Selecção de Recursos de Aprendizagem

Etienne Costa
Universidade do Mihho
Escola de Engenharia
Braga

(24 de janeiro de 2021)

Resumo

O presente artigo tem como principal objectivo explanar o processo de desenvolvimento do software SELRA com recurso a gramática de atributos. Este software consiste na selecção e atribuição de um conjunto de recursos de aprendizagem tirando partido da similiradidade das características entre os intervenientes do sistema.

1 Introdução

O primeiro impacto no processo de aprendizagem tende a ser muita das vezes desmotivante face a má escolha de recursos a utilizar para aprender um determinado conceito. Com base nisso, é primordial numa fase inicial ocorrer um certa indicação dos recursos a usar de modo a acelerar a curva de aprendizagem. Para tal foi desenvolvido um software à custa da gramática de atributos, que compromete-se a realizar essa indicação com base na similiradidade das características entre os discentes e os recursos a serem utilizados.

2 Background

Existem diversos processos de desenvolvimento de software, no entanto há algumas actividades básicas comuns à grande parte dos processos existentes, tais como: **levantamento de requisitos, análise de requisitos, projecto, implementação e testes**.

2.1 Levantamento de requisitos

O levantamento de requisitos é considerada a etapa mais importante , pois é nela que se priorizam as necessidades dos futuros usuários do software, necessidades essas denominadas como requisitos. Os requisitos principais levantados nesta etapa foram os seguintes:

Requisitos de Descrição

- Discente: ID, Nome, Características;
- Linguagem: ID, Nome;
- Recurso: ID, Nome, Intervalo, Descrição, Características, Conteúdos.

Requisitos de Exploração

- Lista das linguagens de programação;
- Lista dos recursos de aprendizagem;
- Lista dos alunos e os seus recursos de aprendizagem.

2.2 Análise de Requisitos

Após uma análise detalhada do problema, agrupou-se os diferentes conceitos, de modo a poder extrair a informação relevante e definir as entidades, relacionamentos e atributos que irão constar no modelo.
A abordagem usada na modelação conceptual foi:

- Identificar entidades e relacionamentos
- Identificar os atributos das entidades e relacionamentos,e definir o domínio dos mesmos
- Determinar as chaves das entidades
- Construção do diagrama conceptual
- Validação com o utilizador final

De seguida é apresentado o diagrama conceptual sendo o mesmo ponto de partida para uma estratégia de solução.

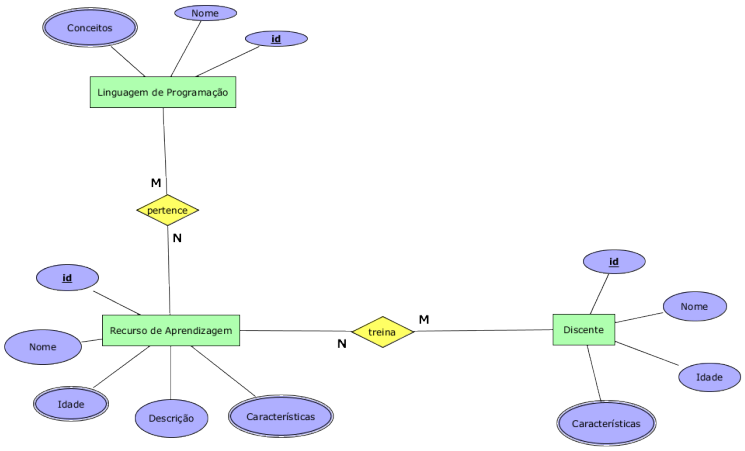

Figura 1: Modelo Conceptual

2.3 Projecto

Nesta fase , foi decidido o modo como o sistema funcionará internamente, de modo a que os requisitos possam ser atendidos.

A implementação do software foi feita à custa de uma gramática de atributos que a posterior seria passada para o ANTLR que lê uma gramática e gera um reconhecedor para a linguagem definida pela gramática,i.e, um programa que lê um fluxo de entrada e gera um erro se o fluxo de entrada não estiver em conformidade com a sintaxe especificada pela gramática. Para o contexo do nosso software é mandatório que o fluxo de entrada esteja no formato **XML**, sendo que de seguida o mesmo é processado com auxílio a linguagem de programação Java de modo a gerar a páginas em html.

