

FACULDADE DE EDUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO
- FAETERJ-Rio

FILIPE RODRIGUES CARDOSO DA SILVA

**UTILIZANDO COLORAÇÃO DE GRAFOS DE INTERSEÇÃO PARA RESOLVER O
PROBLEMA DE PROGRAMAÇÃO DE HORÁRIOS DE EXAMES EM
UNIVERSIDADES**

Rio de Janeiro
2020

FILIPE RODRIGUES CARDOSO DA SILVA

**UTILIZANDO COLORAÇÃO DE GRAFOS DE INTERSEÇÃO PARA RESOLVER O
PROBLEMA DE PROGRAMAÇÃO DE HORÁRIOS DE EXAMES EM
UNIVERSIDADES**

Trabalho de Conclusão de Curso apresentado à Faculdade de Educação Tecnológica do Estado do Rio de Janeiro – FAETERJ-Rio, como requisito parcial para obtenção de graduação em Tecnólogo em Análise de Sistemas Informátizados.

Orientador: Prof. Dr. José Wilson Coura Pinto

Rio de Janeiro

2020

C198m Silva, Filipe Rodrigues Cardoso da
Utilizando Coloração de Grafos de Interseção para Resolver o Problema
de Programação de Horários de Exames em Universidades / Filipe Rodrigues
Cardoso da Silva. – Rio de Janeiro: [s. n.], 2020.
109 p. 29 cm.

Trabalho de Conclusão de Curso (Graduação em Tecnólogo em Análise
e Desenvolvimento de Sistemas) – Faculdade de Educação Tecnológica do
Estado do Rio de Janeiro - FAETERJ-Rio, 2020.

1. Coloração de Grafos. 2. Grafos de Interseção. 2. Programação de Ho-
rários de Exames em Universidades. I. Silva, Filipe Rodrigues Cardoso da. II.
FAETERJ-Rio. III. Utilizando Coloração de Grafos de Interseção para Resolver
o Problema de Programação de Horários de Exames em Universidades

CDD 371.2

*“Não diga pouco em muitas palavras,
mas sim muito em poucas.
(PITÁGORAS)*

RESUMO

Ao agendar avaliações semestrais de uma universidade, pode-se identificar um clássico problema de combinatória, que vem sendo objeto de estudo na literatura desde a década de 1960. Este trabalho objetiva-se em apresentar o problema de programação de horários de exames universitários, suas características e restrições, bem como a modelagem matemática das turmas e alunos com a utilização de Grafos de Interseção. Será proposta uma heurística para aplicar o método de solução recorrendo à otimização de um problema de Coloração Generalizada de Grafos. Neste trabalho será apresentado um protótipo desenvolvido, e sua respectiva documentação, para permitir aos usuários a utilização do método de solução proposto de forma amigável.

Palavras-chave: Problema de Programação de Horários de Exames em Universidades. Grafos de Interseção. Coloração de Grafos. Otimização.

ABSTRACT

When scheduling semester examinations at a university, one can identify a classic combinatorial problem, which has been a study in the literature since the 1960s. This work aims to present the Examination Timetabling Problem, their characteristics and restrictions, as well as the mathematical modeling of classes with the use of Intersection Graphs. A heuristic will be proposed to apply the solution method using the optimization of a Generalized Graph Coloring Problem. In this work, will be presented a developed prototype and its respective documentation to allow users to use the proposed solution method in a friendly way.

Keywords: Examination Timetabling Problem. Intersection Graphs. Graph Coloring. Optimization.

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação geométrica de um Grafo.....	23
Figura 2 – Representação de uma rede de amigos do <i>Facebook</i> como um Grafo	25
Figura 3 – Representação de rede de seguidores do <i>Twitter</i> como um Grafo	25
Figura 4 – Representação de mapa de rotas como um Grafo.....	26
Figura 5 – Turmas modeladas como um Grafo	27
Figura 6 – Representação do Grafo após comparar as turmas A e B	28
Figura 7 – Representação do Grafo após comparar as turmas A e C	28
Figura 8 – Representação do Grafo após comparar as turmas B e C	28
Figura 9 – Representação do Grafo após comparar as turmas C e D	29
Figura 10 – Grafo após definir horário para turma A	31
Figura 11 – Grafo após definir horário para turma B	32
Figura 12 – Grafo após definir horário para turma C	32
Figura 13 – Grafo após definir horário para turma D	33
Figura 14 – Grafo colorido propriamente.....	34
Figura 15 – Exemplo de 4-coloração do mapa do Brasil	34
Figura 16 – Qual cor devo atribuir ao vértice C?.....	36
Figura 17 – Grafo ponderado em arestas	36
Figura 18 – Caso 1: Grafo colorido com vértice C vermelho.....	37
Figura 19 – Caso 2: Grafo colorido com vértice C azul	38
Figura 20 – Melhor coloração com duas cores para o Grafo	38

Figura 21 – Exemplo de Grafo colorido com algoritmo inicial	40
Figura 22 – Diagrama de Casos de Uso	50
Figura 23 – Diagrama de Atividades	57
Figura 24 – Diagrama de Classes	59
Figura 25 – Diagrama de Entidade Relacionamento	61
Figura 26 – Tela de Login	80
Figura 27 – Tela de Cadastro de Usuário	80
Figura 28 – Tela de Dashboard.....	81
Figura 29 – Tela de Criação de Novo Espaço de Trabalho	81
Figura 30 – Tela de Campo de Busca na Wikipédia	82
Figura 31 – Tela de Campo de Busca na Wikipédia Pop-up.....	82
Figura 32 – Tela Sobre o Sistema	83
Figura 33 – Tela de Contato	83
Figura 34 – Opções Rápidas do Espaço de Trabalho	84
Figura 35 – Tela de Definições do Espaço de Trabalho	84
Figura 36 – Tela de Definições do Espaço de Trabalho - Caixa de Seleção para Adicionar Dados	85
Figura 37 – Tela de Definições do Espaço de Trabalho - Caixa de Seleção para Exportar Dados	85
Figura 38 – Tela de Explicação da Solução através do uso do Robô.....	86
Figura 39 – Tela de Gerenciamento de Turmas	86
Figura 40 – Tela de Visualização da Turma	87
Figura 41 – Tela de Visualização da Turma – Cont.	87
Figura 42 – Tela de Edição de Turmas.....	88

Figura 43 – Tela de Edição de Turmas – Cont.	88
Figura 44 – Notificação ao Apagar Turma	89
Figura 45 – Tela de Criação de Nova Turma	89
Figura 46 – Tela de Criação de Nova Turma – Cont.	90
Figura 47 – Tela de Importação de Turmas por Arquivo	90
Figura 48 – Tela de Alocar Turmas em 2 Horários	91
Figura 49 – Tela de Alocar Turmas em 3 Horários	91
Figura 50 – Tela de Alocar Turmas em 3 Horários – Menu	92
Figura 51 – Tela de Alocar Turmas – Exemplo 1	93
Figura 52 – Tela de Alocar Turmas – Exemplo 2	93
Figura 53 – Tela de Alocar Turmas – Exemplo 3	94
Figura 54 – Tela de Alocar Turmas – Exemplo 4	94
Figura 55 – Tela de Alocar Turmas – Exemplo 5	95
Figura 56 – Tela de Alocar Turmas – Exemplo 6	95
Figura 57 – Tela de Alocar Turmas – Exemplo 7	96
Figura 58 – Gerando uma Grade de Prova – pt. 1	96
Figura 59 – Gerando uma Grade de Prova – pt. 2	97
Figura 60 – Gerando uma Grade de Prova – pt. 3	97
Figura 61 – Gerando uma Grade de Prova – pt. 4	98
Figura 62 – Gerando uma Grade de Prova – pt. 5	98
Figura 63 – Gerando uma Grade de Prova – pt. 6	99
Figura 64 – Gerando uma Grade de Prova – pt. 7	100
Figura 65 – Gerando uma Grade de Prova – pt. 8	100
Figura 66 – Gerando uma Grade de Prova – pt. 9	101

Figura 67 – Gerando Relatório 101

LISTA DE ABREVIATURAS E SIGLAS

PPH	Problema de Programação de Horários
PPHE	Problemas Programação de Horários em Escolas
PPHC	Problemas Programação de Horários de Cursos
PPHEU	Problemas de Programação de Horários de Exames em Universidades
GGCP	<i>Generalized Graph Coloring Problem</i>
LF	<i>Largest-first</i>
HTML	<i>Hypertext Markup Language</i>
CSS	<i>Cascading Style Sheet</i>
PHP	<i>PHP: “Hypertext Processor”</i>
RF	Requisito do Software
RN	Regra de Negócio
UC	<i>User Case</i> (Caso de Uso)
DER, ER	Diagrama de Entidade-Relacionamento
ICT	<i>International Competition Timetabling</i>

LISTA DE SÍMBOLOS

$G, G(V, E)$	um grafo simples.
$V(G)$	conjunto de vértices de um grafo.
v, w	um vértice específico pertencente à $V(G)$.
$E(G)$	conjunto de arestas de um grafo.
e	uma aresta específica pertencente à $E(G)$.
$ V , n$	número de nós (vértices de um grafo).
$H(V, E)$	um multigrafo.
$D(V, E)$	um grafo direcionado (digrafo).
\mathbb{N}	conjunto dos números naturais.
p	um rótulo de peso pertencente à \mathbb{N} .
$p(v, w), p(e)$	um peso atribuído a uma aresta.
\in	ícone matemático de pertence.
k	quantidade de horários disponíveis para realização das provas.
K	conjunto dos horários disponíveis para realização das provas $\in \mathbb{N}$.
$\chi(G)$	número cromático do grafo.
C	conjunto das possíveis colorações para o grafo.
c	uma coloração para o grafo.
P_c	quantidade de provas de segunda chamada necessárias para uma coloração c .

