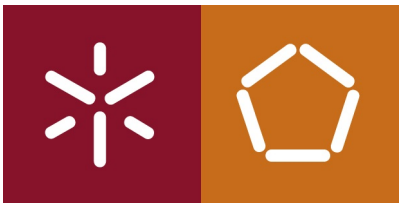
Universidade do Minho

Mestrado Integrado em Engenharia Informática



Gramáticas na Compreensão de Software

SelRA

Guilherme Pereira Martins (A70782)
Maria Inês Vieira Pinto (PG39292)
Ricardo Guerra Leal (A75411)

Dezembro 2020

Conteúdo

Capítulo 1

Introdução

O objetivo principal deste projeto passa pelo desenvolvimento de uma DSL que permite apoiar a equipa docente na escolha de um recurso de aprendizagem adequado a um dado aluno para a aprendizagem de um determinado conceito de programação. Este projeto visa maximizar a eficácia do processo de aprendizagem, atendendo às características emocionais/sociais de cada tipo de aluno que interferem na aprendizagem e também a idade ideal associada a cada tipo de recurso de aprendizagem. Com base em cada tipo de aluno e no conceito de programação que se pretende ensinar, o sistema deve propôr os 3 ou mais recursos de aprendizagem adequados.

Capítulo 2

Análise e Especificação

2.1 Descrição do problema

Durante o processo de ensino, um docente terá por vezes dificuldades em captar a atenção da totalidade da turma durante o periodo completo da aula. Devido à natureza humana, cada aluno possui diferentes caracteristicas e níveis de atenção ao que está a ser lecionado.

Com o desenvolvimento de uma *DSL* designada de *SeiRA*, o objectivo passa por proporcionar apoio tecnológico ao docente de maneira a adequar os seus métodos de ensino para qualquer tipo de aluno com esperanças de ter como retorno o mesmo rendimento na aula, de uma forma geral.

2.2 Especificação dos requisitos

Para se efetuar uma adequada análise de requisitos é necessário entender o propósito da aplicação e quais os campos relevantes para a mesma. Procedeu-se a identificar qual a informação necessária para o suporte do sistema e também a maneira como essa mesma informação ficaria organizada e estruturada.

2.2.1 Dados

Como primeiro passo, foi feita uma pesquisa sobre o tema com o objectivo de identificar dados importantes para o desenvolvimento da aplicação. Visto ser um sistema de auxilio ao ensino, foi necessário perceber que tipo de caracteristicas são relevantes a um aluno no contexto de uma UC **(Timidez, Desconcentração, Stress, Competitividade, Confiança ...)**, que conceitos serão de interesse para ensino **(C, C++, PHP, C, PYTHON, JAVA ...)** e que recursos poderão auxiliar o ensino destes conceitos **(Trabalhos em Grupo, Tutoriais, tutores Online, TPC, debates ...)**

Para o nosso sistema, temos como dados de entrada após o inquérito aos alunos, toda a informação relevante aos **Alunos**, **Conceitos de ensino** e aos **Recursos** disponiveis. Escolhendo as várias características associadas a um certo aluno e o conceito de programação que se pretende lecionar, o objetivo é o sistema conseguir propôr os recursos de aprendizagem e dessa forma, criou-se as 4 classes necessárias para a estrutura deste projeto:

- **Aluno:** Nome, Número, Idade, Caracteristicas;
- **Conceito:** Id, Nome, Curso, SubConceitos, Recursos Associados;
- **Recurso:** Id, Nome, Descrição, Idade Ideal, Caracteristicas;
- **Caracteristica:** Nome, Escala;

Na classe Alunos, criou-se os campos nome, número que representa a identificação do aluno, a idade e as características emocionais/sociais que interferem na aprendizagem. A classe conceito vai conter o id do conceito, o nome, o curso a que corresponde e os subconceitos/módulos que constituem esse conceito. A classe recurso contém o id, o nome do recurso, uma descrição, um intervalo de idade ideal que cada aluno deve corresponder, os conceitos associados a cada recurso e as características já predefinidas para um determinado conceito. Finalmente a classe característica que possui os campos nome e escala, sendo esta um número associado a cada característica funcionando como um nível.

2.2.2 Pedidos

Como pedido principal do nosso sistema, temos a produção *Questao*:
questao : command
numeroAluno nomeConceito;
Dado um número de aluno e o nome de um conceito, é feita uma análise ao perfil do aluno. Tendo em conta as suas caracteristicas emocionais/sociais e a escala associada a cada uma, será gerado a classificação para os recursos do conceito a ensinar. É também possível invocar o comando **listar** com possivel argumento **a** (listar alunos), **r** (listar recursos), **c** (listar conceitos).