2.3.1 Arquitectura do Software

A arquitetura de software de um sistema consiste na definição dos componentes de software, suas propriedades externas, e seus relacionamentos com outros softwares. O projecto de arquitetura está preocupado com a compreensão de como um sistema deve ser organizado e com a estrutura geral desse sistema. O resultado do processo de projecto de arquitetura é um modelo de arquitetura que descreve como o sistema está organizado em um conjunto de componentes de comunicação.

A seguinte arquitetura específica que o software recebe um ficheiro **XML** via um arquivo de computador ou via teclado,que por sua vez é passado para verificar sua estrutura gramatical segundo uma determinada gramática formal. Este proceso trabalha em conjunto com a análise léxica e análise semântica. A **análise sintática transforma** um texto na entrada em uma **estrutura de dados**, em geral uma árvore, o que é conveniente para processamento posterior e captura a hierarquia implícita desta entrada. No contexto do nosso software é feito o preenchimento de **estruturas de dados** à custa do **Parser+Transformador**, sendo feita numa fase final, um segundo processamento que permitirá gerar páginas **HTML** na qual o **end user** usufrui do resultado produzido.

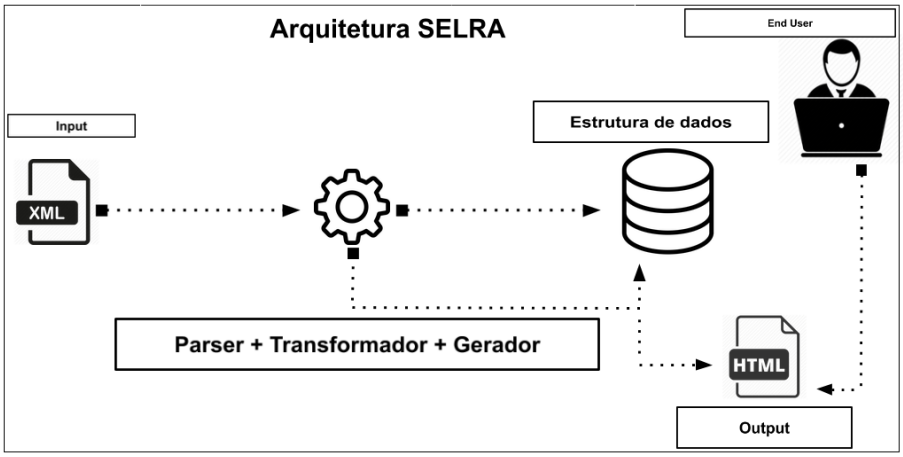

Figura 2: Arquitetura SELRA

2.4 Implementação

No contexto do nosso software o algoritmo de adequação é considerado o kernel, pois é com base nele que suprimos a principal necessidade do utilizador final. O mesmo tem a responsabilidade de calcular a custa da similaridade de características dos recursos de aprendizagem e utilizadores, os recursos adequados para cada um dos utilizadores finais, sendo que para isso o resultado produzido pertence a um intervalo de adequação dividido em 4 partes.

Intervalo de Adequação			
Mau	Medíocre	Bom	Muito Bom
]0-25[	[25-50[	[50-75[	[75-100]

Tabela 1: Tabela com o intervalo de adequação

De seguida é apresentado o pseudocódigo do algoritmo de adequação implementado :

Algorithm 1 Algoritmo de Adequação

```
1: for student ∈ Students do
2:     for resource ∈ Resources do
3:         if student.age ∈ resource.range then
4:             match = #(student.characteristics ∩ resource.characteristics)
5:             rate = (match/resource.characteristics.size) * 100
6:             if rate > 0.0 then
7:                 student.resources' = student.resources ∪ resource
8:
9:
```

A ideia principal do algoritmo consiste na interseção do conjunto de características de um determinado recurso de aprendizagem e de um determinado discente, sendo que a cardinalidade dessa intersecção é dividida pela cardinalidade do recurso em causa de modo a obter um valor dentro da escala previamente definida.