LISTA DE TABELAS

Tabela 1 – Lista de turmas e a matrícula seus respectivos alunos	27
Tabela 2 – Relação entre os vértices e os horários	30
Tabela 3 – Relação entre horários e cores	31
Tabela 4 – Alocação das turmas em três horários.....	33
Tabela 5 – Relação entre horários e cores	38
Tabela 6 – Passos para serem executados pela heurística	42
Tabela 7 – Fluxo Principal de Interações de [UC01].....	51
Tabela 8 – Sequência de Interações de [UC02]	52
Tabela 9 – Fluxo Principal de Interações de [UC03].....	52
Tabela 10 – Fluxo de Interações de [UC04] – Criação de Turma.....	53
Tabela 11 – Fluxo de Interações de [UC04] – Visualização de Turma	54
Tabela 12 – Fluxo de Interações de [UC04] – Edição de Turma.....	54
Tabela 13 – Fluxo de Interações de [UC04] – Exclusão de Turma	55
Tabela 14 – Fluxo Principal de Interações de [UC05].....	56
Tabela 15 – Exemplos de alocações e seus resultados	92

SUMÁRIO

	INTRODUÇÃO.....	16
2	PROBLEMAS DE ESCALONAMENTO DE HORÁRIOS	19
2.1	Timetabling	19
2.1.1	Problemas de Programação de Horários Educacionais (<i>Educational Timetabling</i>)	19
2.2	Problema de Programação de Horários de Exames em Universidades	20
2.3	Criação de Horários de Exames em Universidades.....	21
2.4	Variações do problema.....	21
2.4.1	Alocação de Bandas em dias de um festival.....	21
2.4.2	Escalas de Equipes em uma Companhia Aérea	22
2.4.3	Distribuição de Recursos Militares	22
3	MODELAGEM MATEMÁTICA.....	23
3.1	Conceitos Gerais em Teorias dos Grafos.....	23
3.2	Exemplos de problemas modelados com Grafos	24
3.2.1	<i>Facebook e Twitter</i>	24
3.2.2	<i>Google Maps e Waze</i>	26
3.3	Grafo de Interseção de Turmas	26
4	MÉTODO DE SOLUÇÃO	30
4.1	Coloração Própria de Vértices de um Grafo	33

4.2	Limitação no número de Horários de Prova	35
4.3	Grafos Ponderados em Arestas	36
4.4	Generalized Graph Coloring Problem	37
4.5	Complexidade do Problema e Viabilidade de Solução	39
4.6	Algoritmo Desenvolvido Inicialmente	39
4.7	Heurística Proposta	40
5	SOFTWARE PARA AGENDAMENTO DE HORÁRIOS DE PROVAS ...	46
5.1	Tecnologias Utilizadas no Desenvolvimento	46
5.2	Especificações do Software	48
5.2.1	Requisitos do Software	48
5.2.2	Regras de Negócio	49
5.2.3	Casos de Uso	50
5.2.3.1	Descrição dos Casos de Uso	50
5.2.4	Diagrama de Atividades	56
5.2.5	Diagrama de Classes	58
5.2.6	Diagrama de Entidade-Relacionamento	60
5.2.7	Relatórios	60
6	RESULTADOS OBTIDOS E TRABALHOS FUTUROS.....	62
6.1	Trabalhos Futuros	63
7	CONSIDERAÇÕES FINAIS.....	64
7.1	Limitações da Pesquisa	64
7.2	Recomendações para Pesquisas Futuras.....	65

REFERÊNCIAS	67
--------------------------	-----------

APÊNDICES	74
------------------	-----------

APÊNDICE A – ROTEIRO	75
-----------------------------------	-----------

ANEXOS	78
---------------	-----------

ANEXO A – PROTÓTIPO	79
----------------------------------	-----------

A.1 Instalação.....	79
---------------------------------	-----------

A.1.1 Pré-requisitos do Projeto	79
--	----

A.2 Utilização	79
----------------------------------	-----------

A.2.1 Cadastro de Usuários	80
-------------------------------------	----

A.2.2 Dashboard.....	81
-------------------------	----

A.2.3 Definições do Espaço de Trabalho	84
---	----

A.2.4 Gerenciar Turmas	86
---------------------------------	----

A.2.5 Alocar Turmas em Horários de Prova	90
---	----

A.2.6 Gerar Relatório da Alocação	96
--	----

A.2.7 Imprimir Relatório.....	100
----------------------------------	-----

ANEXO B – RELATÓRIO	102
----------------------------------	------------

INTRODUÇÃO

Os problemas de programação de horários vem sendo estudados desde o início da década de 60, são problemas cotidianos e muitas vezes de difícil resolução. A dificuldade surge devido à natureza combinatória do problema, o que demanda grande esforço e tempo para se solucionar manualmente em alguns casos. Para isso é comum se propor soluções computacionais que sejam capazes de buscar a melhor programação dos horários no menor tempo e com menor esforço possível.

Para permitir a utilização de computadores na resolução de um problema cotidiano, precisaremos realizar um processo de modelagem na qual transformaremos o problema real em um problema de computador, e a partir desse momento utilizaremos os algoritmos, possibilitando que o computador execute uma série de instruções específicas de forma sequencial.

Este trabalho apresentará o processo de modelagem e confecção de um protótipo de sistema para solucionar um clássico problema de natureza combinatória, observado em universidades.

Apresentaremos ao longo deste trabalho uma proposta de resolução para o **Problema de Programação de Horários de Exames em Universidades** (PPHEU), nomeado por Schaefer (1999) e se caracteriza por ser de difícil resolução manual e bastante estudado por pesquisadores da área.

A aplicação de exames, provas ou avaliações em universidades, se torna um problema de natureza combinatória em algumas circunstâncias, e caso isso venha a ocorrer, encontrar uma solução perfeita pode ficar até mesmo impossível em alguns cenários. Nesses casos é recomendável a busca da melhor solução possível, maximizando (ou minimizando) determinado aspecto dela visando encontrar uma solução ótima através dessa otimização.

Os problemas de programação de horários de exames universitários foram vastamente estudados em trabalhos como os de Kiaer e Yellen (1992), Socha, Sampels e Manfrin (2003), Lai et al. (2008), Burke et al. (2010) entre outros, onde são utilizadas diversas formas de modelagem e métodos de solução, contudo foram os trabalhos de Brown e Corneil (1987), Vredeveld (2002) e Scheeren, Pereira e Vaz (2018) que utilizaram as formas de modelagem ou solução mais similares as apresentadas neste trabalho.

A **programação de horários de exames universitários**, consiste na criação de uma tabela de horários para que diversas turmas de uma universidade realizem suas provas semestrais. Contudo, condições adicionais podem ser destacadas, por afetarem consideravelmente a solução e por consequência o resultado obtido.

Inicialmente, deve-se compreender turma em um contexto universitário, isto é, uma

turma está diretamente relacionada a uma única disciplina ofertada semestralmente, e que seus alunos podem ou não integrarem ela, de acordo com o interesse do aluno e das normas da instituição. Com isso, temos um ambiente onde duas turmas podem ou não ter alunos em comum.

Ao estipularmos o horário em que uma turma deverá realizar uma prova, deveremos considerar que se duas, ou mais, turmas com alunos em comum realizarem seus exames de forma simultânea, os alunos terão avaliações em diferentes turmas no mesmo horário, com isso serão prejudicados e deverão realizar a segunda chamada dos exames que não puderam fazer em decorrência da alocação dos horários.

Caso uma instituição possua uma quantidade limitada de horários para a realização das provas, poderá não ser possível realizar a programação dos exames de forma a nenhum aluno ser prejudicado por conflitos nos horários de prova, assim buscaremos minimizar o número de provas de segunda chamada necessárias.

Este trabalho trará uma abordagem em que se obtém o melhor resultado possível considerando as turmas, seus alunos e a quantidade de horários de provas disponíveis na instituição. Será apresentado conceitualmente e de forma prática na aplicação proposta, a melhor alocação e suas possíveis consequências, permitindo dessa forma, que sejam realizadas as aplicações das provas de segunda chamada.

O objetivo deste trabalho é apresentar formalmente o problema de programação de horários de exames universitários com suas respectivas classificações e restrições, demonstrar a modelagem matemática para o problema e propor um algoritmo capaz de obter a melhor solução, além de apresentar um sistema que permita a busca pela resolução de forma automática e que seja amigável para qualquer usuário.

A modelagem do problema utilizará das estruturas matemáticas da **Teoria dos Grafos**, mais especificamente de **Grafos de Interseção** e **Coloração de Vértices**. Este modelo nos beneficiará com um enorme acervo de pesquisas, assim como observado em Carlson e Nemhauser (1966), Bondy e Murty (1978), Brown (1996), Kann et al. (1997) e Leung (2004), que ajudam a facilitar a abordagem sobre tema, além de nos proporcionar uma adequação simplificada do modelo conceitual em um protótipo funcional.

Com o desenvolvimento de uma solução computacional, poderemos automatizar o processo de busca por soluções, permitindo obter uma solução com uma expressiva redução no tempo de busca em comparação com métodos manuais.

O problema de programação de horários de exames universitários, pertence à família de problemas bem conhecidos na computação e matemática, os **problemas de tabela de horários** (também conhecidos como *Timetabling* ou *Timetable*).

Existem alguns exemplos de implementações amigáveis para o usuário final (*user-friendly*), voltadas para o *Timetabling* como, por exemplo, o “**Cronos**” de 4cis Sistemas Ltda.

(2020), o “**GridClass**” de Grupo WPensar (2020) e o “**Urânia**” de Geha Sistemas Especialistas (2020) sendo todos estes voltados para o problema de programação de horários escolares, onde, de forma simplificada, busca-se alocar professores e alunos em salas, em horários específicos, para a realização das aulas.

Este trabalho está organizado da seguinte forma.

No capítulo 2 são apresentados os **Problemas de Escalonamento de Horários**, sua origem e suas subdivisões, incluindo a que pertence o problema de programação de horários de exames universitários.

No capítulo 3 é demonstrada a **Modelagem Matemática**, apresentando fundamentos sobre Teoria dos Grafos e a forma pela qual os grafos de interseção foram utilizados.

No capítulo 4 é apresentado o **Método de Solução** que explica a forma pela qual utilizamos a modelagem apresentada no capítulo anterior para resolver o problema e os algoritmos propostos para isso.

No capítulo 5 está a **Documentação UML** relativa à aplicação desenvolvida, com as tecnologias usadas, metodologias e especificações do “software”.

No capítulo 6 estão descritos os **Resultados Obtidos** com o desenvolvimento do trabalho e são apresentadas as motivações e objetivos para os **Trabalhos Futuros**.

Por fim é apresentada no capítulo 7 as **Considerações Finais** acerca do trabalho desenvolvido, as **Limitações de Pesquisa** observadas ao decorrer do trabalho e as **Recomendações para Pesquisas Futuras**, onde são apresentadas as recomendações para futuros estudos e trabalhos de motivação similar ao apresentado.

Em apêndice A está descrito o roteiro para realização de um teste de usabilidade do sistema proposto, onde busca-se observar se os objetivos do desenvolvimento foram atingidos juntamente a usuários reais.

No anexo A está o **Manual** para instalação e utilização do protótipo criado com base nas especificações do capítulo 5, enquanto o anexo B apresenta um exemplo do **Relatório** sobre o resultado emitido pelo protótipo ao final da resolução do problema.

2 PROBLEMAS DE ESCALONAMENTO DE HORÁRIOS

Problemas de escalonamento de horários (*scheduling*), são problemas amplamente estudados na matemática e computação, são tidos como problemas clássicos de otimização combinatória. Resumidamente, Wren (1996) define esses problemas como: agendar recursos em horários, respeitando condições prévias, de forma a minimizar o custo total de algum conjunto de recursos utilizados.

O problema de escalonamento é vastamente explorado na literatura por ser aplicável a muitas situações que envolvam alocar um ou mais recursos (como pessoas, veículos, maquinário, processadores e etc) em um quadro limitado de horários, onde se deseja maximizar (ou minimizar) alguma característica da solução que esteja em função de algum recurso, sem violar as regras inerentes ao problema enunciado. Na literatura muitas classes de sub-problemas de escalonamento são apresentados, dentre essas, uma que se destaca é o Problema de Programação de Horários (PPH), também chamado de *timetabling*.

