6
Referências

[1] RAMESH, Balasubramaniam; JARKE, Matthias. Toward reference models for requirements traceability. **Software Engineering, IEEE Transactions on**, v. 27, n. 1, p. 58-93, 2001.

[2] NAIR, Sunil; DE LA VARA, Jose Luis; SEN, Sagar. A review of traceability research at the requirements engineering conference re@ 21. In:**Requirements Engineering Conference (RE), 2013 21st IEEE International**. IEEE, 2013. p. 222-229.

[3] SAYÃO, Miriam; BREITMAN, Karin Koogan. Gerência de Requisitos. **Mini-Curso do**, 2005.

[4] SAYÃO, Miriam; DO PRADO LEITE, Julio Cesar Sampaio. Rastreabilidade de requisitos. **RITA**, v. 13, n. 1, p. 57-86, 2006.

[5] GLINZ, Martin; WIERINGA, Roel. RE@ 21 spotlight: most influential papers from the requirements engineering conference. In: **Requirements Engineering Conference (RE), 2013 21st IEEE International**. IEEE, 2013. p. 368-370. .

[6] RAMESH, Bala et al. Lessons learned from implementing requirements traceability. **Crosstalk–Journal of Defense Software Engineering**, v. 8, n. 4, p. 11-15, 1995.

[7] ANTONIOL, Giuliano et al. Recovering traceability links between code and documentation. **Software Engineering, IEEE Transactions on**, v. 28, n. 10, p. 970-983, 2002.

[8] RAJA, U.; KAMRAN, K. **Framework for Requirement Traceability**. Blekinge Institute of Technology, Ronneby, Sweden, 2008. Master Thesis Series No. MSE-2008:06

[9] RAMESH, Balasubramaniam et al. Implementing requirements traceability: a case study. In: **Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on**. IEEE, 1995. p. 89-95 .

[10] GOTEL, Orlena et al. Traceability fundamentals. In: **Software and Systems Traceability**. Springer London, 2012. p. 3-22.

[11] REMPEL, Patrick; MADER, P.; KUSCHKE, Tobias. An empirical study on project-specific traceability strategies. In: **Requirements Engineering Conference (RE), 2013 21st IEEE International**. IEEE, 2013. p. 195-204.

[12] MADER, Patrick; GOTEL, Orlena; PHILIPPOW, Ilka. Motivation matters in the traceability trenches. In: **Requirements Engineering Conference, 2009. RE'09. 17th IEEE International**. IEEE, 2009. p. 143-148.

[13] **Rastreabilidade**, in Dicionário Priberam da Língua Portuguesa. Disponível em: <http://www.priberam.pt/dlpo/rastreabilidade>. Acesso em: 29 jul. 2014.

[14] HERNANDES P. Emprego do sufixo “-dade” (“-idade”). In: **Professor Paulo Hernandes**. Disponível em: <http://www.paulohernandes.pro.br/dicas/001/dica165.html>. Acesso em: 29 jul. 2014.

[15] Rastreável, in **Dicionário Priberam da Língua Portuguesa**. Disponível em: <http://www.priberam.pt/dlpo/rastre%C3%A1vel>. Acesso em: 29 jul. 2014.

[16] Rastrear, in Dicionário Priberam da Língua Portuguesa. Disponível em: <http://www.priberam.pt/dlpo/rastrear>. Acesso em: 29 jul. 2014.

[17] GOTEL, Orlena CZ; FINKELSTEIN, Anthony CW. An analysis of the requirements traceability problem. In: **Requirements Engineering, 1994., Proceedings of the First International Conference on**. IEEE, 1994. p. 94-101.

[18] HAMILTON, V. L.; BEEBY, M. L. Issues of traceability in integrating tools. In: **Tools and Techniques for Maintaining Traceability During Design, IEE Colloquium on**. IET, 1991. p. 4/1-4/3.

[19] WOHLIN, Claes et al. (Ed.). **Engineering and managing software requirements**. Springer Science & Business Media, 2005**.**

[20] MORAES, E. A. **Utilização de uma estratégia para identificação de fontes de informação na fase de elicitação**. 2008. 147 f. Dissertação (Mestrado em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2008.

