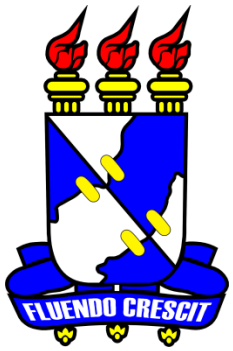

UFS - UNIVERSIDADE FEDERAL DE SERGIPE
CCET - CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DCOMP - DEPARTAMENTO DE COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO

COMP0235 - TURMA T01 – GRAFOS E ALGORITMOS COMPUTACIONAIS

PROJETO

PROFª. LEILA MACIEL DE ALMEIDA E SILVA

ARTUR SANTOS NASCIMENTO – 201310004513
IGOR NASCIMENTO DOS SANTOS – 201320007297

São Cristóvão - SE

Dia, 28 de Setembro de 2016

1. INTRODUÇÃO

Muitas vezes nos vemos diante de problemas de verificação, onde precisamos analisar vários documentos por erros ou inconsistências. A depender do problema ou da formalidade que ele exige, a quantidade de documentos pode se tornar muito grande, o que dificulta muito a verificação, e aumenta consideravelmente a chance de erros por parte humana, que aumenta mais ainda caso alguns documentos dependam de outros. Nesse caso, uma dependência cíclica pode tornar todo trabalho inválido.

Atualmente, vivemos um momento importante para todos cursos na Universidade Federal de Sergipe, onde estamos em processo de renovação das grades curriculares de todos cursos. Porém, o processo de confecção dos documentos necessários para efetivação da renovação da grade curricular é complexa, requer um alto grau de formalidade e a intolerância a inconsistência de dados.

Inspirados por essa problemática, que no caso do Departamento de Computação, se enrola a muito tempo, procuramos criar um aplicativo que pudesse auxiliar minimamente o processo de confecção desse tipo de documento no futuro, utilizando algoritmos de grafos para garantir consistência da grade curricular e que não há dependência cíclica entre as disciplinas.

2. MODELAGEM DOS DADOS

Antes de qualquer verificação, precisamos modelar os dados da grade curricular em um grafo, para que possamos executar algum algoritmo que garanta a não-ciclicidade dos pré-requisitos das disciplinas.

Foi observado que as grades curriculares são organizadas por nível curricular (ou período curricular). Os níveis contém disciplinas, sendo que os primeiros níveis são os que contém as disciplinas mais fundamentais e os níveis mais altos os que contém disciplinas mais avançadas no curso. A disposição dos níveis são tal que representam até mesmo um guia da sequência de disciplinas a serem cursadas pelo aluno até se graduar no nível cursado.

As disciplinas podem ou não conter pré-requisitos. Pré-requisitos são disciplinas que necessitam ter sido cursadas antes de cursar a disciplina em questão. Por exemplo, a disciplina Programação Orientada a Objetos possui como pré-requisito a disciplina Programação Imperativa. Além disso, cada disciplina possui um código único, pelo qual pode ser usado como chave única para busca.

Portanto, podemos modelar o problema com as disciplinas sendo os vértices do grafo, e a relação de pré-requisitos sendo as arestas. Se v é uma disciplina e w um pré-requisito de v, a aresta (v, w) direcionada, contribui com o grau de saída de v, e com grau de entrada de w.

3. METODOLOGIA

Para verificar a consistência da grade curricular, validando a não-ciclicidade dos pré-requisitos e a sequência ascendente em relação aos níveis do curso, utilizaremos o algoritmo de ordenação topológica.

Uma grade curricular consistente é acíclica pois qualquer aresta que exista liga um vértice de um nível para o vértice de um nível superior. Isso garante que nunca haverá um ciclo na grade, e por isso existe ordenação topológica para ela. Se existir ciclo, a grade curricular é inválida, algoritmo deve detectar este ciclo e retornar o erro ao usuário.