2.1 Timetabling

O PPH foi primeiramente apresentado por Gotlieb (1963), demonstrando o problema de alocar professores, turmas e salas em determinados horários de aula, de forma a não se ter um mesmo professor em duas salas em um mesmo horário ou dois professores em uma mesma sala simultaneamente.

O PPH é definido por Wren (1996) como sendo a atribuição, sujeita a restrições, de recursos a eventos dispostos em quantidade de espaço e tempo limitados, visando satisfazer da melhor forma um conjunto de objetivos estabelecidos. Contudo, essa atribuição pode vir a possuir um alto grau de complexidade de acordo com as restrições e os objetivos a serem atingidos.

Existe um grande volume de informações referentes ao PPH, sendo apresentadas formas de solução, aplicação e classificação. Em Rocha (2013) os problemas de PPH são classificados de acordo com suas aplicações, podendo ser: Escalas de Trabalhadores (*Employee Timetabling*), Escalas de Condutores de Veículos de Transporte (*Transportation Timetabling*), Escalas de Jogos de Competições Esportivas (*Sports Timetabling*), Educacional (*Educational Timetabling*).

2.1.1 Problemas de Programação de Horários Educacionais (*Educational Timetabling*)

Os PPH Educacionais são os problemas cujas características (restrições e recursos) são semelhantes a situações enfrentadas em instituições de ensino, como em definir o quadro de aulas, calendário de provas universitárias, distribuição de turmas em salas de aula entre outros. Segundo Santos e Souza (2007), os PPH da área educacional podem ser especificados em três categorias:

Problemas Programação de Horários em Escolas (PPHE), Problemas Programação de Horários de Cursos (PPHC) e Problemas de Programação de Horários de Exames em Universidades (PPHEU).

Em Schaerf (1999) pode ser observada a seguinte classificação:

- PPHE: A programação semanal para todas as classes de uma escola, evitando que os professores possuam duas aulas ao mesmo tempo, e vice-versa;
- PPHC: A programação semanal de todas as palestras de um conjunto de disciplinas universitárias, minimizando as sobreposições de palestras de disciplinas que possuem alunos comuns;
- PPHEU: O agendamento para os exames de um conjunto de disciplinas universitárias, evitando a sobreposição de exames de cursos com alunos comuns, e evitando, o melhor possível, que os alunos tenham exames muito próximos.

Os PPH Educacionais estão entre os problemas mais difíceis da área de otimização combinatória. Podendo ainda serem aplicadas inúmeras novas restrições a problemas já enunciados, de modo a tornar o resultado obtido o mais próximo do ideal e/ou do mundo real, entretanto a dificuldade de se encontrar a melhor solução possível aumenta.

Em Schaerf (1999) é visto que os PPH Educacionais são classificados como NP-Completo para a maioria das formulações. Dessa forma, a solução exata em curto tempo de execução só pode ser garantida para instâncias pequenas, que muitas vezes podem não corresponder à realidade da maioria das instituições de ensino.

2.2 Problema de Programação de Horários de Exames em Universidades

O problema consiste em definir horários de provas para diferentes turmas universitárias. Sendo uma turma universitária relacionada a apenas uma disciplina, diferente do contexto escolar, em que uma turma tem diversas disciplinas.

Um mesmo aluno pode pertencer a diferentes turmas, dessa forma para que ele possa realizar suas provas, busca-se evitar a atribuição em um mesmo horário de diferentes avaliações para um mesmo aluno. Caso não seja possível criar uma grade de horários sem prejudicar nenhum aluno, sendo permitida a realização de avaliações de segunda chamada para aqueles que possuírem duas ou mais provas em um mesmo horário.

Com isso deseja-se alocar as provas em horários buscando o menor número de avaliações de segunda chamada possível.

Neste trabalho iremos considerar esse problema como um problema de programação de horários de cursos universitários (PPHC). Que Schaerf (1999) define como um problema

semelhante ao de definir horários de exames (PPHEU), contudo, ele adverte que em situações onde as turmas são heterogêneas, o comum é definir o problema como um PPHC, onde os cursos (ou palestras) serão tratados como horários de avaliações.

2.3 Criação de Horários de Exames em Universidades

Uma solução trivial para o problema consistiria em ter um número de horários para realização da prova maior ou igual à quantidade de turmas. Com isso bastaria atribuir cada turma a um horário e não haveria nenhuma sobreposição e por consequência nenhuma prova de segunda chamada.

Contudo, este trabalho focará em explorar as situações onde uma solução trivial não possa ser adotada. Isto é, quando a quantidade de turmas for maior que o número de horários disponíveis.

Nesse cenário algumas turmas terão que, obrigatoriamente, compartilharem o mesmo horário de prova. Dessa forma, caso duas turmas possuam alunos em comum, a aplicação da prova de segunda chamada seria necessária para os alunos prejudicados.

Com isso o trabalho irá explorar como identificar as turmas com alunos em comum, de modo a minimizar a alocação em um mesmo horário dessas turmas. Dessa maneira minimizar a quantidade de provas de segunda chamada.

2.4 Variações do problema

No decorrer da pesquisa para realização deste trabalho, alguns problemas foram analisados (e consequentemente modelados) e pôde-se notar uma similaridade entre suas características.

Acreditamos que existam outros problemas que tenham características similares à estes apresentados, com isso a proposta de solução algorítmica (apresentada no **capítulo 4**), visará resolve-los independente da descrição, obtendo resultados com a mesma qualidade.

2.4.1 Alocação de Bandas em dias de um festival

Um evento de música popular visa reunir diversos artistas, músicos independentes e amadores para se apresentar em diferentes palcos e horários.

Muitas bandas possuem membros em comum (definidos aqui como: um conjunto de pessoas que integrem a parte artística uma apresentação, seja instrumentista, dançarino, um técnico ou outros). Deseja-se evitar que muitas bandas sejam prejudicadas com a alocação de seus membros em diferentes palcos em um mesmo horário, pois, isso causaria um desfalque em uma das bandas.

Caso não seja possível evitar todos os desfalques, deve-se minimizar eles para que o festival ocorra do melhor modo possível.

2.4.2 Escalas de Equipes em uma Companhia Aérea

Uma companhia aérea separa seus comissários em diferentes equipes, podendo um comissário de voo estar em mais de uma equipe, a empresa deseja organizar os grupos a trabalhar em seus voos de forma que as equipes possam trabalhar com a menor quantidade de desfalques possível.

Caso um comissário faça parte de uma ou mais equipes escaladas em um mesmo dia, esse desfalcará as demais equipes, pois, só poderá estar em um único voo, e um substituto deverá ser contratado para completar o grupo.

Deseja-se distribuir as equipes em diferentes voos buscando-se minimizar a quantidade de desfalques, e por consequência, contratações de substituintes.

2.4.3 Distribuição de Recursos Militares

Em uma operação militar existem diversos recursos, como equipamentos, armamentos, munições e especialistas, que devem ser distribuídos de melhor forma possível visando garantir o sucesso da missão.

Ao se distribuir soldados em operações em diferentes lugares, busca-se formar separar as unidades militares de forma que subunidades sejam capazes de atuar de forma simultânea. Um soldado dessa forma é preparado para poder atuar em diferentes grupos caso seja necessário.

Quando diferentes operações demandam o destacamento das subunidades o oficial deve designar os soldados de forma a manter os grupos o mais coesos e similares aos grupos com os quais os soldados já realizaram treinamento.

Dessa maneira, um soldado de uma unidade pode pertencer a qualquer uma das subunidades com a qual já realizou treinamento, e caso seja designado a realizar uma operação com esse grupo, irá desfalar os demais.

Deseja-se buscar uma distribuição das subunidades de forma que ocorra o menor número possível de desfalques nas operações. Visando concluir da melhor forma a missão recebida.

3 MODELAGEM MATEMÁTICA

Visando permitir o uso do poder computacional na busca de uma solução para o problema apresentado no capítulo 2, necessita-se modelar o problema de forma matemática.

Dessa forma foi optado o uso da teoria dos grafos como estrutura matemática visando modelar o problema.

3.1 Conceitos Gerais em Teorias dos Grafos

Um grafo simples $G(V, E)$ é definido por Szwarcfiter (1986) como sendo uma estrutura matemática composta por um conjunto finito não vazio de vértices V e um conjunto E de pares, não ordenados, de diferentes vértices denominados arestas.

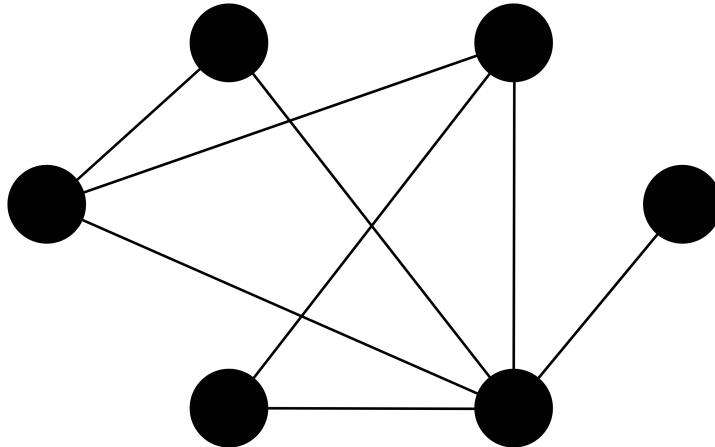


Figura 1 – Representação geométrica de um Grafo

Um grafo é comumente representado graficamente conforme a figura 1, onde os vértices são representados por círculos e as arestas por linhas que interligam dois vértices.

Elementos componentes de um grafos conforme definidos por Szwarcfiter (2018) e Pinto (2018) a seguir.

Definição 1. Vértice: um vértice é denotado por $V(G)$ é representado graficamente através de um ponto, a quantidade dos vértices de um grafo é expressa por $|V|$, onde um grafo com $|V| = 1$ é classificado como um grafo trivial.

Definição 2. Aresta: uma aresta $e \in E(G)$ é denotada por um par de vértices $e = (v, w)$. Dessa forma os vértices v e w estão conectados as extremidades da aresta e , dessa forma são denominados adjacentes. Já uma aresta é denominada incidente aos vértices em seus extremos v e w .

Definição 3. *Grau de um vértice:* o grau de um vértice $v \in V(G)$, é denotado por $\text{grau}(v)$, sendo esse o número de vértices adjacentes a v .

Definição 4. *Multigrafo:* denotado por $H(V,E)$, é um grafo que admite diferentes arestas incidentes a um mesmo par de vértices (v,w) ou laços, sendo arestas que são incidentes a um único vértice (v,v) .

Definição 5. *Grafos Direcionados:* em um grafo direcionado (digrafo) $D(V,E)$ é um conjunto finito não vazio $V(G)$ (os vértices), e um conjunto $E(G)$ (as arestas) de pares ordenados de vértices distintos. Portanto, também que (v,w) possui uma única direção de v para w .

Definição 6. *Grafo Ponderado em Arestas:* um grafo é classificado como ponderado quando é atribuído um valor (peso) p as arestas, sendo esse $p(v,w)$ ou $p(e)$.

Definição 7. *Rótulo:* um rótulo consiste na atribuição de um valor (numérico ou alfabético) que simbolize ou represente um elemento do grafo de modo a facilitar a identificação de um vértice ou aresta específica em uma representação.