3 Teste

Diversas actividades de testes são executadas a fim de se validar o produto de software, testando cada funcionalidade de cada módulo, buscando, levando em consideração a especificação feita na fase de projeto. Onde o principal resultado é o relatório de testes, que contém as informações relevantes sobre erros encontrados no sistema, e seu comportamento em vários aspectos. No final dessa actividade, os diversos módulos do sistema são integrados, resultando no produto de software. A análise destes resultados foram feitas à custa de um log de dados com o registo de eventos relevantes e um breve inquérito com utilizadores finais.

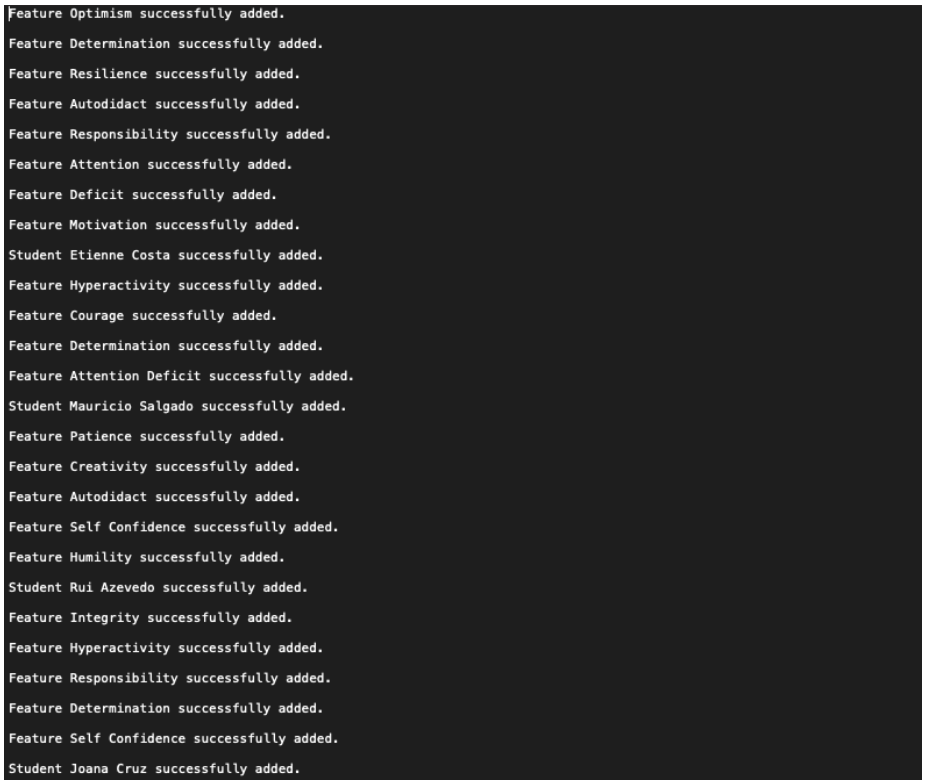

Figura 3: Relatório de erros e sucessos.

4 Resultados Obtidos

4.1 Página Inicial

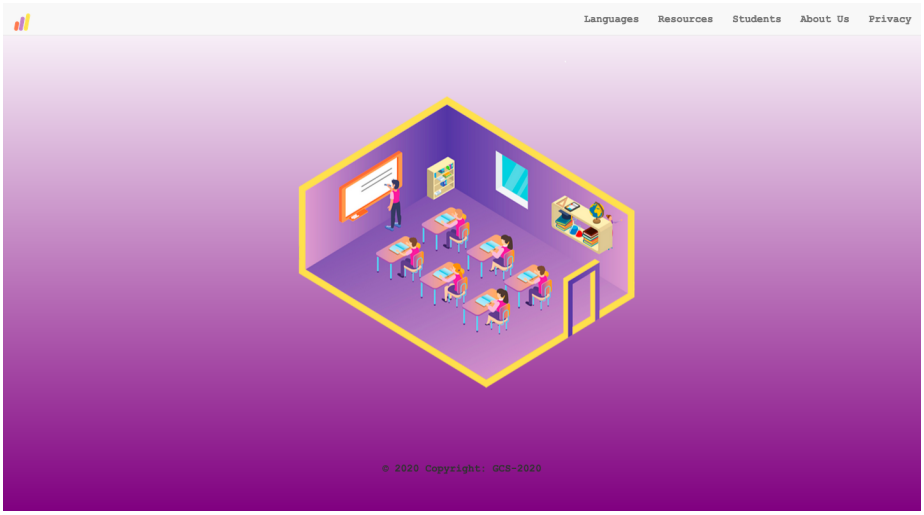

Figura 4: Página Inicial do Software.