2.2.3 Relações

Visto tratar-se de um sistema que contempla várias classes, as relações são imprescindíveis para as relacionar. Um recurso de aprendizagem vai ser proposto, dependendo de alguns fatores, nomeadamente as características dos alunos e de um tipo de conceito de aprendizagem que vai ser lecionado. Desta forma, irá ter que existir uma relação que ligue os conceitos de programação (CPrg) e os alunos (Al) aos recursos de aprendizagem (RA) - **lista.**

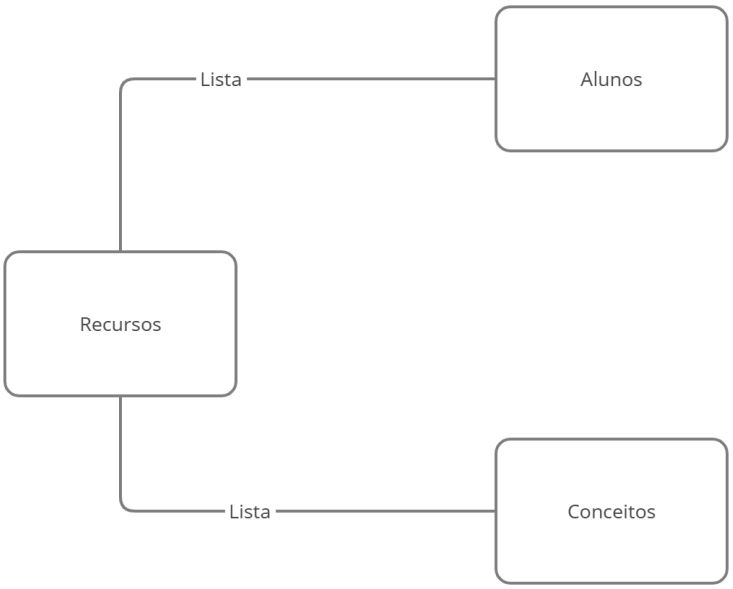


Figura 2.1: Relações entre as classes

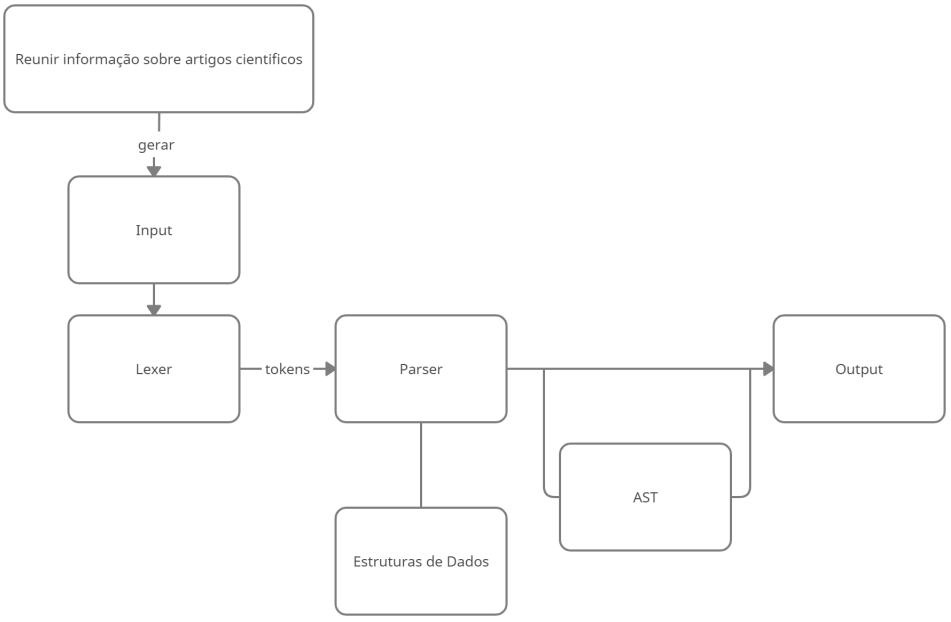
Desta forma, decidiu-se que se iria definir para cada conceito os recursos de aprendizagem associados a cada um deles , obtendo assim a seguinte ligação: **recurso - lista - conceito**. Por outro lado, existia a relação entre o aluno e o recurso: **recurso - lista - aluno** sendo que, cada recurso irá ser correspondido a um determinado aluno, a partir das suas características emocionais/sociais e dependendo também do nível de cada característica. Resumidamente, um recurso é definido por características emocionais/sociais já predefinidas e atribuídas a cada conceito de programação e cada característica do recurso de aprendizagem está definida com um intervalo de uma escala para ao fazer a comparação com a escala existente nas características definidas pelo aluno ser possível determinar o recurso mais adequado para esse tipo de aluno. Um recurso vai funcionar como uma classe principal, sendo que este vai estar relacionado tanto ao conceito, como ao aluno de forma ao sistema conseguir propôr recursos a um determinado aluno que estuda um determinado conceito.

Capítulo 3

Concepção da Resolução

3.1 Arquitectura de Sistema

Durante o desenvolvimento do nosso sistema, fomos seguindo uma estrutura de contrução do projecto. Iniciamos com a angariação de dados relevantes ao problema através da análise de alguns artigos cientificos. Apartir desses dados, fomos capazes de gerar um ficheiro de input com a informação a ser captada pela gramática posteriormente desenvolvida. Concluindo o lexer e o parser construimos a estrutura de dados que suporta o sistema.



3.2 Estruturas de Dados

Para o armazenamento e estruturação dos dados foram criados várias estruturas de dados em JAVA para cada entidade. Cada uma das entidades principais (Aluno,Conceito,Recurso) foi elaborado um objecto com os dados necessários para posteriormente se inserir na estrutura de dados adequada.

- **Alunos:** Para o armazenamento de um objecto **aluno** que contém a informação (nomeAluno,numero,idadeAluno e lista de caracteristicas) foi criada um HashMap <String, Aluno>com o numero de Aluno como key.
- **Recursos:** Da mesma forma, armazenamos o objecto **recurso** com a informação (id,nome,descricao.idadeIdeal e lista de carateristicas) num *HashMap<String, Recurso >* tendo como key o id do respectivo recurso.
- **Conceito:** Para os conceitos, guardamos o objecto **conceito** e a sua respectiva informação (id,ensina,nome,curso,subconceitos) num *HashMap<String, Conceito >* sendo o id a respectiva key.
- **Caracteristica:** Ambas as entidades **Aluno** e **Recurso** contêm uma lista de caracteristicas pertinentes para a relação entre os dois. São guardadas num *ArrayList<Caracteristica >*.