[21] KANNENBERG, Andrew; SAIEDIAN, Hossein. Why software requirements traceability remains a challenge. **CrossTalk The Journal of Defense Software Engineering**, v. 22, n. 5, p. 14-19, 2009.

[22] NISTALA, Padmalata; KUMARI, Priyanka. An approach to carry out consistency analysis on requirements: Validating and tracking

requirements through a configuration structure. In: **Requirements Engineering Conference (RE), 2013 21st IEEE International**. IEEE, 2013. p. 320-325.

[23] NIU, Nan; REDDIVARI, Sandeep; CHEN, Zhangji. Keeping requirements on track via visual analytics. In: **Requirements Engineering Conference (RE), 2013 21st IEEE International**. IEEE, 2013. p. 205-214.

[24] MAHMOUD, Anas; NIU, Nan. Supporting requirements traceability through refactoring. In: **Requirements Engineering Conference (RE), 2013 21st IEEE International**. IEEE, 2013. p. 32-41.

[25] GUO, Jin; CLELAND-HUANG, Jane; BERENBACH, Brian. Foundations for an expert system in domain-specific traceability. In: **Requirements Engineering Conference (RE), 2013 21st IEEE International**. IEEE, 2013. p. 42-51.

[26] SULTANOV, Hakim; HAYES, Jane Huffman. Application of reinforcement learning to requirements engineering: requirements tracing. In: **Requirements Engineering Conference (RE), 2013 21st IEEE International**. IEEE, 2013. p. 52-61.

[27] BOUILLON, Elke; MÄDER, Patrick; PHILIPPOW, Ilka. A survey on usage scenarios for requirements traceability in practice. In: **Requirements Engineering: Foundation for Software Quality**. Springer Berlin Heidelberg, 2013. p. 158-173

[28] CHINOSI, Michele; TROMBETTA, Alberto. BPMN: An introduction to the standard. **Computer Standards & Interfaces**, v. 34, n. 1, p. 124-134, 2012.

[29] SPANOUDAKIS, George; ZISMAN, Andrea. Software traceability: a roadmap. **Handbook of Software Engineering and Knowledge Engineering**, v. 3, p. 395-428, 2005.

[30] VON KNETHEN, Antje; PAECH, Barbara. A survey on tracing approaches in practice and research. **Frauenhofer Institut Experimentelles Software Engineering, IESE-Report No**, v. 95, 2002

[31] WINKLER, Stefan; PILGRIM, Jens. A survey of traceability in requirements engineering and model-driven development. **Software and Systems Modeling (SoSyM)**, v. 9, n. 4, p. 529-565, 2010.

[32] CLELAND-HUANG, Jane. Just enough requirements traceability. In: **Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International**. IEEE, 2006. p. 41-42.

[33] BLAAUBOER, Floris; SIKKEL, Klaas; AYDIN, Mehmet N. Deciding to adopt requirements traceability in practice. In: **Advanced Information Systems Engineering**. Springer Berlin Heidelberg, 2007. p. 294-308

[34] GOTEL, Orlena; FINKELSTEIN, Anthony. Extended requirements traceability: results of an industrial case study. In: **Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on**. IEEE, 1997. p. 169-178.

[35] HEINDL, Matthias; BIFFL, Stefan. A case study on value-based requirements tracing. In: **Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering**. ACM, 2005. p. 60-69**.**

[36] BOUQUET, Fabrice et al. Requirements traceability in automated test generation: application to smart card software validation. In: **ACM SIGSOFT Software Engineering Notes**. ACM, 2005. p. 1-7.

[37] J CLELAND-HUANG, Jane et al. Goal-centric traceability for managing non-functional requirements. In: **Proceedings of the 27th international conference on Software engineering**. ACM, 2005. p. 362-371.

[38] JARKE, Matthias. Requirements tracing. **Communications of the ACM**, v. 41, n. 12, p. 32-36, 1998.

[39] VERHANNEMAN, Tine et al. Requirements traceability to support evolution of access control. In: **ACM SIGSOFT Software Engineering Notes**. ACM, 2005. p. 1-7.