4.2 Lista de Linguagens

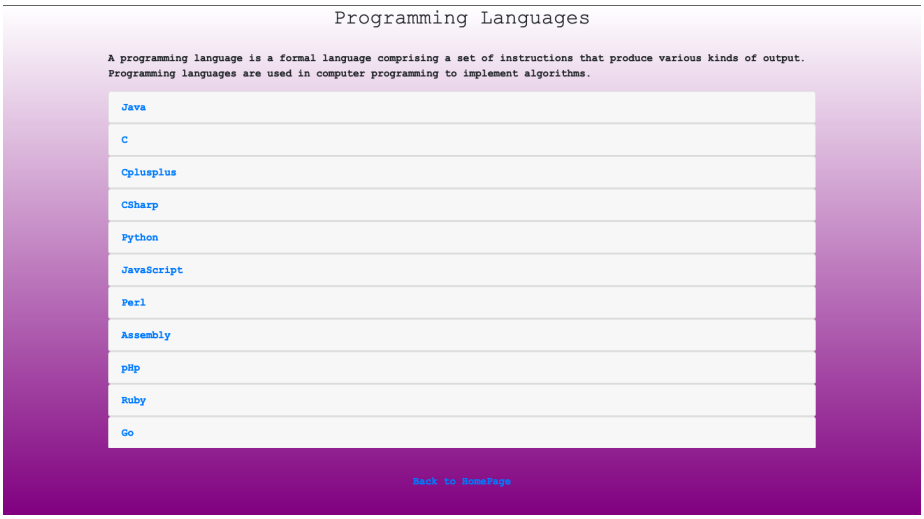

Figura 5: Lista de Linguagens do Sistema.

4.3 Lista de Recursos

Figura 6: Lista de Recursos do Sistema.

4.4 Recurso de Aprendizagem

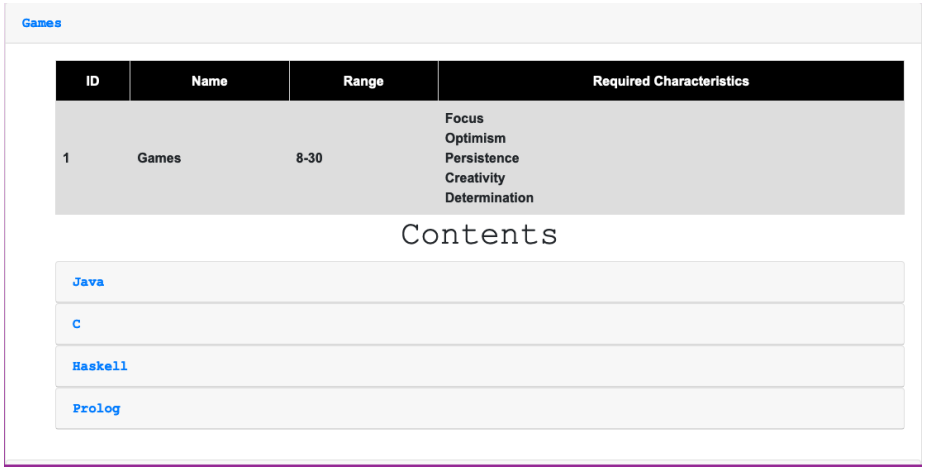

Figura 7: Exemplo de Games como Recurso de Aprendizagem.