3.3 Algoritmos

Os algoritmos relevantes a mencionar são:

- **Procura de recursos que treinam um dado conceito, para um dado aluno**. Este algoritmo é invocado na produção das **questao: numeroAluno idConceito**, que tem como atributos herdados as estruturas de dados dos alunos, recursos e conceito. Para cada questão:
  -Aceder ao objeto Aluno presente no **HashMap¡String,Aluno¿** atraves da key (numeroAluno)
  -Aceder ao objeto Conceito presente no **HashMapString,Conceito¿** atraves da key (idConceito)
  -Para cada identificador(foEach) Recurso presente na listaIds do Conceito
  -Aceder ao objeto Recurso
  -Invocar o segundo algoritmo para a lista de caracteristicas e idade do Objeto Recurso e a lista de caracteristicas e idade do Objeto Aluno
  -Retornar ao utilizador o Recurso e sua Classificação

- **Comparação e classificação das características entre aluno e recurso**. Este algoritmo é invocado anteriormante e tem como função **comparar as caracteristicas e idades entre Recurso e Aluno**, e **gerar uma escala de adequação entre 0.0-10.0**. Para cada invocação:
  -Calcular a diferença de idades, se esse valor for maior do que um certo valor(6 no nosso caso), as idades são muito divergentes e o considera-se o recurso de desadequado (return 0)
  -Iterar entre as duas listas e para cada caracteristica em comum(match), calcular a adequacao entre a escala da caracteristica e adicionar ao total
  -Retornar o total

  Para calcular a adequação utilizou-se a seguinte fórmula:

  difEscala = Math.abs(Recurso.escala-Conceito.escala)

  dtotal+=(10-difEscala)/max

  Sendo **max** o tamanho do maior array entre caracteristicas de aluno e caracteristicas do recurso.

Todo o código do programa onde foram realizados estes algoritmos encontram-se nos Apêndices deste relatório (Apêndice B).

3.4 Ontologia

Decidiu-se complementar este projeto, construíndo uma ontologia, lecionada nas aulas, para ser visível as classes criadas, os campos que constitui cada uma e as relações existentes entre elas, facilitando a perspetiva do projeto. Utilizou-se o programa Protegê para a criação desta ontologia, começando pela classes.

Foram assim criadas 4 classes: Aluno, Característica, Conceito e Recurso. A classe conceito tem como subclasse o subConceito que representa os vários módulos associados a cada conceito e na classe característica foram atribuídas como subclasses o nome de todas as características existentes.

Figura 3.1: Classes criadas no Protegê

Seguidamente, foram criadas as ligações existentes entre cada uma das classes, denominadas Object Properties. As ligações criadas para as ontologias não as mesmas que foram utilizadas no código do projeto, sendo que nesse utilizou-se apenas duas relações, não sendo necessárias mais. Para a ontologia, optou-se por criar 4 tipos de ligações de forma a representar e a interligar todas as classes. A ligação "aprende"que liga um aluno a um conceito, a "ensina"que liga o recurso ao aluno, a "pertence"que representa o recurso associado ao conceito, a ligação "possui"que liga as características ao aluno e ao recurso e por fim, a ligação "temSubConceito"que atribui o respetivo Sub Conceito à classe Conceito.