[40] DO PRADO LEITE, Julio Cesar Sampaio; BREITMAN, Karin Koogan. Experiences using scenarios to enhance traceability. In: **2nd International Workshop on Traceability in Emerging Forms of Software Engineering in Conjunction with the 18th IEEE International Conference on Automated Software Engineering, Montreal, Canada**. 2003. p. 63À70.

[41] RAVICHANDAR, Ramya; ARTHUR, James D.; PÉREZ-QUIÑONES, Manuel. Pre-requirement specification traceability: bridging the complexity gap through capabilities. **arXiv preprint cs/0703012**, 2007.

[42] BUBL, Felix; BALSER, Michael. Tracing cross-cutting requirements via context-based constraints. In: **Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on**. IEEE, 2005. p. 80-90.

[43] RAMESH, Balasubramaniam et al. Requirements traceability: Theory and practice. **Annals of software engineering**, v. 3, n. 1, p. 397-415, 1997.

[44] EGYED, Alexander; GRUNBACHER, Paul. Automating requirements traceability: Beyond the record & replay paradigm. In: **Automated Software Engineering, 2002. Proceedings. ASE 2002. 17th IEEE International Conference on**. IEEE, 2002. p. 163-171.

[45] CLELAND-HUANG, Jane et al. Best practices for automated traceability. **Computer**, n. 6, p. 27-35, 2007.

[46] DICK, Jeremy. Design traceability. **Software, IEEE**, v. 22, n. 6, p. 14-16, 2005.

[47] CLELAND-HUANG, Jane; CHANG, Carl K.; GE, Yujia. Supporting event based traceability through high-level recognition of change events. In: **Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International**. IEEE, 2002. p. 595-600.

[48] BASHIR, Muhamamd Farhan; QADIR, Muhammad Abdul. Traceability techniques: A critical study. In: **Multitopic Conference, 2006. INMIC'06. IEEE**. IEEE, 2006. p. 265-268.

[49] SPANOUDAKIS, George et al. Rule-based generation of requirements traceability relations. **Journal of Systems and Software**, v. 72, n. 2, p. 105-127, 2004.

[50] CLELAND-HUANG, Jane; CHANG, Carl K.; WISE, Jeffrey C. Automating performance-related impact analysis through event based traceability. **Requirements Engineering**, v. 8, n. 3, p. 171-182, 2003.

[51] KECECI, Nihal; GARBAJOSA, Juan; BOURQUE, Pierre. Modeling functional requirements to support traceability analysis. In: **Industrial Electronics, 2006 IEEE International Symposium on**. IEEE, 2006. p. 3305-3310.

[52] SOMMERVILLE, Ian; KOTONYA, Gerald. **Requirements engineering: processes and techniques**. John Wiley & Sons, Inc., 1998.

[53] CHAMBERLIN, Donald D. et al. A history and evaluation of System R. **Communications of the ACM**, v. 24, n. 10, p. 632-646, 1981.

[54] History of SQL. In: **Oracle Database SQL Reference** <http://docs.oracle.com/cd/B12037_01/server.101/b10759/intro001.htm>. Acesso em: 10 jan. 2014

[55] PL/SQL. In: **The Oracle Faq**. Disponível em: <http://www.orafaq.com/wiki/index.php?title=PL/SQL&oldid=13593>. Acesso em: 20 jan. 2014

[56] Oracle/PLSQL. In: **Tech On The Net**. Disponível em: <http://www.techonthenet.com/oracle/index.php>. Acesso em: 12 fev. 2014.

[57] The History of PL/SQL. In: **dbanotes.com**. Disponível em: <http://www.dbanotes.com/oracle-database/the-history-of-plsql/>. Acesso em: 12 fev. 2014.

[58] PROCEDURE. In: **Psoud.org**. Disponível em: <http://psoug.org/definition/PROCEDURE.htm>. Acessado: 15 mar.2014

[59] FUNCTION. In: **Psoud.org**. Disponível em: <http://psoug.org/definition/FUNCTION.htm>. Acessado: 15 mar.2014

[60] PACKAGE. In: **Psoud.org**. Disponível em: <http://psoug.org/definition/PACKAGE.htm>. Acessado: 15 mar.2014

[61] A Greek-English Lexicon. In: **Perseus.** Disponível em: <http://www.perseus.tufts.edu/hopper/text?doc=Perseus%3Atext%3A1999>.04.0057%3Aentry%3Dstrathgi%2Fa>. Acessado: 15 mar.2014

[62] MINTZBERG, Henry. Patterns in strategy formation. **Management science**, v. 24, n. 9, p. 934-948, 1978.