Segundo [SZWARFICTER, p.80], uma ordenação topológica de um dígrafo é uma ordem linear dos seus vértices de forma que todos os vértices apareçam antes de outros vértices cujo vértice possua aresta de saída. Uma explicação mais gráfica, seria dizer que, se desenharmos a ordenação topológica de um dígrafo, todas as arestas estarão orientadas apenas para direita ou apenas para esquerda.

```
Algoritmo ordenação topologica
inicio
        // Dados dígrafo D(V, E)
        Para j = 1 até n, faça:
        Inicio
                Escolha um vertice w com grau de entrada nulo em D
                Retirar de D o vertice w e as arestas dele divergentes
                Definir Vj := w
        Fim
Fim
```

Texto 1: Algoritmo de Ordenação topológica retirada do livro de Jayme Luiz Szwarficter, p80.

Podemos alterar o algoritmo para detectar ciclos no dígrafo, simplesmente alterando a linha que escolhe o próximo vértice a ser retirado [vide Texto 2].

```
Algoritmo ordenação topológica com detecção de ciclos
inicio
        // Dados dígrafo D(V, E)
        Para j = 1 até n, faça:
        Inicio
                Se houver vértice com grau de entrada nulo no grafo, então
                        Escolha um vertice w com grau de entrada nulo em D
                Senão  Grafo contém ciclo
                Retirar de D o vertice w e as arestas dele divergentes
                Definir Vj := w
        Fim
Fim
```

Texto 2: Algoritmo ordenação topológica com detecção de ciclos

Dessa forma, podemos implementar um algoritmo semelhante ao descrito no Texto 2 para solucionar o problema proposto.

4. TECNOLOGIAS USADAS E MODELAGEM DO PROBLEMA

4.1 TECNOLOGIAS

Após discussão entre as linguagens conhecidas pelos dois alunos, bem como pensando na interface de interação com o usuário, nossa decisão foi utilizar uma interface web, pois teríamos diversas opções para gerar uma interface agradável e amigável.

Utilizamos linguagem de programação PHP, versão 5, com servidor local Apache.

Para finalização da interface gráfica, usamos PaaS Cloud 9, para que trabalhássemos em paralelo sem complicações. Em todas etapas foram utilizados Sistemas Operacionais Linux (distribuições Gentoo e Ubuntu).

4.2 MODELAGEM

Construímos uma interface simples, constituída por uma página [Figura 1] onde se é submetido um arquivo com a estrutura de disciplinas da grade [Figura 4].

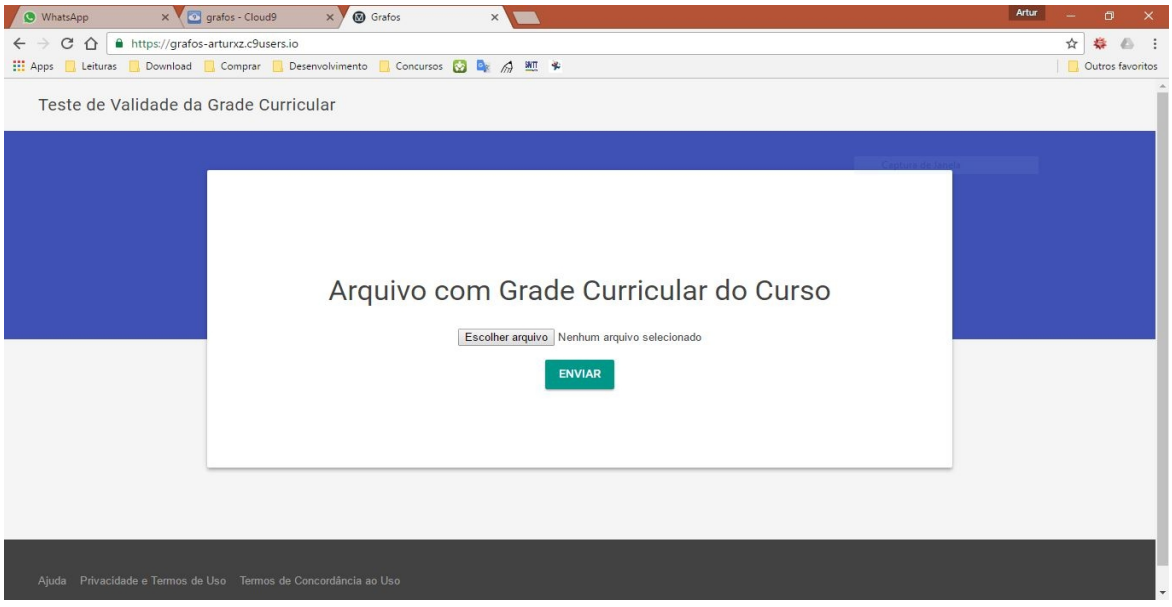

Figura 1: Tela inicial

Quando submetido, e clicado em “enviar”, uma nova página é gerada, mostrando a grade de horário submetida. Se houverem erros, será dito, e onde estiverem os erros, estará destacado na tabela [Figura 2].