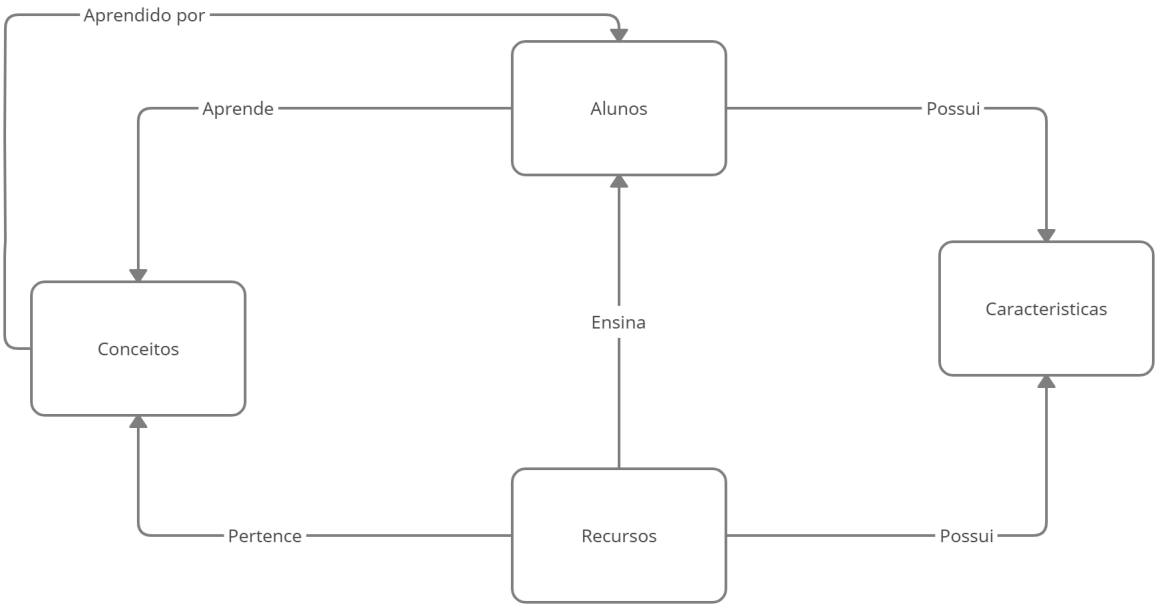


Figura 3.2: Esquema das ligações criadas no Protegê

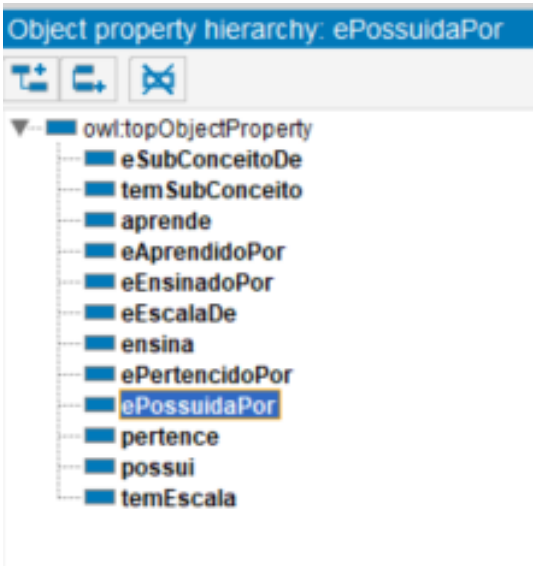


Figura 3.3: Ligações criadas no Protegê

Finalmente, foram criados os campos que cada classe contempla, como descrito no capítulo 2, onde cada campo tem um valor associado.

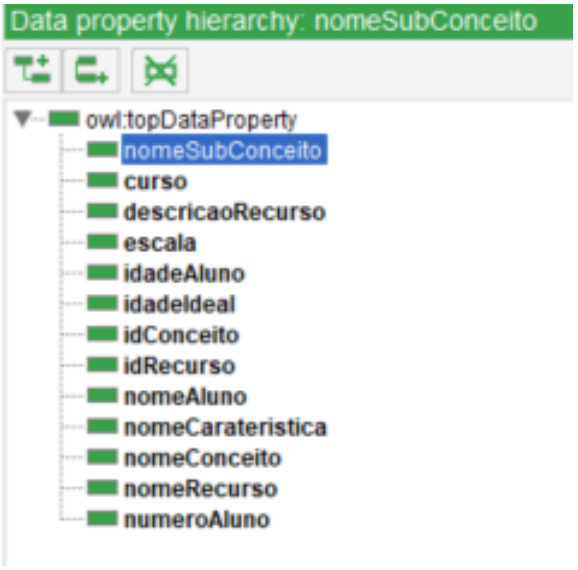


Figura 3.4: Campos criados no Protegê

De modo a representar um certo aluno, foram criados os indíviduos no Protegê conforme o ficheiro input que se tinha criado para testar o programa. Criado o indivíduo de um aluno, é necessário criar os restantes indivíduos para cada classe (conceitos, recursos, características), atribuir as ligações existentes entre cada um e adicionar também as propriedades de cada classe. Sendo assim, os resultados obtidos foram este (apenas um exemplo):

```
###  http://www.semanticweb.org/maria/ontologies/2020/11/SeIRa#miguel
:miguel rdf:type owl:NamedIndividual ,
                 :aluno ;
        :possui :Miguel_Inseguranca ,
                :Miguel_Preocupacao ;
        :eEnsinadoPor : rA1 ;
        :aprende :cPrg1 ;
        :idadeAluno 18 ;
        :nomeAluno "Miguel" ;
        :numeroAluno "A56432" .
```

Figura 3.5: Indivíduo de um Aluno

```
###  http://www.semanticweb.org/maria/ontologies/2020/11/SeIRa#Miguel_Preocupacao
:Miguel_Preocupacao rdf:type owl:NamedIndividual ,
                             :preocupacao ;
                    :escala 6 ;
                    :nomeCarateristica "Preocupacao" .
```

Figura 3.6: Indivíduo de uma Característica de um Aluno

```
###  http://www.semanticweb.org/maria/ontologies/2020/11/SeIRa#rA1
:rA1 rdf:type owl:NamedIndividual ,
             :recurso ;
     :ensina :miguel ;
     :pertence :cPrg1 ;
     :possui :rA1_participacao ,
             :rA1_preocupacao ;
     :descricaoRecurso "fghhddfh" ;
     :idRecurso "rA1" ;
     :idadeIdeal 17 ;
     :nomeRecurso "Trabalhos em Grupo" .
```

Figura 3.7: Indivíduo de um recurso de aprendizagem

```
###  http://www.semanticweb.org/maria/ontologies/2020/11/SeIRa#rA1_preocupacao
:rA1_preocupacao rdf:type owl:NamedIndividual ,
                          :preocupacao ;
                 :nomeCarateristica "Preocupacao" ;
                 :escala : 6 .
```

Figura 3.8: Indivíduo de uma Característica de um Recurso

```
###  http://www.semanticweb.org/maria/ontologies/2020/11/SeIRa#cPrg1
SeIRa:cPrg1 rdf:type owl:NamedIndividual ,
                     SeIRa:conceito ;
            SeIRa:temSubConceito SeIRa:tokens ;
            SeIRa:curso "MIEI" ;
            SeIRa:idConceito "cPrg1" ;
            SeIRa:nomeConceito "C" .
```