A Teoria dos Grafos é uma área amplamente estudada tanto na matemática quanto na computação. Muitos problemas do mundo podem ser descritos como problemas de grafos, dessa forma, ao se propor uma solução para o modelo de grafos também de obtém a solução para o problema real.

Muitos problemas e soluções já foram propostos com o passar dos anos, isso possibilita que a teoria dos grafos tenha modelos que se adéquem a muitas descrições, e algoritmos (sequência de instruções para se fazer uma operação no computador) eficientes para resolvê-los.

3.2 Exemplos de problemas modelados com Grafos

Atualmente muitos serviços tecnológicos utilizam das estruturas de grafos para resolverem problemas do dia-a-dia das pessoas. Esses serviços podem ser aplicativos para *smartphones*, sistemas *web* ou até mesmo sistemas *desktop*.

Dentre os aplicativos que utilizam de modelos em grafos para resolver seus problemas, destacam-se os seguintes exemplos:

3.2.1 Facebook e Twitter

As mídias sociais que se popularizaram desde a criação da “internet”, muitas podem ter suas aplicações modeladas com uso de grafos, e não apenas isso, mas algumas dessas mídias são criadas de forma a obter benefícios dessas estruturas.

Dois exemplos podem ser destacados pela forma diferente de se utilizar a estrutura dos grafos, o **Facebook** e o **Twitter**.

O *Facebook* pode ter as relações de amizade representadas como um grafo simples, onde duas pessoas que sejam amigos na plataforma representariam uma relação. Nesse caso cada usuário será representado por um vértice $v \in V$, e usaremos uma aresta (v, w) para representar relações de amizade entre dois usuários. Como apresentado na figura 2.

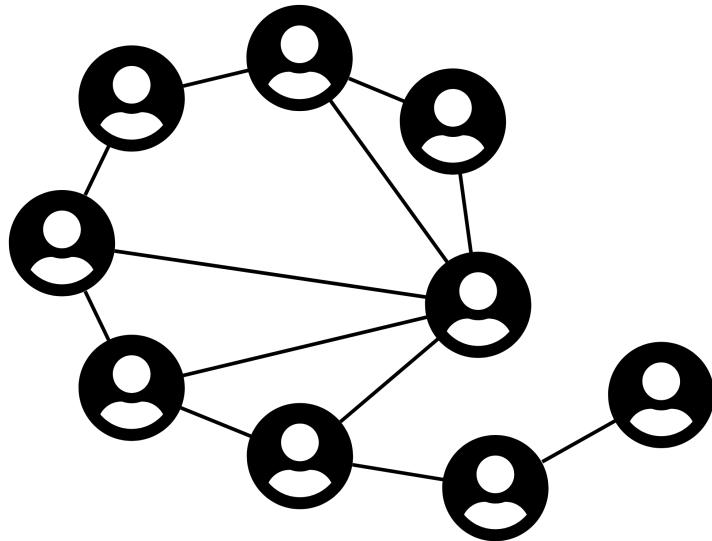


Figura 2 – Representação de uma rede de amigos do *Facebook* como um Grafo

Diferente do caso do *Facebook*, onde para termos uma relação de amizade precisamos que os dois usuários sejam amigos entre si, o *Twitter* apresenta o conceito de seguir outros usuários, e não necessariamente ser seguido por eles.

Dessa forma, a representação dessas relações seriam expressas melhor por um multigrafo direcionado, onde os usuários são representados pelos vértices v , e uma aresta (v, w) representará aqueles que ele segue, sendo essas arestas com sentido, com a origem no usuário que segue v e o destino no usuário que é seguido w . Dessa forma podemos observar essas diferenças na figura 3.

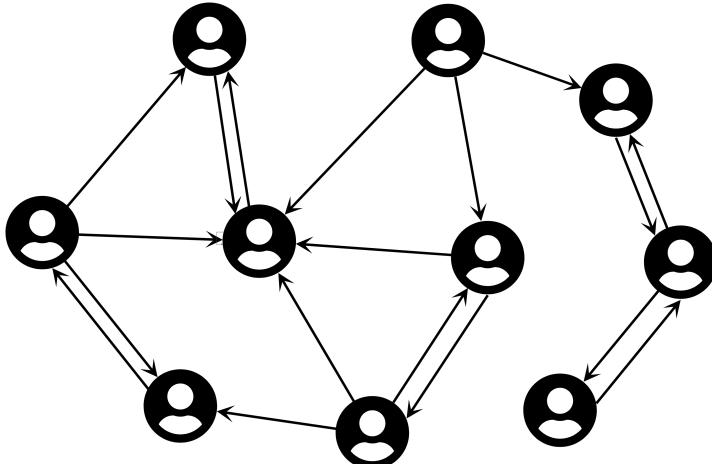


Figura 3 – Representação de rede de seguidores do *Twitter* como um Grafo

3.2.2 Google Maps e Waze

Aplicativos de geolocalização atuais podem utilizar grafos para determinar melhores rotas, podemos considerar como vértices os locais (ou pontos de referência) no qual o usuário deverá passar, e as arestas como o percurso que interligará esses dois pontos.

De acordo com a necessidade, o grafo pode ganhar informações adicionais, como pesos atribuídos as arestas, simbolizando a distância entre dois pontos, ou a velocidade média de outros veículos na travessia entre os dois pontos, esses fatores serão considerados para se propor uma rota, seja a que cruze o menor número de pontos, a que percorra a menor distância total, ou que obtenha a maior velocidade média de modo a percorrer em menor tempo a rota.

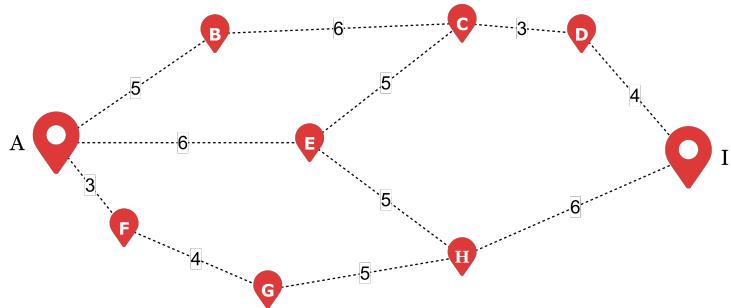


Figura 4 – Representação de mapa de rotas como um Grafo

3.3 Grafo de Interseção de Turmas

Pinto (2018) define um grafo de interseção, quando seja possível esse grafo $G(V, E)$ representar as relações entre elementos de um conjunto através das relações entre seus vértices $v \in V$.

Para modelarmos nosso problema original na forma de um grafo, será necessário ainda ressaltar algumas das características pertencentes ao problema estudado.

Observando uma instituição universitária, suas turmas têm caráter diferente do contexto escolar (ensino fundamental e ensino médio), em uma universidade as turmas são equivalentes uma disciplina que pode ser ofertada a um grupo de alunos, onde esses podem se inscrever nela ou não, de acordo com as normas estabelecidas pela instituição e vontade do aluno.

Um aluno, dessa forma, irá se inscrever em uma série de turmas todo semestre letivo. Cada turma terá ao início de cada semestre uma lista de alunos inscritos.

Para modelar essas turmas em um grafo de interseções, iremos primeiramente observar as listas de alunos inscritos nas turmas da instituição, como no exemplo mostrado a seguir na tabela 1.

Turma A	Turma B	Turma C	Turma D
001	003	001	020
002	006	003	021
003	009	006	022
004	011	009	023
005	012	011	024
006	013	012	025
007	014	013	026
008	015	018	027
009	016	019	028
010	017	020	029

Tabela 1 – Lista de turmas e a matrícula seus respectivos alunos

Observamos que essa instituição possui quatro turmas, com dez alunos em cada turma.

As turmas nesse exemplo são: “turma A”, “turma B”, “turma C ”e “turma D”. Nesse cenário iremos atribuir um vértice para cada turma, sendo o rótulo do vértice a letra que representa a respectiva turma (A, B, C e D).

Ao realizar esse processo teremos como resultado o grafo da figura 5, com quatro vértices, um para cada turma.

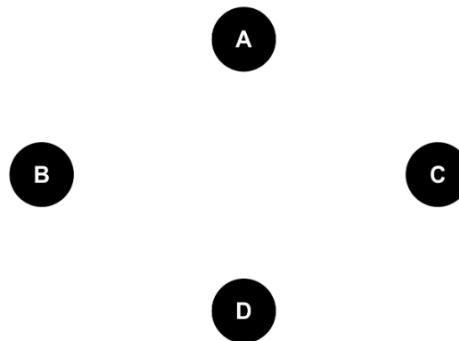


Figura 5 – Turmas modeladas como um Grafo

Foi definido em nossa modelagem que uma aresta representaria a interseção de alunos entre duas turmas. Com isso toda turma que possuir alunos em comum irá receber uma aresta (e) que interligue os vértices (v e w) referentes as respectivas turmas.

Observando novamente a tabela 1, ao compararmos os alunos inscritos nas turmas A e

B, notaremos que existem alguns alunos que pertencem a ambas turmas, como os alunos 003, 006 e 009.

Essa interseção observada irá gerar uma aresta no grafo que interliga os vértices **A** e **B**, como pode ser visto na figura 6. Representando assim que ambas turmas possuem alunos em comum.

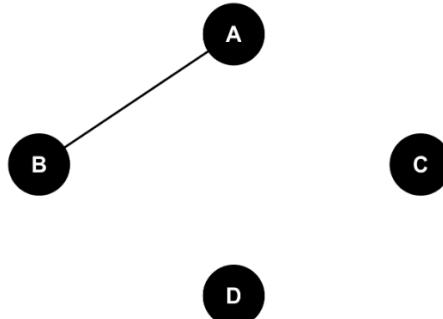


Figura 6 – Representação do Grafo após comparar as turmas A e B

Repetindo o processo de observar a existência de alunos em comum entre cada par de turmas, iremos formar respectivamente os grafos exibidos nas figuras 7, 8 e 9.

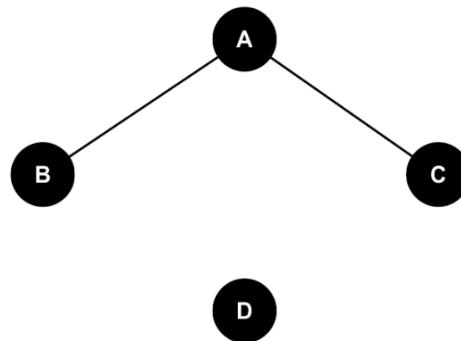


Figura 7 – Representação do Grafo após comparar as turmas A e C

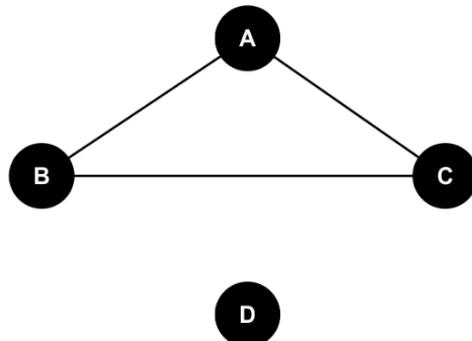


Figura 8 – Representação do Grafo após comparar as turmas B e C

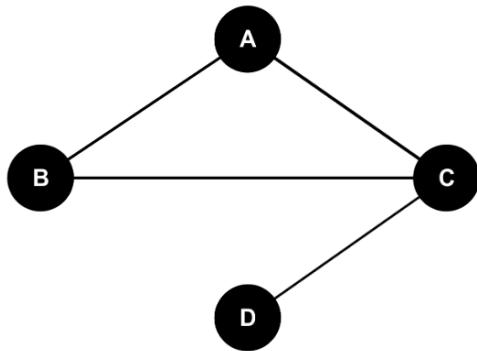


Figura 9 – Representação do Grafo após comparar as turmas C e D

Finalizando o processo de identificação de interseções entre as turmas teremos o grafo de interseção das turmas da instituição, e através dele (figura 9), podemos observar todas as turmas que possuem alunos em comum.