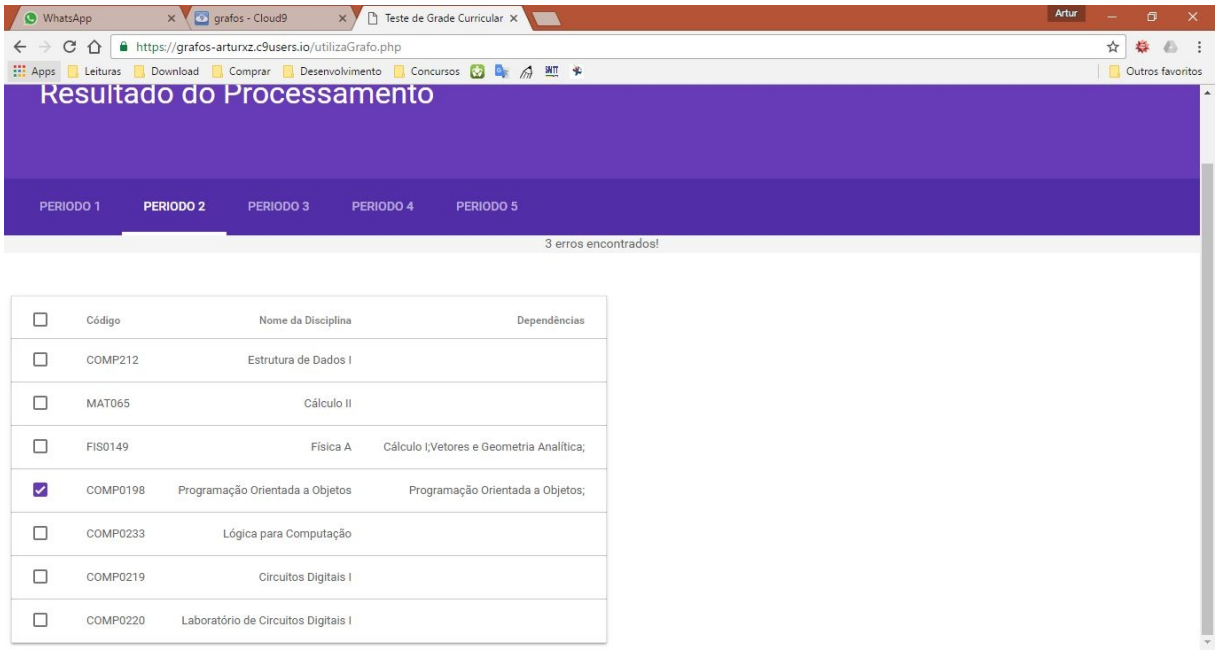

Figura 2: Parte do resultado, com um dos erros destacado.

Caso não tenham sido encontrados erros, apenas a grade é mostrada, atestando a validade da grade curricular [Figura 3].