[63] MCKEOWN, Max. **The Strategy Book: How to Think and Act Strategically to Deliver Outstanding Results**. Pearson UK, 2012

[64] KVINT, Vladimir. **The global emerging market: strategic management and economics**. Routledge, 2010.

[65] SEMINOTTI, Nedio; CARDOSO, Cassandra. As configurações vinculares no pequeno grupo potencializando e/ou limitando seu processo. **Vínculo**, v. 4, n. 4, p. 26-37, 2007.

[66] RATIONAL Unified Process: Visão Geral, Conceitos chave, Conceitos Básicos do Rational Unified Process, Processo de Engenharia de Software In: **Wthreex**. Disponível em: <http://www.wthreex.com/rup/portugues/index.htm>. Acesso em: 6 fev.2014

[67] 5W e 1H (what/why/who/when/where/how). In: **Lannes Consulting**. Disponível em: <http://lslannes.com.br/adm/contcli/457/5w1h.pdf>. Acesso em: 6 fev. 2014

[68] 5W1H: Write Like a Journalist**.** In: **Davis Baldwin Consulting**. Disponível em: <http://www.davebaldwinconsulting.com/5W1H.html>. Acesso em: 6 fev. 2014.

[69] THE Kipling method. In: **Creating Minds**. Disponível em: <http://creatingminds.org/tools/kipling.htm>. Acesso em: 2 fev. 2014

[70] 5W1H – Ferramenta da qualidade. In: **Marketing Futuro**. Disponível em: http://marketingfuturo.com/5w1h-ferramenta-da-qualidade/. Acesso em: 10 jan. 2014.

[71] RATIONAL Unified Process: Disciplinas, Requisitos, Conceitos, Gerenciamento de Requisitos In: **Wthreex**. Disponível em: <http://www.wthreex.com/rup/portugues/index.htm>. Acesso em: 6 fev.2014.

Apêndice I
Algoritmos mais Relevantes

I. 1 Recupera todos os pacotes do BD.

```
/**
  * @return lista de todos os pacotes que estão no banco.
  */
 public static List<DBPackage> parseAllPackages(){
```

Figura 25 – Recupera os pacotes do BD

I. 2 Recupera todos os nomes dos pacotes.

```
/*
 * @return nomes de todos os pacotes do banco.
*/
 public static List<String> allPackageNames() {
        List<String> packages = new ArrayList<String>();
        String sql = "select alls.name from all_source alls
where alls.type = 'PACKAGE' and alls.owner = '"+DBUser+"' group by
alls.name order by alls.name";
        //System.out.println("Query " + sql);
        try {
        //    connection = DBUtils.getConnectionDefault();
              PreparedStatement pstmt =
connection.prepareStatement(sql);
              ResultSet rs = pstmt.executeQuery(sql);
```

Figura 26 – Recupera todos os nomes dos pacotes

I. 3 Realiza o parse nos pacotes.

O conteúdo de um pacote é incluído em um arquivo temporário de forma que seja lido o conteúdo linha a linha (parsePackage(String pkg_name)) Figura 24.

A seguir é realizado um parse no arquivo temporário, procurando pelo símbolo de abertura de comentário e pelo fechamento de comentário.

Quando um bloco de comentário é reconhecido como o último comentário de uma procedure ou function, esse conteúdo é atribuído a um buffer. Para encontrar o último comentário, é realizada uma busca pela string “Procedure” ou “Function” junto com um espaço em branco seguido pelo <nome da procedure> ou o <nome da function>, e caso não seja encontrado esse padrão, o conteúdo é descartado. Ao encontrar o último comentário de uma procedure ou function, é realizado outro parse no buffer para encontrar uma parte do buffer que corresponda ao padrão de instrumentação. Ao encontrar o padrão de instrumentação, é realizado outro parse para verificar quais itens do padrão são contemplados.