Figura 3.9: Indivíduo de um Conceito

```
###  http://www.semanticweb.org/maria/ontologies/2020/11/SeIRa#tokens
SeIRa:tokens rdf:type owl:NamedIndividual ,
                      SeIRa:subConceito ;
             SeIRa:nomeSubConceito "Tokens" .
```

Figura 3.10: Indivíduo de um SubConceito

Com este exemplo, é possível verificar que foi criado um aluno com o nome de Miguel que possui insegurança e preocupação como características emocionais/-sociais e também que é ensinado pelo recurso de aprendizagem número 1 (rA1) e aprende o conceito de programação número 1 (cPRG1) [Figura 3.5]. Na figura 3.6 é possível visualizar a característica denominada preocupação do aluno Miguel em que a escala que o mesmo atribuiu com o valor de 6. O recurso de aprendizagem que ensina o Miguel (rA1) [Figura 3.7] também pertence ao conceito de programação número 1 (cPrg1) e possui duas características: participação e preocupação (com o valor da escala a ser 6 que será recomendado para o tipo de característica deste aluno em específico) [Figura 3.8]. O cPrg1 que é um indíviduo de conceito tem os vários campos associados, como o seu nome que é a linguagem C [Figura 3.9] que tem associado como Sub Conceito "tokens"[Figura 3.10].

Com a realização desta ontologia, é possível ver as várias classes e ligações existentes entre as mesmas [Figura 3.11]. Neste caso em concreto, é possível perceber que a característica preocupação associada tanto ao aluno como ao recurso, continham o mesma escala e pertenciam ao mesmo conceito de programação pretendido. Por essa razão, o recurso de aprendizagem adequado para o aluno Miguel para o conceito de programação de C, seria o recurso número 1, denominado de trabalhos em grupo.

```
Ontologia SeIRa -> Exemplo de um aluno

conceitos {Aluno, Caracteristica, Conceito, Recurso}
subconceitos {subConceito}
individuos {miguel, Miguel_Preocupacao, rA1, rA1_preocupacao, cPrg1, tokens}
relacoes{aprende, ensina, pertence, possui}
triplos { Aluno = possui => Caracteristica
          Aluno = aprende => Conceito
          Recurso = pertence => Conceito
          Recurso = possui => Caracteristica
          Recurso = ensina => Aluno
          Conceito = temSubConceito => subConceito
          "miguel" = iof => Aluno
          "Miguel_Preocupacao" = iof => Caracteristica
          "rA1" = iof => Recurso
          "rA1_preocupacao" = iof => Caracteristica
          "cPrg1" = iof => Conceito
          "tokens" = iof => subConceito }
```

Figura 3.11: Esquetimazação da ontologia SeIRa

Capítulo 4

Codificação e Testes

4.1 Decisões e Problemas de Implementação

Para o desenvolvimento do nosso sistema, começou-se com uma pesquisa aprofundada sobre o tema que serviu como base para a estruturação da gramática inicial.

Durante o posterior desenvolvimento do código da gramática foi necessário tomar decisões importantes sobre como armazenar as entidades, como as relacionar e como obter o resultado final. Fomos nos deparando com problemas que nos obrigaram a alterar as estruturas de dados para acomodar novos dados e maneiras de relacionar o **Recurso** com o **Conceito** e finalmente o **Recurso** com o **Aluno.** Encontramos novamente dificuldades na decisão dessas mesmas relações quando se desenvolveu a **ontologia** para estabelecer uma boa hierarquia entre as entidades.

4.2 Testes Realizados e Resultados

Foram realizados vários testes com diferentes ficheiros de entrada. O ficheiro input contém diversos alunos com as características e escala correspondente. São descritos vários recursos com características associadas com a respetiva escala para cada uma delas e seguidamente os conceitos com os recursos que melhor estão associados aos mesmos. Por último a questão à qual pretendemos obter resposta onde se insere o número do aluno e o conceito de programação que ele pretende estudar fornecendo-lhe assim uma listagem dos recursos mais adequados para o mesmo. Mostra-se a seguir um exemplo de um ficheiro input [Figura 4.1]:

```
ALUNOS
paulo A79119 25 [(stress,2),(preocupacao,3),(motivacao,2), (participacao,6)];
raquel A76767 24 [(participacao,4),(motivacao,8),(timidez,2)];
lima A74442 23 [(preocupacao,8),(tarefacurta,3),(inseguranca,2)]

RECURSOS
rA1 TrabalhosEmGrupo Descricao1 17 [(participacao,7),(preocupacao,6),(motivacao,3)];
rA2 FeedbacksConstantes Descricao2 17 [(participaca,5),(introvertivo,8),(confianca,4)];
rA3 Tutoriais Descricao3 22 [(timidez,7),(stress,6),(concentracao,3)];
rA4 TutoresOnline url 19 [(confianca,4),(desconcentracao,4),(inseguranca,3)];
rA5 DiarioSemanal Descricao5 17 [(participaca,5),(motivacao,8), (confianca,7),(tarefacurta,8)];
rA6 Livro cLuisDamas 21 [(stress,3),(motivacao,8),(confianca,7),(participacao,0)];
rA7 Video Descricao7 20 [(stress,2),(concentracao,4),(confianca,7)];
rA8 Forum Descricao8 17 [(participacao,4),(preocupacao,1),(timidez,5),(inseguranca,2)];
rA9 PodCast Descricao9 23 [(concentracao,3),(preocupacao,7),(confianca,6)];
rA10 Debate Descricao10 19 [(confianca,8),(participacao,7),(competitividade,7)];
rA11 Pesquisa Descricao11 20 [(preocupacao,7),(participacao,3),(stress,6),(tarefacurta,3)];
rA12 Exercicios Descricao12 18 [(inseguranca,6),(introvertivo,5),(tarefacurta,7)];
rA13 TPC link 24 [(preocupacao,8),(tarefacurta,3),(inseguranca,2),(timidez,6)];
rA14 RecursoEstatico aqrueossitios 22 [(visual,7),(motivacao,8)];
rA15 RecursoEstatico2 lugares 21 [(motivacao,7),(concentracao,6),(introvertido,6)]

CONCEITOS
cPrg1 C Tokens MIEI [rA3,rA6,rA1,rA13];
cPrg2 C Variable MIEI [rA12,rA13,rA11]


QUESTAO
listar r,
A79119 cPrg1,
A74442 cPrg2,
A76767 cPrg1
```