4.5 Conceitos

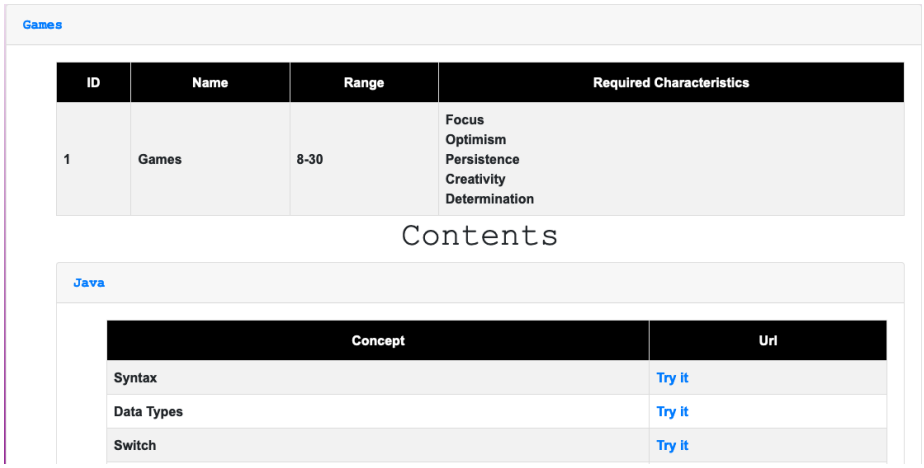

Figura 8: Conceitos de uma Linguagem de Programação do Recursos de Aprendizagem Games.

4.6 Lista de Alunos

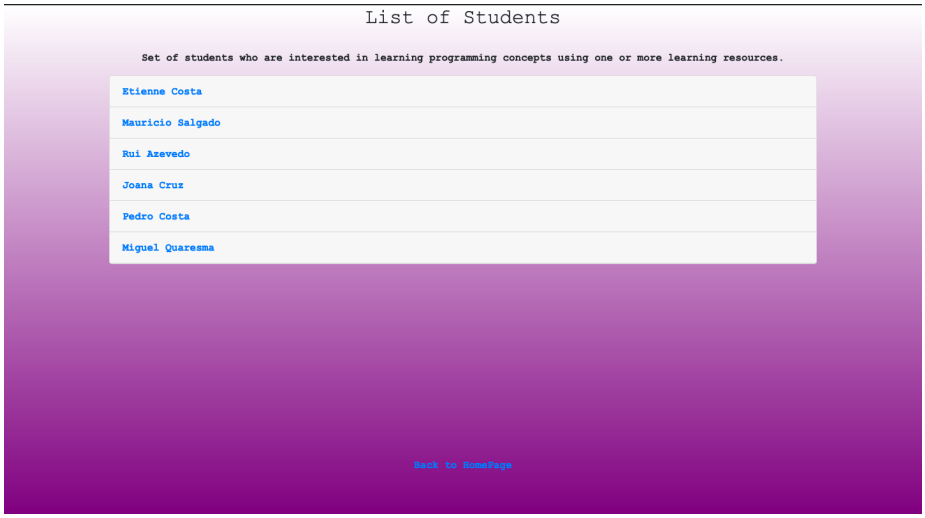

Figura 9: Lista de Alunos do Sistema.