Agora que sabemos como transformar um conjunto de turmas em um grafo, precisaremos encontrar um meio de utilizar essa estrutura para simplificar a forma como criariam a grade de exames universitários.

A forma pela qual criaremos as grades de horários, será apresentada no capítulo de 4, onde será utilizado o grafo criado no exemplo desse subcapítulo (figura 9) para demonstrar os benefícios e limitações provenientes dessa modelagem e suas devidas soluções.

4 MÉTODO DE SOLUÇÃO

Após realizar a modelagem na forma de um problema de grafo, iremos utilizar essa estrutura para criar uma grade de horário de provas que satisfaça as restrições do problema.

Considerando um grafo G qualquer, definimos como as turmas pertencentes á G como $V(G) = \{v_1, v_2, v_3, \dots, v_n\}$, ou seja, o conjunto de vértices $V(G)$ corresponderá às disciplinas ofertadas na instituição, sendo n a quantidade de vértices do grafo $|V(G)|$ e de disciplinas ofertadas.

Definimos que os horários serão representados por números inteiros positivos $K = \{1, 2, 3, \dots, k\} \in \mathbb{N}$, sendo k o número correspondente a quantidade máxima de horários para realização das provas.

Ao atribuirmos uma turma a um horário de prova, estaremos relacionando um elemento de K á um vértice v de $V(G)$.

Sendo $V(G) = \{v_1, v_2, v_3, v_4\}$ e $K = \{1, 2, 3, 4\}$, podemos realizar uma atribuição que gere o resultado apresentado na tabela a seguir:

Vértices ($V(G)$)	Horários (K)
v_1	1
v_2	2
v_3	3
v_4	4

Tabela 2 – Relação entre os vértices e os horários

Dessa forma, teremos cada turma realizando a prova em um horário distinto. Como apresentado no capítulo 2, esse seria um exemplo de solução trivial para o problema.

Contudo, assim como explicado anteriormente, a solução trivial não é o foco do trabalho, dessa forma para se realizar a atribuição de turmas aos horários deveremos por uma ou mais turmas em um mesmo horário.

Para definir quais provas poderão ser aplicadas em um mesmo horário, iremos considerar os alunos pertencentes á ambas turmas. Com isso, caso duas turmas não possuam alunos em comum, elas poderão realizar seus exames em um mesmo horário.

Para sabermos se as turmas possuem alunos em comum, realizamos a modelagem das turmas na forma de um grafo (conforme apresentado no capítulo 3), dessa maneira podemos

visualizar a existência de relações de acordo com as arestas do grafo.

Agora, para realizarmos a atribuição das turmas em horários aproveitaremos a forma gráfica do grafo, atribuindo uma cor a cada horário.

Para demonstrar, iremos utilizar o grafo resultante da modelagem no capítulo 3 (figura 9), que utilizou como base os alunos e as turmas descritas na tabela 1.

Caso tivéssemos quatro ou mais horários disponíveis, bastaria realizar a solução trivial para o problema, então definiremos que será necessário aplicar as provas dessas turmas em apenas três horários. Sendo assim, teremos $k = 3$ horários e $n = 4$ turmas.

Para facilitar a visualização das operações que serão realizadas, relacionaremos os horários que temos disponíveis para realização de prova á cores únicas, conforme a tabela abaixo:

Horário	Cor
1	Vermelho
2	Azul
3	Verde

Tabela 3 – Relação entre horários e cores

A partir disso iremos atribuir á primeira turma o primeiro horário, dessa forma podemos pintar o vértice **A** (correspondente a turma A) de vermelho (cor que corresponde ao horário 1), assim como pode ser observado na figura 10.

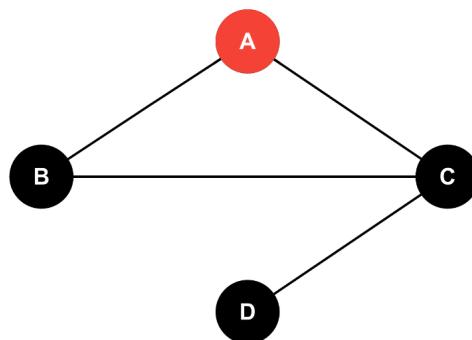


Figura 10 – Grafo após definir horário para turma A

Ao olhar para o vértice **B**, percebemos que a turma que ele representa possui alunos em comum com a turma representada pelo vértice **A**, logo, caso ambas as turmas necessitem fazer prova em um mesmo horário, esses alunos em comum terão que obrigatoriamente realizar a segunda chamada de uma das disciplinas.

Assim, visando evitar esse conflito de horários iremos atribuir a “turma B ”ao horário 2, com isso pintaremos o vértice **B** de Azul.

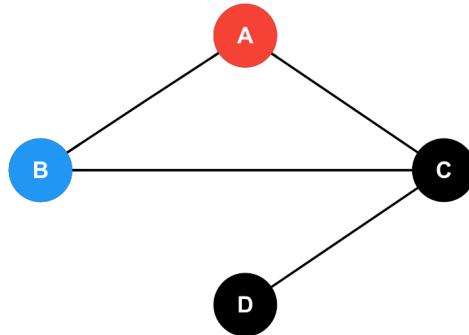


Figura 11 – Grafo após definir horário para turma B

Observando que o vértice **C** é adjacente (ligado por arestas) aos vértices **A** e **B**, sabemos que ele possui alunos em comum com ambas as turmas, dessa forma ele não deverá ser colocado em um horário diferente, assim atribuiremos ele ao horário 3 e pintaremos ele de verde.

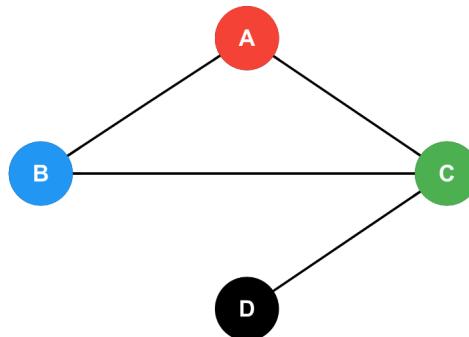


Figura 12 – Grafo após definir horário para turma C

O vértice **D**, é adjacente apenas ao vértice **C**, logo poderíamos pinta-lo de vermelho ou de azul, e dessa forma seus alunos realizariam a prova sem conflitos. Iremos então atribuí-lo ao horário 1 no mesmo horário que a turma A, e pintaremos o vértice D de vermelho.

Dessa forma podemos observar na figura 13 a distribuição das turmas em seus respectivos horários através do grafo com todos os seus vértices pintados.

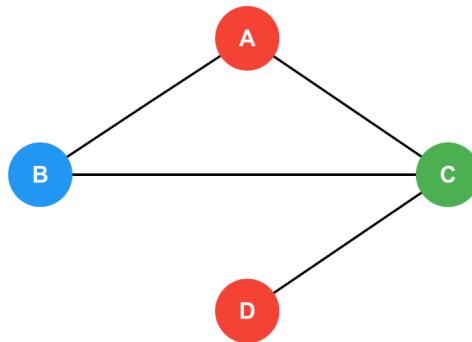


Figura 13 – Grafo após definir horário para turma D

Com isso obtemos como resultado, a grade de horários representada pela tabela a seguir:

Turmas	Horário
A, D	1
B	2
C	3

Tabela 4 – Alocação das turmas em três horários

O processo de pintar os vértices do grafo, de forma que vértices adjacentes não recebam mesma cor, consiste em uma área muito estudada dentro da Teoria de Grafos, que é a Coloração de Grafos.

No caso específico apresentado, pode ser observado que realizar a alocação das turmas em horários, corresponde à busca por uma coloração própria de vértices do grafo.

4.1 Coloração Própria de Vértices de um Grafo

A coloração de um grafo qualquer é dada através da atribuição de rótulos aos elementos pertencentes ao grafo (vértices ou arestas), relacionamos esses rótulos à cores de forma a facilitar a visualização da coloração. Contudo para uma coloração ser válida ela precisa cumprir determinada restrição, elementos adjacentes (ou incidentes) não podem possuir mesma “cor”.

Entre os tipos de coloração existem as de vértices, arestas e total, que visam colorir tanto vértices quanto arestas. Para Goldbarg e Goldbarg (2012) colorir os vértices de um grafo propriamente corresponde a atribuir cores aos vértices de forma que os vértices adjacentes recebam cores distintas.

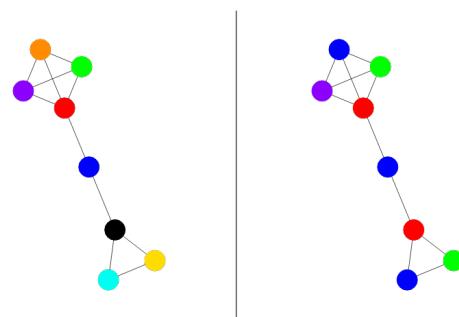


Figura 14 – Grafo colorido propriamente

Uma coloração própria para os vértices de um grafo pode ser realizada ao se colorir cada vértice de uma cor diferente, dessa forma nenhum vértice adjacente (ou vizinho) terá mesma cor. No entanto quando pretendemos colorir os vértices buscando utilizar um número mínimo de cores, nos deparamos com um problema que pode não ser muito simples de se resolver.

Definição 8. *O menor número possível de cores para se colorir um grafo é chamado de **número cromático do grafo** denotado por $\chi(G)$.*

Encontrar uma coloração com o número cromático do grafo é classificado como um problema NP-Completo quando $\chi(G) \geq 3$. Dessa forma, encontrar tal coloração é classificado como um dos problemas mais difíceis estudados na computação.

Sendo uma das áreas mais estudadas na matemática e computação atualmente, a coloração de grafos se popularizou através da demonstração para uma conjectura existente por mais de um século. O matemático Francis Guthrie conjecturou em 1852 que **quatro cores eram o suficiente para se colorir um mapa**. Por se tratar de um problema muito difícil a demonstração matemática foi apresentada apenas em 1960 com auxílio do poder computacional (LIMA, 2014).