Figura 4.1: Dados de Input

Como resultado a esta questão para listar os recursos adequados aos alunos Paulo, Raquel e Lima que pretendem aprender diferentes cursos de programação, foi retornado como output o seguinte ficheiro:

```
--------------------------------------------------------------------------------
Ensinar cPrg1: C,Tokens ao Aluno A79119: paulo
Classificação 1.5 ID: rA3 Nome: Tutoriais Descricao: Descricao3
Classificação 4.25 ID: rA6 Nome: Livro Descricao: cLuisDamas
A idade do Recurso não se adequa ao Aluno
Classificação 0.0 ID: rA1 Nome: TrabalhosEmGrupo Descricao: Descricao1
Classificação 1.25 ID: rA13 Nome: TPC Descricao: link
--------------------------------------------------------------------------------
Ensinar cPrg2: C,Variable ao Aluno A74442: lima
Classificação 4.0 ID: rA12 Nome: Exercicios Descricao: Descricao12
Classificação 7.5 ID: rA13 Nome: TPC Descricao: link
Classificação 4.75 ID: rA11 Nome: Pesquisa Descricao: Descricao11
--------------------------------------------------------------------------------
Ensinar cPrg1: C,Tokens ao Aluno A76767: raquel
Classificação 1.6666666666666667 ID: rA3 Nome: Tutoriais Descricao: Descricao3
Classificação 4.0 ID: rA6 Nome: Livro Descricao: cLuisDamas
A idade do Recurso não se adequa ao Aluno
Classificação 0.0 ID: rA1 Nome: TrabalhosEmGrupo Descricao: Descricao1
Classificação 1.5 ID: rA13 Nome: TPC Descricao: link
```

Figura 4.2: Output

Para o aluno Paulo com o número de identificação A79119 que pretende aprender o conceito de programação C definido com as características stress (escala 2), preocupação (escala 3), motivação (escala 2) e participação (escala 6), são listados todos os recursos referentes ao conceito de programação que o mesmo pretende aprender, neste caso a linguagem C e a respetiva classificação para desta forma se conseguir perceber qual o recurso que está melhor adequado para este aluno. O recurso rA1 denominado "Trabalhos em Grupo"retornou um valor de classificação de 0, sendo que a idade do recurso não se adequa ao aluno, tendo uma idade 6 anos superior à do recurso. Os recursos rA3 "Tutoriais"e rA13 "TPC"obtiveram o valor de 1.5 e 1.25 de classificação respetivamente. O recurso com um melhor valor de classificação foi o rA6 denominado de Livro com um valor de 4.25, concluíndo que este seria o recurso mais adequado para este aluno.

Capítulo 5

Conclusão

Com o desenvolvimento da nossa *DSL* designada de *SeiRA*, todo o grupo consolidou a importância de uma fase inicial de pesquisa e análise de artigos visto ser a base de toda a construção do sistema. Foram esse dados que facilitaram a contrução da estrutura inicial da gramática e das suas produções principais e que possibilitaram o desenvolvimento das relações entre entidades, das estruturas de dados e finalmente da atribuição do Recurso adequado ao aluno.

Com a finalização do nosso sistema, conseguimos perceber que apartir de um perfil não muito aprofundado do aluno, um docente tem a possibilidade de optimizar o seu ensino e possivelmente obter melhores resultados em situações de avaliação. No entanto, visto que o perfil não é muito completo, a atribuição do recurso pode ainda ser mais optimizada e personalizada para um determinado aluno.

