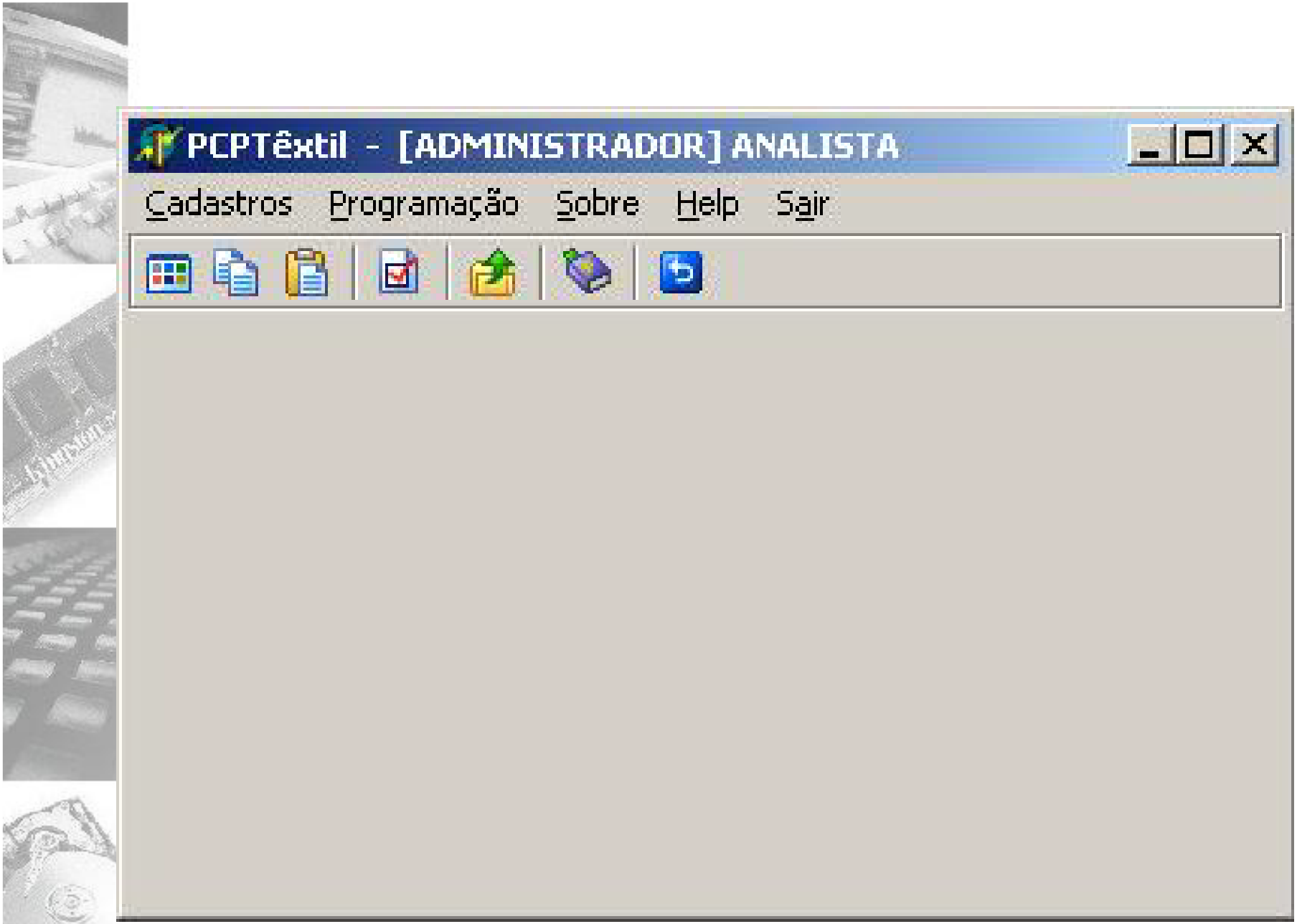
Criação/edição metadados



Geração de código

Resultados e discussão

- **Utilização de um caso real para validar a operacionalidade da implementação:**
  - **definição do metadados**
  - **geração de código: classes para a linguagem Object Pascal**
  - **geração de código: *scripts* SQL**
  - **criação do banco de dados**
  - **implementação do aplicativo: expansão de código**



Tela principal do sistema implementado




```
unit uclaCadMAQUINA;

interface
uses
  uDefinicao, SqlExpr, SysUtils;
type
  TCadMAQUINA = class(TCadastroGenerico)
  private
     fNUMERO : STRING ;
     fNOME_MAQUINA : STRING ;
     fCARGA_MAXIMA : DOUBLE ;
     fCARGA_MINIMA : DOUBLE ;
  public
    Constructor Create(const coSqlConection : TSQLConnection); Override;
    Destructor Destroy; Override;
    property NUMERO : STRING read fNUMERO write fNUMERO;
    property NOME_MAQUINA : STRING read fNOME_MAQUINA write fNOME_MAQUINA;
    property CARGA_MAXIMA : DOUBLE read fCARGA_MAXIMA write fCARGA_MAXIMA;
    property CARGA_MINIMA : DOUBLE read fCARGA_MINIMA write fCARGA_MINIMA;
    procedure Insere; override;
    function Exclui: Boolean; override;
  end;

implementation

uses DB;

constructor TCadMaquina.Create(const coSqlConection: TSQLConnection);
begin
  inherited;
end;

destructor TCadMaquina.Destroy;
begin
  inherited;
end;

function TCadMaquina.Exclui: Boolean;
begin
end;

procedure TCadMaquina.Insere;
begin
end;

end.
```

Classe de acesso a dados

Resultados e discussão

- **Resultados obtidos:**
  - **facilidade para criação e manutenção do banco de dados**
  - **geração de código fonte**
  - **padronização e qualidade do código**
  - **agilidade na implementação e suporte do sistema**

Resultados e discussão

Aplicativos / Características	Power Designer	Rational Rose	Oracle SQL Developer	Santos (2005)	Coelho (2006)	EditDB
diagramas UML	S	S	N	N	N	N
suporte a vários BD	S	S	S	N	S	N
scripts SQL DDL	S	S	S	N	N	S
scripts SQL DML	N	N	S	N	N	S
engenharia reversa BD	S	S	S	S	S	S
geração de código fonte	S	S	N	S	S	S
utilização de *templates*	N	N	N	N	S	S
suporte a várias linguagens	S	S	N	N	N	S
distribuição gratuita	N	N	S	N	N	S

Comparativo das características

Conclusão

- **A ferramenta permite o usuário através de um editor, criar ou alterar a estrutura de um banco de dados;**
- **Permite a geração de *scripts* SQL e geração de código para classes de acesso a dados;**
- **Flexibilidade para formatação do código de saída com a utilização de *templates*;**
- **Foi construído um motor de *templates* para a linguagem Object Pascal;**
- **Agilidade e padronização na geração de código.**

Extensões

- **permitir a utilização de outros bancos de dados como o PostGreeSQL, SQLServer e o MySQL;**
- **permitir o reconhecimento das chaves estrangeiras de cada tabela no banco de dados;**
- **permitir construir e editar o arquivo de *template* na própria ferramenta;**
- **alterar a linguagem do motor de *templates* de forma que o *template* possua variáveis para informar a linguagem de programação desejada e seus respectivos tipos de dados;**

Extensões

- **alterar a linguagem do motor de *templates* de forma que possa criar os *scripts* SQL juntamente com a geração das classes;**

- **permitir o usuário executar os *scripts* SQL na própria ferramenta;**

- **permitir integração com diagramas DER de ferramentas comerciais como o Power Designer e o Rational Rose.**