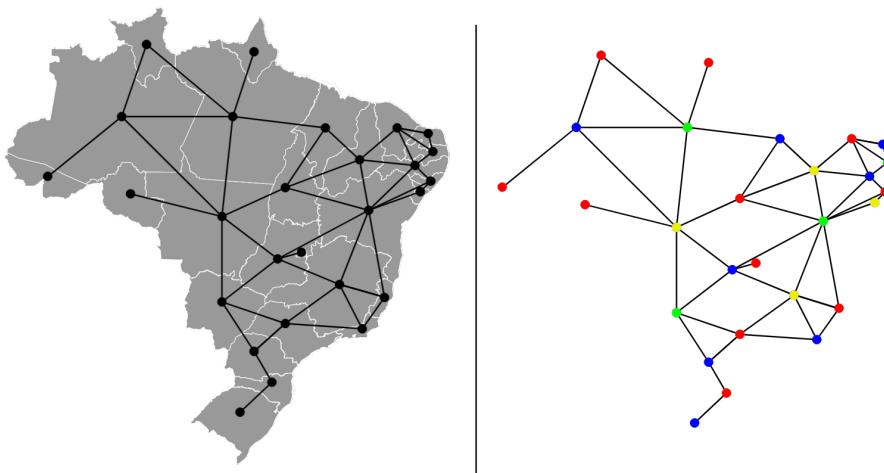


Figura 15 – Exemplo de 4-coloração do mapa do Brasil

Para essa demonstração foi utilizada uma estrutura de grafo, onde buscava-se modelar um mapa de forma que os vértices representassem locais e esses vértices possuiriam arestas caso possuíssem fronteira. Então ao se colorir propriamente o grafo era possível demonstrar que a mesma coloração poderia ser realizada no mapa, afim de que países que tenham fronteiras em comum não possuam mesma cor.

Desde então muitos outros modelos e aplicações utilizaram da coloração de grafos como uma forma de solução para um problema abstrato ou concreto.

Além da clássica aplicação concreta da coloração de grafos para se colorir mapas, existem ainda muitas outras formas de se utilizar essa ferramenta matemática.

Algumas aplicações em que se é utilizado a coloração de grafos, e que são formalmente apresentadas em diversos trabalhos e pesquisas acadêmicas, estão formas de se resolver jogos como *SUDOKU*, realização de transporte de componentes que podem ou não se misturados, visando minimizar custos e aumentar a segurança, armazenamento seguro ou inteligente de itens conforme suas características, visando deixá-los mais próximos ou distantes entre muitos outros observados na literatura.

Existem diversas propostas de algoritmos para encontrar uma coloração própria de um grafo, alguns se destacam por sua simplicidade e desempenho. Desde métodos exatos até heurísticas, as propostas de solução para o problema das colorações são as mais diversificadas possíveis e em alguns casos, a utilização dessas soluções não é o suficiente para se encontrar uma coloração que atenda as condições do problema modelado.

Nesse caso busca-se identificar os fatores que contribuem para a não obtenção do resultado ideal, e assim modelar o problema novamente solucionando então esses possíveis empecilhos.

4.2 Limitação no número de Horários de Prova

A solução do problema de exames universitários por meio da coloração própria de vértices, não resolverá todos os casos possíveis, já que é possível existir um caso onde a quantidade de horários disponíveis para realização da prova k é menor que o número cromático do grafo $\chi(G)$.

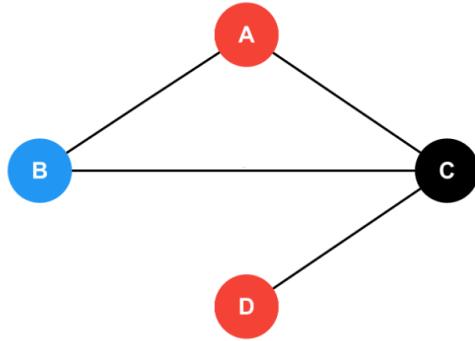


Figura 16 – Qual cor devo atribuir ao vértice C?

Nesse caso, como uma coloração própria não será suficiente, adotaremos uma nova forma de solução.

Considerando que não será possível separar todas as turmas que possuem alunos em comum em horários de forma que não gere conflitos, iremos buscar uma solução que minimize a quantidade de provas de segunda chamada, ou seja, a número de alunos prejudicados.

Para sabermos quantos alunos estarão sendo prejudicados precisaremos incluir mais uma etapa em nossa modelagem.

Iremos atribuir às arestas do grafo um peso, correspondente a quantidade de alunos que pertencem às duas turmas. Dessa forma teremos um Grafo Ponderado em Areias.

4.3 Grafos Ponderados em Areias

Retornando a tabela 1 no capítulo 3, observamos agora não apenas a existência de alunos em comum às turmas, mas também a quantidade de alunos.

Ao realizar esse processo e escrevermos sobre as arestas a quantidade de alunos na interseção, teremos um grafo ponderado como na figura 17.

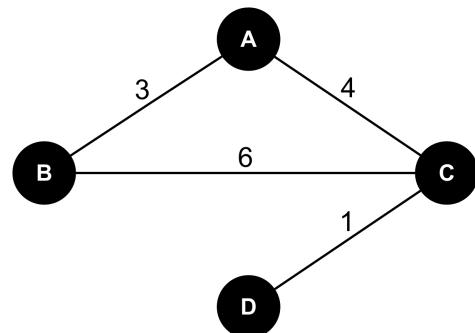


Figura 17 – Grafo ponderado em arestas

Utilizando do grafo ponderado conseguimos observar quantos alunos seriam prejudicados caso alocarmos quaisquer duas turmas em um mesmo horário.

Colorir o grafo agora consistirá em encontrar uma coloração não própria cuja soma dos pesos das arestas incidentes em vértices coloridos com mesma cor seja o menor possível.

Esse processo é chamado de Generalized Graph Coloring, onde efetuamos um tipo especial de coloração de modo a atingir algum objetivo pré-estabelecido.

4.4 Generalized Graph Coloring Problem

Definição 9. Vredeveld (2002), define o *k-GGCP* como sendo um problema que consiste em um grafo $G(V, E)$, uma função de peso $z : E \rightarrow \mathbb{Z}$ sobre as arestas, e um número inteiro $k \geq 2$.

Onde deseja-se encontrar uma atribuição de cores $c : V \rightarrow \{1, \dots, k\}$ dos vértices que minimize o peso total das arestas monocromáticas do grafo, isto é, arestas que incidem em vértices com mesma cor.

Usando o exemplo da figura 16, onde não sabemos como colorir o vértice **C**, pela ausência de uma terceira cor, iremos testar como seria se escolhermos uma das duas cores para colorir esse vértice.

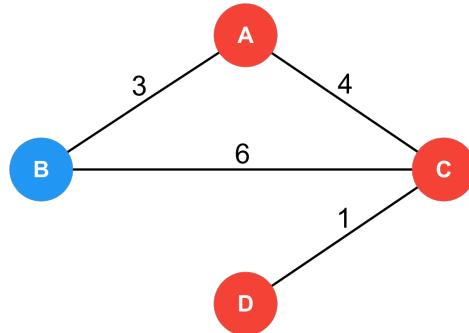


Figura 18 – Caso 1: Grafo colorido com vértice C vermelho

Caso optemos por colorir o vértice **C** de vermelho, teremos as turmas A, C e D em um mesmo horário de prova. Ao somar o peso das arestas que incidem nesses vértices, temos que cinco provas de segunda chamada serão necessárias, sendo quatro alunos prejudicados nas turmas A e C, e um aluno nas turmas C e D.

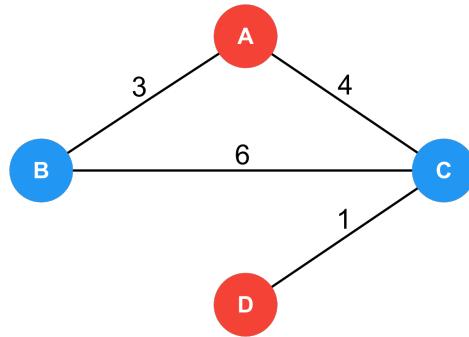


Figura 19 – Caso 2: Grafo colorido com vértice C azul

Escolhendo colorir o vértice de azul, temos a “turma C” no mesmo horário que a “turma B”, o que causaria conflito para os seis alunos que pertencem às duas turmas.

Ao buscar a distribuição que cause o menor número de conflitos chegaremos à coloração representada na figura 20, no qual podemos observar que teremos apenas os três alunos que fazem parte das turmas A e B prejudicados.

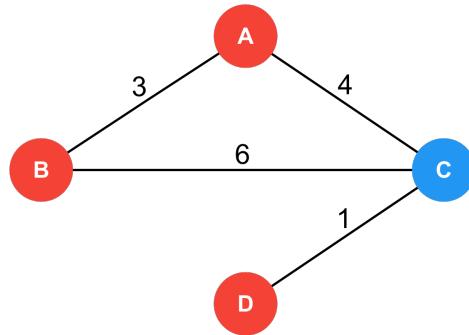


Figura 20 – Melhor coloração com duas cores para o Grafo

Dessa forma o resultado para a grade de horário de prova para as turmas do exemplo será:

Turma	Horário
A, B, C	1
C	2

Tabela 5 – Relação entre horários e cores

Causando dessa forma **três provas de segunda chamada** por consequência da alocação.

4.5 Complexidade do Problema e Viabilidade de Solução

O PPHC é um problema classificado NP-Completo por Garey e Johnson (1979), para todo $k \geq 3$, isso é, para qualquer quantidade de horários maior ou igual a 3 quando k não for fixo, Vredeveld (2002) define o problema como NP-Difícil quando o valor k é fixo, por ser equivalente ao MaxCut quando $k = 2$ e equivalente ao problema Max k -Cut para qualquer $k \geq 3$, enquanto estes são classificados como NP-Difícil.

Na teoria da computação a complexidade de um problema está diretamente atrelada a capacidade que temos de solucioná-la computacionalmente e o tempo que custará para isso. A maioria dos problemas podem ser divididos em duas diferentes classes P e NP.

São considerados P os problemas que possuem pelo menos um algoritmo capaz de resolve-lo em tempo polinomial, já um problema NP é um problema não determinístico, porém que pode ter uma solução verificada em tempo polinomial.

Um subconjunto dos NP é chamado de NP-Completo, sendo uma classe de problemas em que para se pertencer a ela, o problema deve ser classificado como NP e deve ser possível equipara-lo a outro problema já classificado como NP-Completo, ou seja, por definição os NP-Completos são uma classe que ao se encontrar uma solução para um dos problemas os demais também serão solucionados devido a suas semelhanças.

Outro subconjunto dos NP são os NP-Difíceis, que são classificados como problemas pelo menos tão difíceis quanto os NP-Completos.

Ao se deparar com situações desse tipo, o mais comum na literatura é buscar soluções não exatas. Assim, muitos trabalhos publicados para problemas similares ao apresentado aqui, demonstraram bons resultados utilizando diversas técnicas não exatas para se encontrar uma solução, com uso de heurísticas e meta-heurísticas como: *GRASP*, *Busca Tabu*, *Algoritmos Genéticos*, entre muitos outros (ROCHA, 2013).