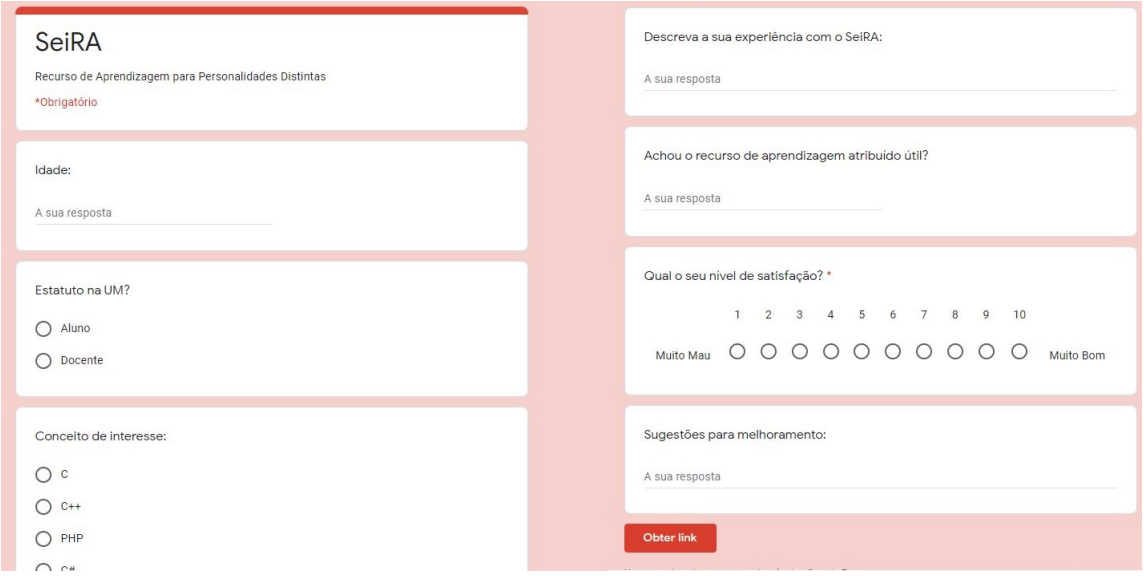
Como ideias de futuro para o projecto, poderiamos aprofundar a análise de cada aluno para perceber se seria possivel um melhoramento de ensino e uma atribuição de recurso mais eficiente. Seria tambem mais intuitivo para um aluno completar o seu perfil tendo uma interface aplativa e simples de utilizar.

Concluindo, o tema foi analisado e aprofundado com grande interessante visto pertencermos ao grupo de estudo a que a aplicação esta direcionada, vemos então da perspectiva de um aluno os melhoramentos possiveis no nosso próprio percurso académico. Ficamos com a ideia de que o nosso sistema é uma pequena demostração de um tema com enorme potencial.

Appendices

Apêndice A

Formulário de Avaliação



https://docs.google.com/forms/d/e/1FAIpQLSfaYVJJeZt2oJdX1CadkUkaialSkTUgtuGOtXXD2S96Z9re-w/viewform