4.7 Discentes e o Conjunto De Recursos

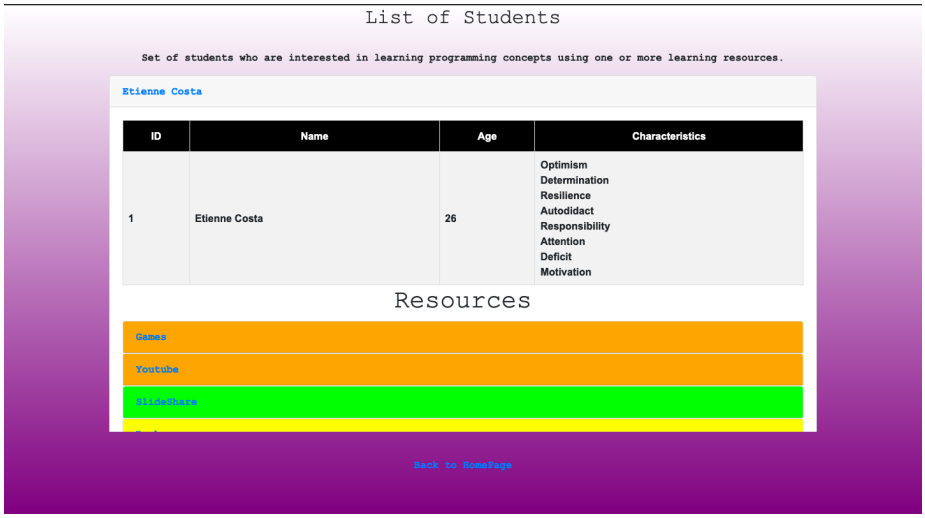

Figura 10: Recursos Associados ao Discente Etienne

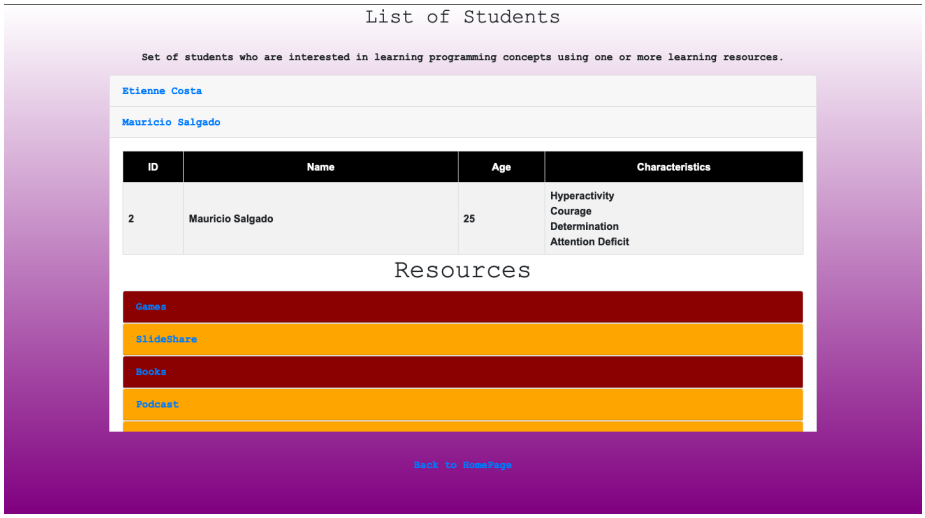

Figura 11: Recursos Associados ao Discente Maurício.

4.8 Funcionamento

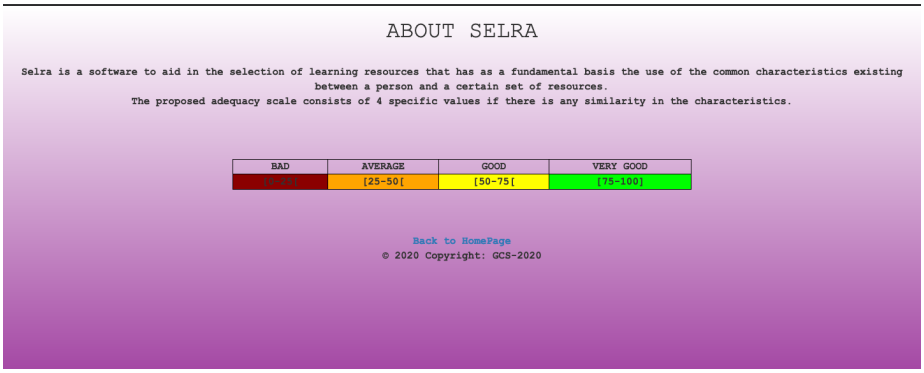

Figura 12: Intervalo de Adequação

5 Conclusão

De um modo geral, acreditamos que o software seja capaz de auxiliar numa fase inicial a aprendizagem do conjunto de discentes presentes no sistema partindo da premissa que o software seja povoado com diversos recursos e linguagens de programação de forma a garantir uma taxa de sucesso maior. Sendo assim , apresenta-se uma lista de resultados atingidos:

1. **Algoritmo de adequação satisfatório.**
2. **E um conjunto de discentes com recursos de aprendizagem.**

Como trabalho futuro propõe-se a realização do **refactoring**, pois todo código está susceptível de ser melhorado e através dos resultados obtidos da validação com o utilizador final fazer os ajustes que sejam adequados de modo a ter um algoritmo de adequação extremamente eficaz.