4.6 Algoritmo Desenvolvido Inicialmente

Um algoritmo foi desenvolvido inicialmente, visando encontrar a melhor alocação de turmas em uma quantidade limitada de horários.

Esse algoritmo possui como entradas:

- (A) Um Grafo de interseção ponderado nas arestas G ;
- (B) a quantidade k de horários de prova disponíveis.

E como saída:

- (C) A coloração c (ou c^{min}) com menor custo obtido;

- (D) a quantidade P_c de provas de segunda chamada necessárias com a coloração de menor custo.

A entrada e a saída gerada pelo algoritmo consistia em arquivos de texto e o algoritmo apresentou bons resultados para uma série de testes que foram efetuados.

O método pelo qual o algoritmo buscava a melhor solução consistia em uma busca exaustiva por força bruta, na qual todas as combinações possíveis eram testadas e tinham seus resultados comparados, de modo a encontrar a solução com menor valor p de provas de segunda chamada.

Apesar da garantia da obtenção de uma solução ótima para o problema, o custo computacional (e tempo de processamento) eram muito elevados para grandes instâncias com muitos horários de prova. Visando encontrar uma solução mais viável para grandes instâncias foi proposta a criação de uma heurística que nos permitisse encontrar uma solução boa com menor custo computacional.

4.7 Heurística Proposta

Para desenvolver a heurística foi utilizado como base o bem conhecido algoritmo chamado LF (*Largest-first*), também chamado de método heurístico guloso de coloração própria de vértices de um grafo.

Esse algoritmo foi proposto inicialmente por Welsh e Powell (1967) e se baseia na estratégia de escolher colorir primeiramente os vértices que possuem maior grau, ordenando os vértices de acordo com o grau e os colorindo seguindo essa sequência.

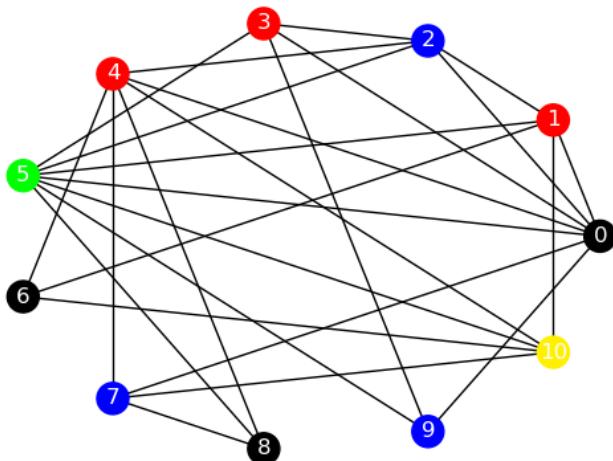


Figura 21 – Exemplo de Grafo colorido com algoritmo inicial

```

1 import networkx as nx
2
3 vertices = { ... }
4 cores = { ... }
5
6 def verificaVizinhos(no, cor, Grafo):
7     for vizinho in Grafo.neighbors(no):
8         corVizinho = Grafo.node[vizinho]['cor']
9         if corVizinho == cor:
10             return False
11     return True
12
13 def atribuiCor(no, Grafo):
14     for corCandidata in cores:
15         if verificaVizinhos(no, corCandidata, Grafo):
16             return corCandidata
17
18 if __name__ == "__main__":
19     Grafo = nx.Graph()
20
21     for no in vertices:
22         Grafo.add_node(no, cor=' ')
23         for i in range(0, len(vertices[no])):
24             Grafo.add_edge(no, vertices[no][i])
25
26     for no in Grafo.nodes:
27         Grafo.nodes[no]['cor'] = atribuiCor(no, Grafo)

```

Algoritmo 4.1 – Implementação do LF em Python

O algoritmo apresentado em 4.1 foi desenvolvido inicialmente buscando encontrar colorações onde a quantidade de horários de provas disponíveis eram maiores ou iguais ao número cromático do grafo $\chi(G)$. Porém, conforme o problema evoluiu para um cenário onde deseja-se encontrar uma solução que satisfaça também modelos onde a quantidade de horários disponíveis seja menor que o número cromático do grafo, foi necessário uma mudança na estratégia.

Ao implementar o método por busca exaustiva citado na seção 4.6, notou-se a necessi-

dade de se obter uma solução em um menor tempo de processamento, ao se observar que em modelos com muitas turmas e muitos horários esse tempo de execução poderia dificultar ou até impossibilitar a utilização da mesma.

Foi então proposto um algoritmo adaptado do LF, observado a seguir, onde seria priorizada a coloração com diferentes cores dos vértices adjacentes as arestas de maior peso do grafo, ao invés do vértice de maior grau, como no algoritmo clássico de Welsh e Powell (1967).

O algoritmo proposto pode ser exemplificado através dos passos apresentados através da tabela a seguir.

Passo	Ação
1	Buscar a aresta vw com maior peso do Grafo, cujo o vértice v ou w não possuam cor.
2	Verificar se o vértice v possui uma cor.
3	Selecionar uma cor i para colorir o vértice v .
4	Verificar se o vértice v pode ser colorido com a cor i .
5	Colorir o vértice v com a cor i .
6	Verificar se o vértice w possui uma cor.
7	Selecionar uma cor j para colorir o vértice w .
8	Verificar se o vértice w pode ser colorido com a cor j .
9	Colorir o vértice w com a cor j .
10	Repetir o passo 1 para a próxima aresta até que todos os vértices estejam coloridos.

Tabela 6 – Passos para serem executados pela heurística

A cada interação um novo par de vértices uv são selecionados e duas cores são escolhidas para colori-los, contudo em caso de um resultado positivo o ao executar o passo 2 o algoritmo deverá pular para o passo 6.

Caso não seja possível colorir um vértice com determinada cor, em função dele ser adjacente a outro vértice de mesma cor, o algoritmo deve retornar ao passo anterior e selecionar uma nova cor, se não houverem cores disponíveis o vértice será colorido com a cor que incide na aresta de menor peso.

A seguir está o código em linguagem C que tem como objetivo implementar os passos da heurística a fim de encontrar uma coloração ótima em um tempo reduzido de execução em comparação com o método de busca exaustiva.

```

1  int verificaVizinhos(int cor, int** G, int n, int v){
2      int i, soma = 0;
3
4      for (i=0; i<n; i++){
5          if (cor[i] == cor && i != v){
6              soma += G[i][v];
7          }
8      }
9      return soma;
10 }
11
12 int coloriVertice(int** G, int cores[], int v, int w,
13                         int k, int n){
14
15     int cor, soma, min_cor, min_soma = G[v][w];
16
17     if (cor[v]==-1){
18         for (cor=0; cor<k; cor++){
19             soma = verificaVizinhos(cor, G, n, v);
20             if (soma==0){
21                 cores[v] = cor;
22                 return 0;
23             } else {
24                 if (min_soma >= soma){
25                     min_cor = cor;
26                 }
27             }
28         }
29         cores[v] = min_cor;
30         return 1;
31     }
32     return -1;
33 }
34
35 int somaConflitos(int cores[], int** G, int n){
36     int soma = 0, i, j;
37
38     for (i=0; i<n-1; i++){

```

```

39         for (j=i+1;j<n;j++){
40             if (cores[i] == cores[j]){
41                 soma += G[i][j];
42             }
43         }
44     }
45     return soma;
46 }
47
48 int buscaAresta(int** G, int cores[], int k, int n){
49     int v, w, result_v, result_w, maior_aresta = 0;
50     int cont = 0;
51
52     for (i=0;i<n-1;i++){
53         for { j=i+1;j<n;j++){
54             if (G[i][j] > maior_aresta && cores[i]==-1
55                 || cores[j]==-1){
56                 maior_aresta = G[i][j];
57                 v = i;
58                 w = j;
59             }
60         }
61     }
62
63     result_v = coloriVertice(G, cores, v, w, k, n);
64     result_w = coloriVertice(G, cores, w, v, k, n);
65
66     for (i=0;i<n;i++){
67         if (cores[i]==-1){
68             cont++;
69         }
70     }
71     if (cont>0){
72         return buscaAresta(G, cores, k, n);
73     }
74     return somaConflitos(cores, G, n);
75 }
```

Algoritmo 4.2 – Heurística Proposta em C

Após desenvolver um modelo no qual pudéssemos representar as turmas e as relações de interseção de alunos entre elas, buscamos então demonstrar um algoritmo capaz de solucionar os desafios apresentados ao longo do capítulo de forma a gerar uma programação de horários de prova com mínimos conflitos de horários de prova para os alunos e consequente aplicação de avaliações de segunda chamada.

Contudo, para possibilitar uma utilização mais diversificada da solução. Foi proposto um sistema, que será apresentado no capítulo 5, no qual um usuário com menor conhecimento técnico fosse capaz de gerar grades de horários de prova.

5 SOFTWARE PARA AGENDAMENTO DE HORÁRIOS DE PROVAS

Para solucionar o problema apresentado no capítulo 2, este trabalho irá apresentar uma proposta de “software” que ofereça ao usuário a possibilidade de se agendar horários de prova, de forma simples e obtendo informações necessárias referentes a solução gerada.

Foi definido como objetivo do desenvolvimento, a criação de um “software” que atendesse as seguintes definições:

I – Simplicidade

Buscando ter um sistema que atenda a simplicidade definida por Pressman (2011, p. 341) como um conteúdo informativo, sucinto, apropriado para as informações entregues. É dito ainda que esse conteúdo deve apresentar uma estética agradável sem exageros, ter sua arquitetura centrada em atender os objetivos da aplicação da maneira mais simples possível e ter uma navegação de forma intuitivamente óbvia.

II – Compatibilidade

Pressman (2011, p. 342) define como compatibilidade a necessidade da aplicação de ser usada em uma variedade de ambientes distintos (equipamento físico diferentes, categorias de conexão com “internet”, sistemas operacionais, navegadores) e que a mesma deve ser compatível com esses diversos ambientes.

III – Navegabilidade

A navegação nas páginas deve ser mantida de forma simples, e previsível para Pressman (2011, p. 342), visando propiciar ao usuário um comportamento consistente. Os links devem ser sinalizados, e estarem posicionados de forma a facilitar o acesso aos recursos que esses forneçam.

IV – Reusabilidade

Para Pressman (2011, p. 362) deve-se pensar o sistema de forma a se permitir a reusabilidade, sendo importante por possibilitar a reutilização do programa, ou de parte dele em outras aplicações.

5.1 Tecnologias Utilizadas no Desenvolvimento

Pressman (2011, p. 35) define como um sistema “web”, uma categoria de “software” centralizada em redes, sendo em sua forma mais simples um conjunto de arquivos de hipertexto interconectados.

Um sistema “web” pode ser dividido em duas partes, chamadas de “Frontend” e “Backend”. Souto (2019) classifica “frontend” como sendo a parte visual de um sistema, com a qual um usuário pode interagir, e classifica “backend” como a parte do sistema responsável por realizar a ponte entre os dados no navegador e o banco de dados, aplicando regras de negócio e validações, para garantir o acesso às funcionalidades do sistema.