I.3.1. Chamadas aos parses nos pacotes

O sistema realiza a chamada à recuperação do conteúdo do pacote (**getPackageContents** (String packageName)) Figura 25. A seguir faz a inclusão do conteúdo no arquivo temporário, e em seguida recupera o nome de todas as procedures no pacote (getProcedureNamesAt(String packageName)) e realizada a chamada ao parse da procedure (parseText(File fileCode, String procName)). Ao final retorna um objeto DBPackage criado com o nome do pacote e a lista de procedures criadas. Durante o processo, é realizado todo o parse no pacote e suas procedures.

-**parseText** (File fileCode, String procName) – Reconhece um bloco de cabeçalho de procedure / function e chama o método parseComment(StringBuilder commentContext, String name) (Figura 26).

-**parseComment** (StringBuilder commentContext, String name) – Reconhece o início e o final do padrão de instrumentação e chama o método parseProcedure(String name, String header, String code)(Figura 27).

-**parseProcedure** (String name, String header, String code) – Faz a verificação do padrão de instrumentação (bloco correspondente ao padrão). Cria

três listas correspondentes ao conjunto de representação (UC, RN e Contexto) e ao final cria um objeto procedure (Figura 28).

```
    /**
     * Dado o nome do pacote, cria o objeto
    */
     public static DBPackage parsePackage(String pkg_name) {
            String fulltext =
DataBaseObject.getPackageContents(pkg_name).toString();
            File file = new File("tmp.sql");
            try {
                  FileWriter fw = new
FileWriter(file.getAbsoluteFile());
                  BufferedWriter bw = new BufferedWriter(fw);
                  bw.write(fulltext);
                  bw.close();
            } catch (IOException e) {
                  e.printStackTrace();
            }
```

Figura 27 – Arquivo temporário e chamadas aos parses

```
    public static StringBuilder getPackageContents(String
packageName) {
            StringBuilder result = new StringBuilder();
            String sql = "select * from all_source alls where
alls.owner='"+DBUser+"' and alls.type='PACKAGE BODY' and
alls.name='"
                                    + packageName + "' order by
alls.line";
            //System.out.println("Query " + sql);

            try {
            //    connection = DBUtils.getConnectionDefault();
                  PreparedStatement pstmt =
connection.prepareStatement(sql);
```

Figura 28 – Recupera conteúdo de um pacote

```
    private static Procedure parseText(File fileCode, String
procName) {
            String sCurrentLine;
            StringBuilder commentContext = new StringBuilder();
            int commentFlag = 0;
            Procedure procedure = null;
            try {
                  BufferedReader br = new BufferedReader(new
FileReader(fileCode.getAbsolutePath()));
                  while ((sCurrentLine = br.readLine()) != null) {
                        if (sCurrentLine.trim().contains("*/")) {
// encontrou comentario fechado
                              commentFlag = 2;
                        }
                        else if (commentFlag == 1)
                              commentContext.append(sCurrentLine);
                        else if
(sCurrentLine.trim().contains("/*")) { // encontrou comentario
aberto
                              commentFlag = 1;
                        }
                        else if (commentFlag == 2 && !
sCurrentLine.trim().equals("")) // analiza a próxima linha para ver
se tem procedure nome_procedure
                        {
                              if (sCurrentLine.contains("procedure
" + procName) || sCurrentLine.contains("PROCEDURE " + procName) ||
                                    sCurrentLine.contains("function
" + procName) || sCurrentLine.contains("FUNCTION " + procName))
                              {
                                    procedure =
```

Figura 29 – Reconhece um bloco de cabeçalho

```
    private static Procedure parseComment(StringBuilder
commentContext, String name) {
            String text = commentContext.toString();
            int init = text.indexOf("* <!-- begin-Trace -->");
            if (init != -1)
                  init += "* <!-- begin-Trace -->".length();
```