Apêndice B

Código do Programa

```
cNe: alunos recursos conceitos questoes[$alunos.alunosHash, $recursos.recursosHash, $conceitos.conceitosHash]
    ;

questoes [HashMap<String, Aluno> alunosHash, HashMap<String, Recurso> recursosHash, HashMap<String, Conceito> conceitosHash]

        : 'QUESTAO'
          questao[$questoes.alunosHash, $questoes.recursosHash, $questoes.conceitosHash]

       (',' questao[$questoes.alunosHash, $questoes.recursosHash, $questoes.conceitosHash])*

        ;


questao [HashMap<String, Aluno> alunosHash, HashMap<String, Recurso> recursosHash, HashMap<String, Conceito> conceitosHash]

    : command [$questao.alunosHash, $questao.recursosHash, $questao.conceitosHash]

    | numeroAluno idConceito
      {
        //Consultar caracteristicas do aluno
        Aluno a = $questao.alunosHash.get($numeroAluno.text);
        if(a!=null){
            List<Caracteristica> caracteristicasA = a.caracteristicas;
            //Consultar o conceito
            if($questao.conceitosHash.containsKey($idConceito.text)){
                Conceito c = $questao.conceitosHash.get($idConceito.text);
                System.out.print("-------------------------------------------------------------------------------\n");
                System.out.print("Ensinar "+$idConceito.text+": "+c.nomeConceito+","+c.subConceito+" ao Aluno "+$numeroAluno.text+": "+a.nomeAluno+"\n");
                c.idsRecurso.forEach(value -> {
                    //consultar recurso
                    Recurso r = $questao.recursosHash.get(value);
                    if(r!=null){
                        Questao q = new Questao();
                        double total = q.adequa(a.idadeAluno,r.idadeIdeal, caracteristicasA, r.caracteristicas);
                        System.out.print("Classificação "+total+" ID: "+r.id+" Nome: "+r.nome+" Descricao: "+r.descricao+"\n");
                    }else{
                        System.out.print("O Recurso "+value+" não existe\n");
                    }
                });
            }else{
                System.out.print("O Conceito "+$idConceito.text+" não existe\n");
            }
        }else{
            System.out.print("O Aluno "+$numeroAluno.text+" não existe\n");
        }
       }

    ;
```

```
command [HashMap<String, Aluno> alunosHash, HashMap<String, Recurso> recursosHash, HashMap<String, Conceito> conceitosHash]
      : commandListar commandType
          {
            Questao q = new Questao();

            if($commandType.text.equals("a")){
                q.listarAlunos($command.alunosHash);
            }else if($commandType.text.equals("r")){
                q.listarRecursos($command.recursosHash);
            }else if($commandType.text.equals("c")){
                q.listarConceitos($command.conceitosHash);
            }else{
                System.out.println("Argumento do comando inválido\n");
            }
          }
      | commandInterativo
        {
            System.out.println("Comando Interativo\n");
        }
        ;

//alunos-------------------------------------------------------------------------------
alunos
       returns[HashMap<String, Aluno> alunosHash]
@init  {
        HashMap<String, Aluno> aux = new HashMap<String, Aluno>();
       }

    : ALUNOS
        al = listaAlunos[aux] {
                                $alunos.alunosHash = $al.auxOut;
                              }
    ;


 listaAlunos [HashMap<String, Aluno> auxIn]
             returns[HashMap<String, Aluno> auxOut]
 @after {
         $listaAlunos.auxOut=$listaAlunos.auxIn;
        }

         : a1 = aluno[$listaAlunos.auxIn]{
                 $listaAlunos.auxIn=$a1.alunosHashOut;
                }
      (';' a2 = aluno[auxIn]{
                 $listaAlunos.auxIn=$a2.alunosHashOut;
                })*

 ;
 aluno [HashMap<String,Aluno> alunosHashIn]
       returns[ String num, HashMap<String,Aluno> alunosHashOut]
 @after {
         $aluno.alunosHashOut = $aluno.alunosHashIn;
        }
     : nomeAluno numeroAluno idadeAluno caracteristicas
     {
      int idade = Integer.parseInt($idadeAluno.text);
      Aluno a = new Aluno($nomeAluno.text, $numeroAluno.text, idade, $caracteristicas.caracteristicasOut);
      //adicionar Aluno ao HashMap
      $aluno.alunosHashIn.put($numeroAluno.text,a);
     }
     ;
```

```
recursos
        returns [HashMap<String, Recurso> recursosHash]
@init  {
        HashMap<String, Recurso> aux = new HashMap<String, Recurso>();
       }
@after {
        $recursos.recursosHash = aux;
       }

      : RECURSOS

         r1 = rA[aux]{
                aux=$r1.recursoHashOut;
             }
    (';' r2 = rA[aux]{
                aux=$r2.recursoHashOut;
             })*
    ;

rA  [HashMap<String,Recurso> recursoHashIn]
      returns[HashMap<String,Recurso> recursoHashOut]
@after{
        $rA.recursoHashOut=$rA.recursoHashIn;
      }
    : idRecurso nomeRecurso descr idadeIdeal caracteristicas
    {
     int idade = Integer.parseInt($idadeIdeal.text);

     Recurso r = new Recurso($idRecurso.text, $nomeRecurso.text, $descr.text, idade, $caracteristicas.caracteristicasOut);
     //adicionar Recurso ao HashMap
     $rA.recursoHashIn.put($idRecurso.text,r);

    }
  ;
```

```
conceitos
         returns[HashMap<String, Conceito> conceitosHash]
@init  {
        HashMap<String, Conceito> aux = new HashMap<String, Conceito>();
       }
@after {
        $conceitos.conceitosHash = aux;
       }

      : CONCEITOS

        c1 = cProg[aux] {
                aux=$c1.conceitoHashOut;
            }
   (';' c2 = cProg[aux]{
                aux=$c2.conceitoHashOut;
            })*
    ;


cProg[HashMap<String, Conceito> conceitoHashIn]
      returns[HashMap<String, Conceito> conceitoHashOut]
@after{
        $cProg.conceitoHashOut=$cProg.conceitoHashIn;
      }

    : idConceito nomeConceito subConceito curso listaIdRecurso
    {
     Conceito c = new Conceito($idConceito.text, $nomeConceito.text, $subConceito.text, $curso.text, $listaIdRecurso.listaIdRecOut);

     if($cProg.conceitoHashIn.containsKey($idConceito.text)){
        System.out.println("Key existente - Não é possível inserir o conceito "+c.idConceito);
     }else{
        $cProg.conceitoHashIn.put($idConceito.text, c);
        //System.out.println("Nova Key - Conceito inserido "+c.idConceito);
     }

    }
    ;
```

```
listaIdRecurso
        returns [ArrayList<String> listaIdRecOut]
@init {
        ArrayList<String> aux = new ArrayList<String>();
      }
@after {
         $listaIdRecurso.listaIdRecOut = aux;
       }
       : LPAREN
            idRecurso{
                      aux.add($idRecurso.text);
                     }

       (',' idRecurso{
                      aux.add($idRecurso.text);
                     })*
       RPAREN

;

//caracteristicas-------------------------------------------------------------------------------

caracteristicas returns [ArrayList<Caracteristica> caracteristicasOut]
@init {
        ArrayList<Caracteristica> aux = new ArrayList<Caracteristica>();
       }
@after {
         $caracteristicas.caracteristicasOut = aux;
       }
    : LPAREN
          caracteristica {
                          int escalac1 = Integer.parseInt($caracteristica.escal);
                          Caracteristica c1 = new Caracteristica($caracteristica.car, escalac1);
                          aux.add(c1);
                         }
      (',' caracteristica {
                          int escalac2 = Integer.parseInt($caracteristica.escal);
                          Caracteristica c2 = new Caracteristica($caracteristica.car, escalac2);
                          aux.add(c2);
                         })*
      RPAREN
     ;


                    caracteristica returns[String car, String escal]
                        : '(' PALAVRA ',' escala ')' {}
                        ;
```

Bibliografia

[1] ANTLR (ANother Tool for Language Recognition)
https://www.antlr.org/

[2] Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte
https://impactum-journals.uc.pt/rppedagogia/article/view/1647-8614$_4 2 - 2_9$

[3] Estratégias para motivar os alunos
https://www.researchgate.net/publication/277069461$_E strategias_p ara$
$_m otivar_o s_a lunos/fulltext/55a30c8f08ae1c0e04653390/Estrategias - para -$
$motivar - os - alunos.pdf$

[4] ENSINAR E APRENDER ALGORITMOS E PROGRAMAÇÃO NO ENSINO SUPERIOR: Desafios e melhores práticas
https://www.researchgate.net/publication/310628856$_E NSINAR_E$
$_A PRENDER_A LGORITMOS_{EP} ROGRAMACAO_N O_E NSINO_S UPERIOR_D esafios$
$_{em} elhores_p raticas$