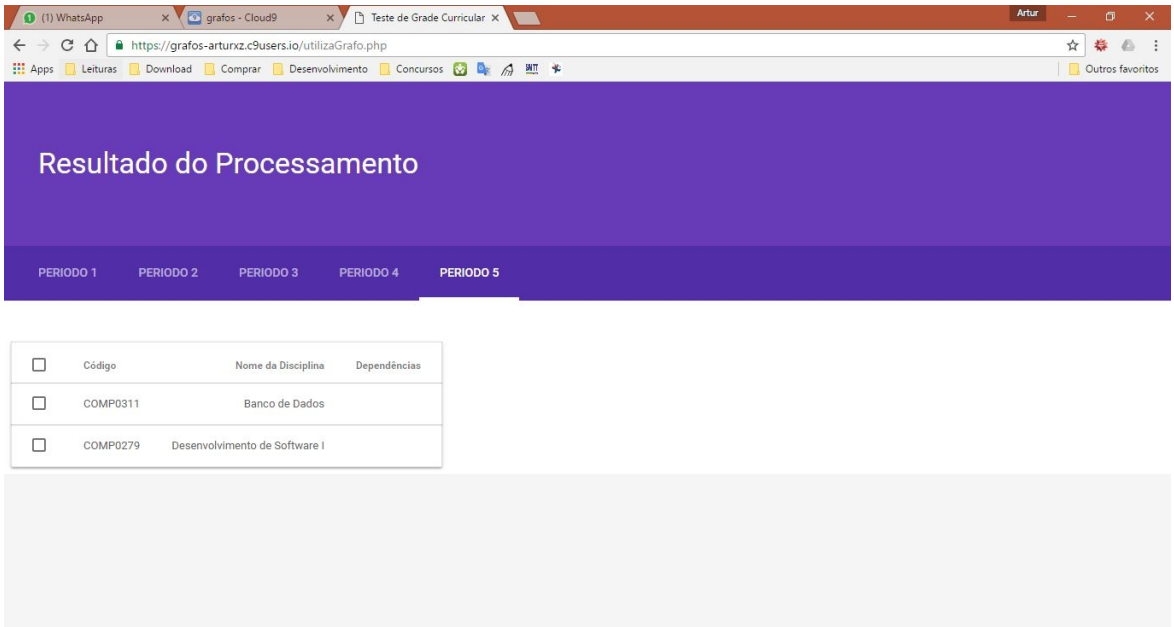

Figura 3: Parte do resultado, sem erros.

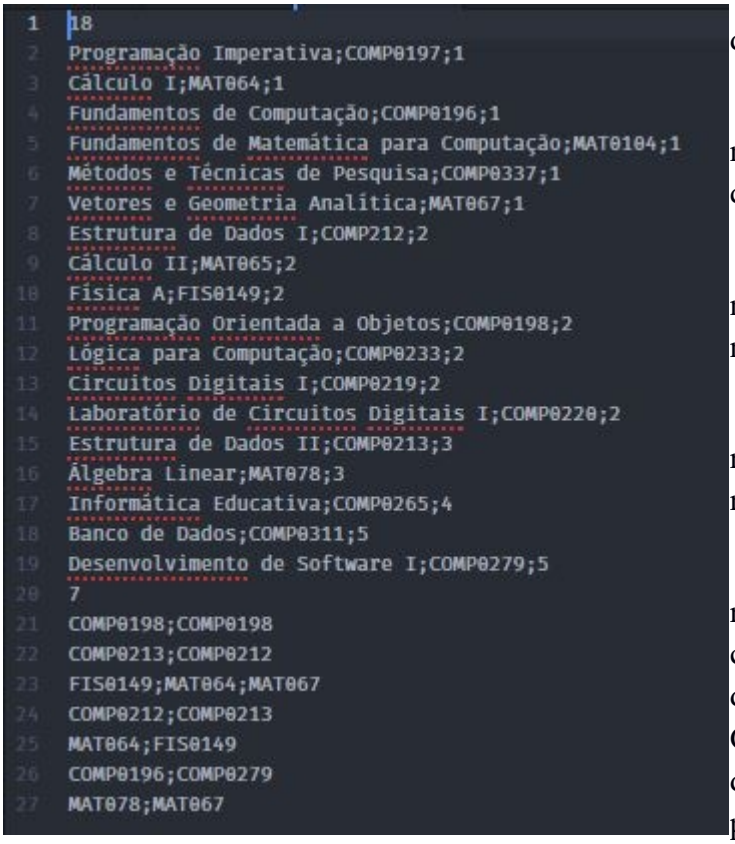

Figura 4: Estrutura do arquivo de submissão da grade

O arquivo submetido, contém os dados das disciplinas.

Na primeira linha, existe um número N que representa a quantidade de disciplinas da grade.

Seguem-se N linhas, contendo nome da disciplina, código e o nível, separados por ponto e vírgula.