Figura 30 – Reconhece o início e o final do padrão de instrumentação

```
    public static Procedure parseProcedure(String name, String
header, String code){

            // Domain
            String regexDom = "%Domain\\s*:([^%]*)";
            Pattern patternDom = Pattern.compile(regexDom);
            // Faz o match do texto com a regex armazenando todos
os trechos onde o padrão é identificado
            // (i.e. Armazena todos os contextos)
            Matcher matcherDom = patternDom.matcher(header);

            // Armazena os contextos
            List <String> domains = new ArrayList<String>();
            if(matcherDom.find()){
                  String[] dms = matcherDom.group(1).split(",");
                  for(int j=0; j < dms.length; j++){
                        String trimmed = dms[j].trim();
                        if(! trimmed.isEmpty())
                              domains.add(trimmed);
                  }
            }

            // Business Rules
            String regexRN = "%RN\\s*:([^%]*)";
            Pattern patternRN = Pattern.compile(regexRN);
            // Faz o match do texto com a regex armazenando todos
os trechos onde o padrão é identificado
            // (i.e. Armazena todas as RNs)
            Matcher matcherRN = patternRN.matcher(header);

            // Armazena regras de negócio
            List <String> business_rules = new ArrayList<String>();
            if(matcherRN.find()){
                  String[] RNs = matcherRN.group(1).split(",");
                  for(int j=0; j < RNs.length; j++){
                        String trimmed = RNs[j].trim();
                        if(! trimmed.isEmpty())
                              business_rules.add(trimmed);
                  }
```

```
    // Cria um objeto Procedure e o adiciona a uma lista
            Procedure procedure = new Procedure(name, header,
code, inArgs, outArgs, business_rules, use_cases, developer,
creation_date, domains, null);
            //procedure.print();
```

Figura 31 – Verifica o padrão e cria o objeto procedure

Apêndice II
Diagrama de Classes da Ferramenta

II.1.
Telas do sistema

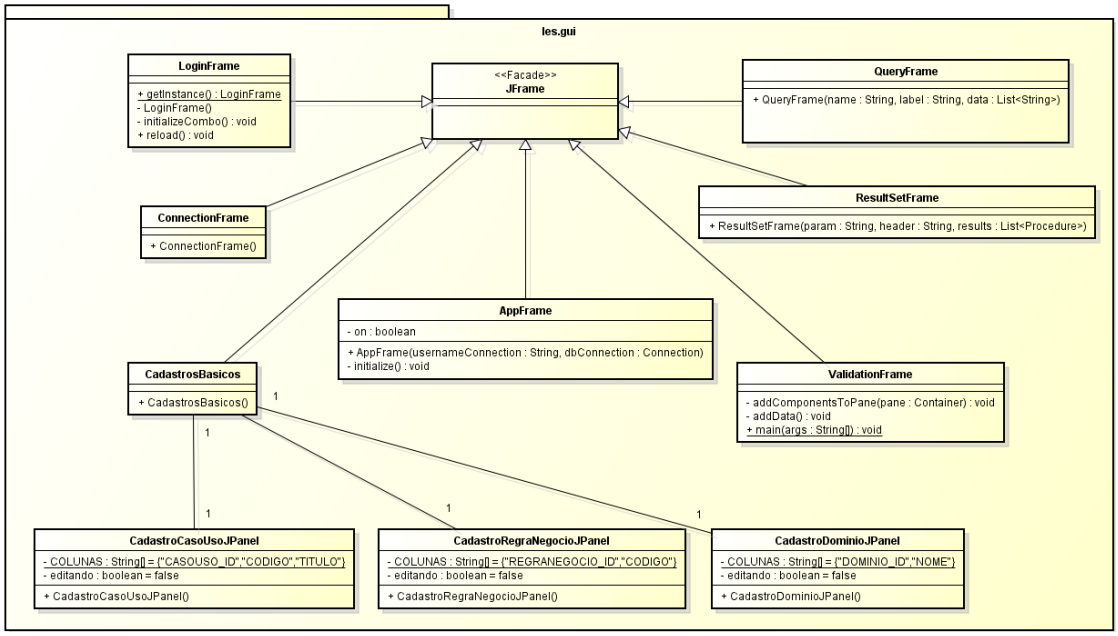

Figura 32 – Diagrama de classes das telas do sistema

II.2.
Classes de conexão

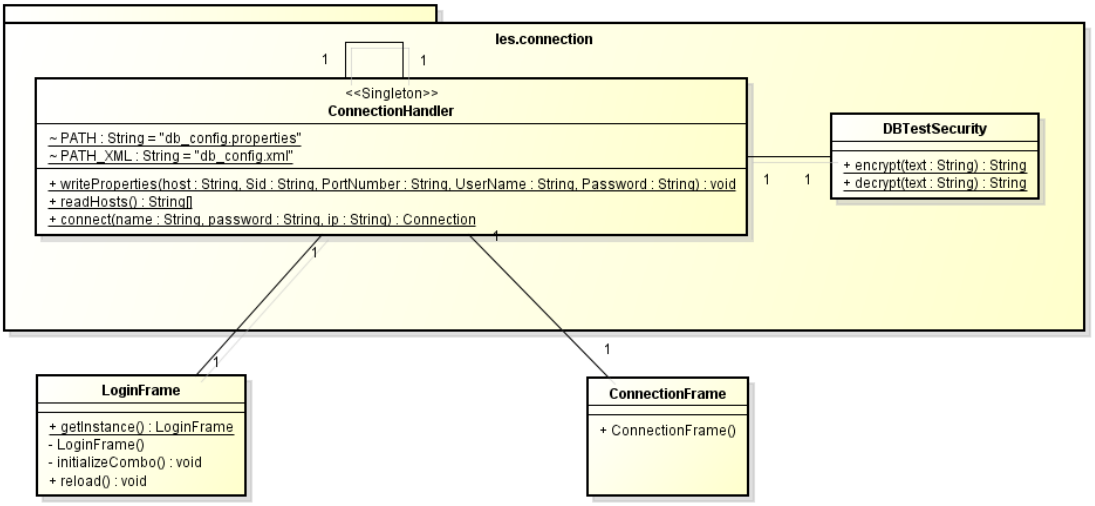