Então, segue-se um número M, representando a quantidade de relações de pré-requisitos.

Em seguida, M linhas contendo as relações de pré-requisitos, com os códigos das disciplinas. No exemplo da figura 4, a linha 24 contém COMP0212;COMP0213. Isso quer dizer que a disciplina COMP0212 é pré-requisito da disciplina COMP0213.

5. RESOLUÇÃO DO PROBLEMA

O pseudocódigo que faz referência ao algoritmo utilizado para resolução do problema, lê o grafo e faz a ordenação topológica. Caso existam problemas de ciclicidade no grafo, eles são guardados em uma estrutura auxiliar. Essa estrutura é retornada no fim do algoritmo.

As figuras 5 e 6 mostram respectivamente, os pseudocódigos de visita e validação de grade, extraídos do algoritmo PHP usado para resolver o problema.

As linhas 09 a 12 do algoritmo de visita são as mais importantes para detecção do erro. Nelas, como temos garantido que a lista de vértices é ordenada crescentemente em relação ao nível, podemos nos dispor de que, se uma aresta contribuir para o grau de entrada de v, e w possuir nível maior que v, com certeza, há um ciclo no grafo. A variável erro, é uma variável global no problema, e adicionamos as arestas problemáticas lá.

```
ALGORITMO VISITA(v, P)
// ENTRADA: v é um vértice a ser visitado, vtx é a lista de vértices do grafo e P uma pilha
INICIO
    v.visitado := TRUE;

    PARA TODO w ADJACENTE A v, FAÇA
    INÍCIO
        i := 0;
        SE( ( w.visitado := FALSE ) ENTÃO
            VISITA(i, vtx, P);
        SENÃO
            erro.add( (w, v) );

        i = i + 1;
    FIM

    pilha.push(v);
FIM
```

Figura 5: Pseudocódigo do algoritmo de visita

Já o a função VALIDA_GRADE (Figura 6), apenas gerencia o início das visitas e inicializa a pilha. A configuração do algoritmo como Ordenação Topológica se dá pelo comportamento conjunto das duas funções.

```
ALGORITMO VALIDA_GRADE(vtx, n)
// vtx é a lista de vértices, ordenada em ordem crescente dos níveis das disciplinas, n é o total de vertices
INÍCIO
    PILHA P := VAZIO;

    PARA TODO v EM vtx, FAÇA
        v.visitado := FALSE;

    PARA i:=0 até n, FAÇA
        SE( vtx[i].visitado = FALSE ) ENTÃO
            erro := VISITA(i, P);

    RETORNE erro;
FIM
```

Figura 6: Pseudocódigo do algoritmo valida_grade

6. CONCLUSÃO

O aplicativo alcançou as expectativas em validar a grade no que tange à corretude da não-ciclicidade de pré-requisitos numa grade curricular. A interface é amigável e fácil de utilizar, bem como multiplataforma, visto que o benefício que diversos dispositivos podem acessar páginas de internet, conduzindo a uma experiência muito boa de utilização.

7. BIBLIOGRAFIA

CONCEITOS DE GRAFOS E PSEUDOCÓDIGOS INICIAIS

SZWARVFITER, Jayme Luiz, GRAFOS E ALGORITMOS COMPUTACIONAIS, 2ª Edição, Editora CAMPUS, 1986, Rio de Janeiro RJ;

INSTALAÇÃO E CONFIGURAÇÃO DO SERVIDOR APACHE

Wiki do Apache, Disponível em: <http://wiki.apache.org/general/>, (Acesso em 19/10/2016);

Wiki do Ubuntu, Disponível em: <https://wiki.ubuntu.com/>, (Acesso em 19/10/2016);

Wiki do Gentoo, Disponível em: <https://wiki.gentoo.org/>, (Acesso em 19/10/2016);

LINGUAGEM PHP

Wiki do PHP, Disponível em: <https://secure.php.net/>, (Acesso em 19/10/2016);

AMBIENTE DE PROGRAMAÇÃO EM CONJUNTO (PaaS)

PaaS Cloud9, disponível em: <https://c9.io>, (Acesso em 19/10/2016).