Figura 33 – Diagrama de classes de conexão

II.3.
Classes de administração dos itens de requisitos

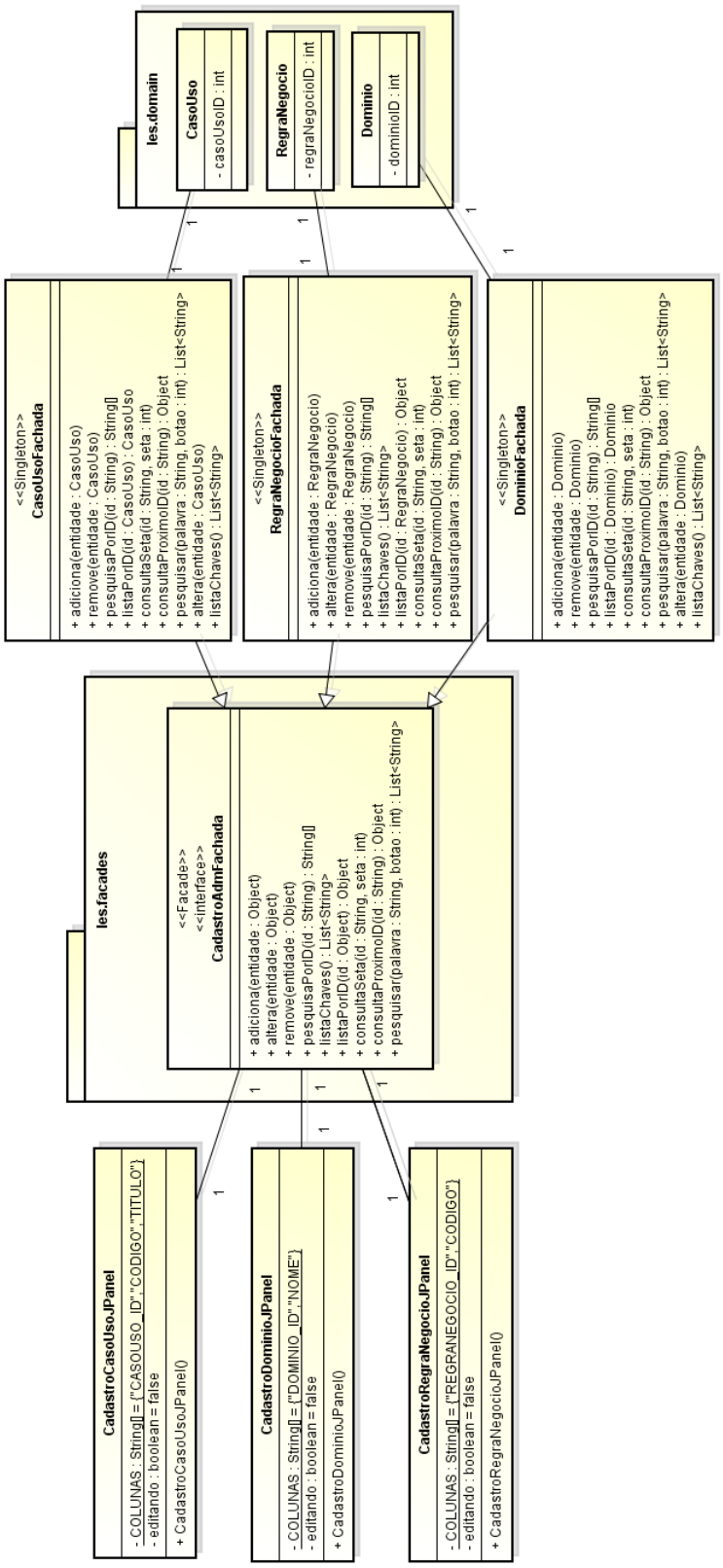

Figura 34 – Diagrama de classes de administração das representações

II.4.
Diagrama de classes do “parser”

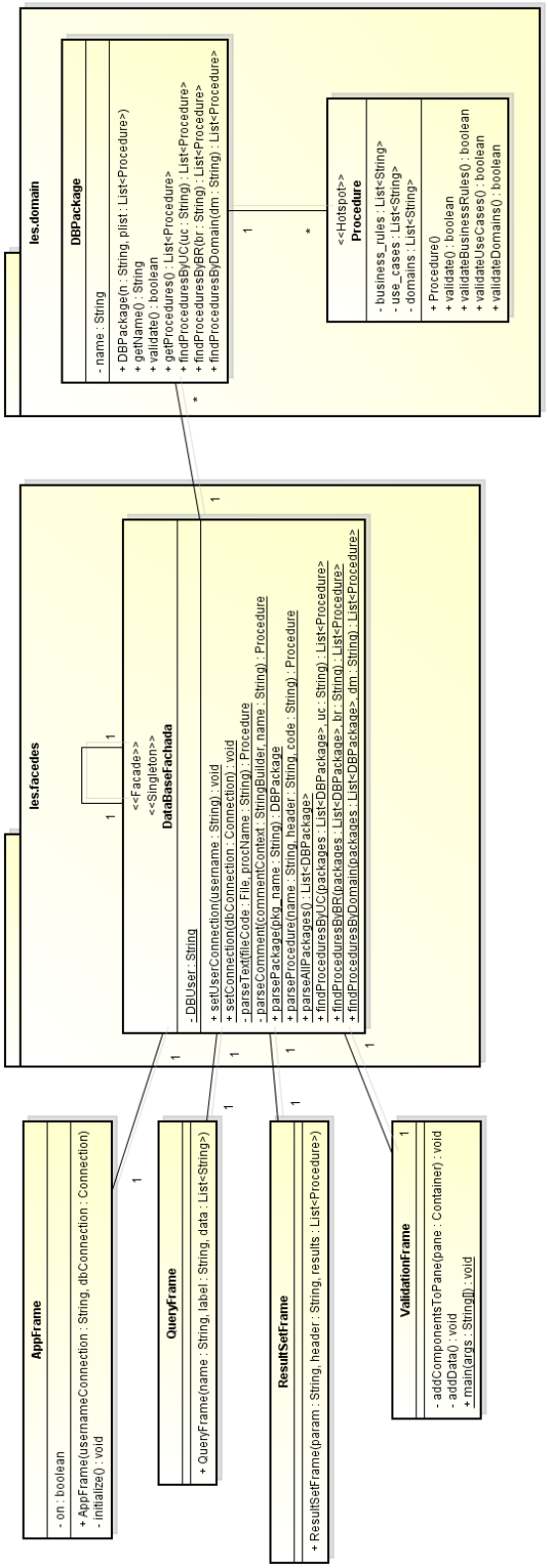

Figura 35 – Diagrama de classes do “parser”