As tecnologias usadas para desenvolver o “Frontend” do protótipo foram:

HTML 5

HTML é uma abreviação de “Hypertext Markup Language” - Linguagem de Marcação de Hipertexto. Resumindo em uma frase: o HTML é uma linguagem para publicação de conteúdo (texto, imagem, vídeo, áudio e etc) na Web (FERREIRA; EIS, 2020).

A partir da versão 5 muitos novos recursos foram desenvolvidos e disponibilizados, possibilitando maior utilização de elementos visuais na página com uso de apenas HTML.

CSS 3

O CSS (“Cascading Style Sheet” ou Folhas de Estilo em Cascata em português) é definido por GonÇalves (2019) como uma linguagem de marcação para estilizar páginas HTML.

Em sua versão 3 o CSS se tornou ainda mais compatível com os navegadores modernos e em conjunto do HTML5 possibilitam a criação de elementos que antes seriam necessárias a inclusão de outras linguagens para realização.

JavaScript

O JavaScript é definido por Junior (2014) como uma linguagem de “Scripts” que reside dentro de um documento em HTML, promovendo diferentes níveis de interatividade entre o usuário e a página Web, através da Lógica de Programação.

Cytoscape.JS

É uma biblioteca que utiliza a Teoria dos Grafos escrita em JavaScript de código aberto que permite a visualização, análise de estruturas de Grafos, com elementos animados.

Bootstrap

Bootstrap é uma biblioteca de “frontend” livre e de código aberto para a criação de sites e aplicações “web”. Ele contém HTML e modelos de “design” baseados em CSS para tipografia, formas, botões, navegação e outros componentes da “interface”, bem como extensões JavaScript opcionais. O Bootstrap destina-se a facilitar o desenvolvimento de sites dinâmicos e aplicações “web” (GSTI, 2020).

Com a utilização do Boostrap é possível ter benefícios para o desenvolvedor quanto para o usuário, para o desenvolvedor existirá um padrão que agilizará o desenvolvimento do

“software”, para o usuário as páginas possuirão elementos que lembram elementos que ele já utilizou (visualizou) em outras páginas “Web”.

As tecnologias usadas para desenvolver o “Backend” do protótipo foram:

PHP 7.0

PHP (PHP: “Hypertext Processor”) é uma linguagem de “scripts” de código aberto, que se caracteriza por ser embutido em páginas HTML e executado no lado do servidor, sendo usada principalmente para realização de conexões com banco de dados.

MariaDB 10

É um Sistema de Gerenciamento de Banco de Dados (SGBD), baseado no MySQL, possuindo todos os comandos, “interfaces”, bibliotecas e APIs existentes no MySQL.

Apache Server

É um “software” de servidor “web” de código aberto desenvolvido pela *Apache Software Foundation*, que possibilita a comunicação entre um servidor físico e as máquinas dos usuários para realização de transmissão de informações pela “internet”.

5.2 Especificações do Software

5.2.1 Requisitos do Software

Sommerville (2011, p. 58-59) define como requisitos de um sistema: as descrições do que o sistema deve fazer, os serviços que oferece e as restrições de funcionamento.

Os requisitos funcionais de “software”, são os serviços que o sistema deve fornecer e como o sistema deve se comportar em cada situação, de acordo com a definição de Sommerville (2011, p. 58-59), apresentados a seguir:

[RF001] – Ler Turmas

Descrição O “software” deve ler turmas e seus respectivos alunos (entrada).

[RF002] – Ler Horários

Descrição O “software” deve receber a quantidade de horários disponíveis para prova (entrada).

[RF003] – Gerar Relatório

Descrição O “software” deve emitir um relatório com os horários de prova por turma (saída).

[RF004] – Gerar Quantidade de Provas de Segunda Chamada

Descrição O “software” deve informar a quantidade de provas de segunda chamada que necessárias (saída).

[RF005] – Gerar Lista de Alunos em Segunda Chamada

Descrição Caso existam conflitos de horários o “software” deve informar os alunos e as turmas que precisarão de segunda chamada (saída).

[RF006] – Cadastrar Usuários

Descrição O “software” deve cadastrar diferentes usuários.

[RF007] – Criar Espaços de Trabalho

Descrição Os usuários devem poder criar, acessar e modificar seus espaços de trabalho.

[RF008] – Importar Arquivos

Descrição O “software” deve permitir a importação e exportação de turmas através de arquivos XLSX (planilhas MS Excel).

[RF009] – Gerenciar Espaços de Trabalho

Descrição O “software” deve permitir separar turmas em diferentes espaços de trabalho.

[RF010] – Autenticar Usuário

Descrição O “software” deve salvar as informações do usuário mediante a “login” e senha.

[RF011] – Gerenciar Turmas

Descrição O “software” deve permitir a edição dos dados dos alunos que compõem uma turma.

5.2.2 Regras de Negócio

A Regra de Negócio é definida como “a forma de fazer o negócio, refletindo a política interna, o processo definido e/ou as regras básicas de conduta” (DEXTRA, 2013), serve para estabelecer padrões mínimos de qualidade com base em princípios decididos previamente dos quais as funcionalidades e recursos do sistema devem seguir.

[RN01] – Estar logado

Descrição Para utilizar o sistema o usuário precisa ser cadastrado e estar logado.

[RN02] – Turmas devem ter nomes únicos

Descrição Só pode existir uma turma com um determinado nome, caso seja adicionado uma turma com nome igual a uma turma já existente essa inclusão será ignorada.

[RN03] – Matricula e Nome do Aluno

Descrição Uma matrícula está associada diretamente a um aluno, logo não podem existir dois alunos diferentes com a mesma matrícula, assim como não podem existir duas matrículas para um mesmo aluno.

[RN04] – Disponibilidade de Horário Mínima

Descrição Uma tabela de horário de prova deve ter no mínimo dois horários disponíveis para realizar os exames.

5.2.3 Casos de Uso

Casos de Uso são definidos por Fowler (2007, p. 104) como uma técnica para captar os requisitos funcionais de um sistema, servindo para descrever relações e interações entre usuários e recursos, além de fornecer uma narrativa sobre como o sistema será utilizado.

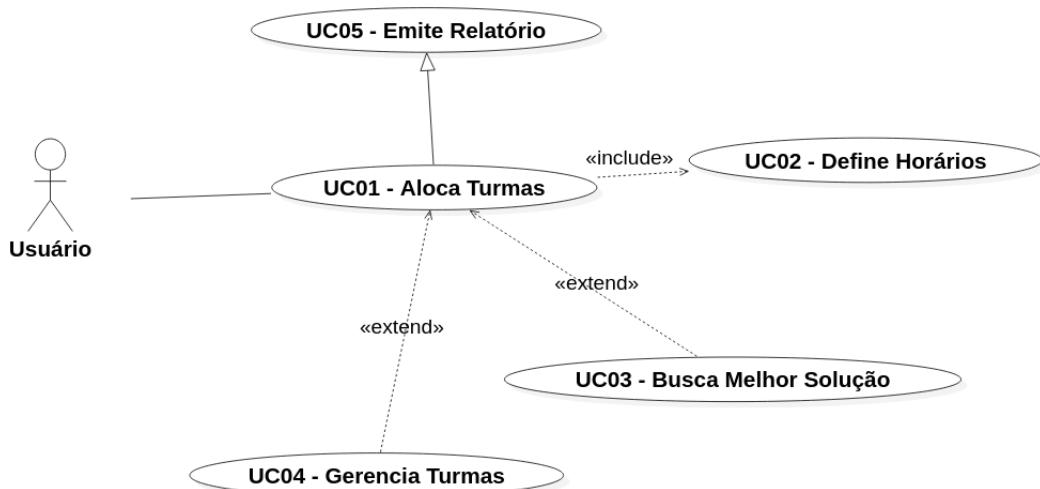


Figura 22 – Diagrama de Casos de Uso

5.2.3.1 Descrição dos Casos de Uso

Descrição dos Atores

* Usuário

Descrição

Toda pessoa ou “software” que irá interagir com o sistema através de sua “interface” gráfica.

Casos de Uso

* [UC01] – Aloca Turmas

Descrição

Este caso de uso permite que o usuário possa definir horários específicos de prova para alocar as turmas.

Pré-condições

1. O usuário deve estar logado.
2. O usuário deve ter turmas e seus respectivos alunos cadastrados no sistema.
3. O usuário deve definir a quantidade de horários disponíveis para realização das provas.

Pós-condições

1. O sistema deve atribuir cada turma á um único horário.
2. O sistema deve informar a quantidade total de conflitos de alunos em duas turmas em um mesmo horário.

Tabela 7 – Fluxo Principal de Interações de [UC01]

Sequência	Usuário	Sistema
1	Seleciona uma turma e move para uma caixa de horário.	
2		Atribui a turma á um horário.
3		Calcula o número de alunos que tenham duas provas em um mesmo de horário.

* [UC02] – Define Horários

Descrição

Este caso de uso realiza a leitura da quantidade de horários de prova que o usuário dispõe para a realização das provas.

Pré-condições

1. O usuário deve estar logado.

Pós-condições

1. O sistema deve registrar como quantidade de horários um número maior ou igual a dois.
2. O sistema deve permitir a execução do caso de uso [UC01].

Tabela 8 – Sequência de Interações de [UC02]

Sequência	Usuário	Sistema
1	Informa a quantidade de horários de prova disponível.	
2		Atualiza o número de caixas de horários.

* **[UC03] – Busca Melhor Solução**

Descrição

Este caso de uso permite ao sistema a executar um algoritmo de modo a buscar automaticamente uma alocação com o menor custo possível para as turmas selecionadas.

Pré-condições

1. O caso de uso [UC01] deve ter sido executado.
2. O usuário precisa indicar todas as turmas que deverão realizar as provas nos horários indicados.

Pós-condições

1. O sistema deve encontrar a alocação que resulte no menor número de provas de segunda chamada forçadas pela alocação.
2. O sistema deve encaminhar automaticamente o usuário para o caso de uso [UC05].

Tabela 9 – Fluxo Principal de Interações de [UC03]

Sequência	Usuário	Sistema
1	Seleciona a opção de buscar a melhor solução.	
2		Executa algoritmo para buscar solução ideal.
3		Redireciona para tela de Relatório com a solução ideal.

* **[UC04] – Gerencia Turmas**

Descrição

Este caso de uso permite que o usuário crie, edite, remova, importe e exporte turmas, e seus respectivos alunos dentro do sistema.

Pré-condições

1. O usuário deve estar logado.

Pós-condições

1. O sistema deve atualizar o banco de dados com as novas informações assim que alteradas pelo usuário.
2. O sistema deve permitir alterações de forma simplificada pelo usuário.
3. O sistema deve exibir mensagens de alerta para o usuário em caso de alterações que não possam ser desfeitas.

Tabela 10 – Fluxo de Interações de [UC04] – Criação de Turma

Sequência	Usuário	Sistema
1	Seleciona a opção de “Adicionar Turma” na página de gerenciamento de turmas	
2		Redireciona a página atual para a página de “Criação de Turmas”
3	Informa o nome da turma que deseja criar	
4	Seleciona a opção para inserir o aluno	
5		Cria a opção de inclusão de matrícula e nome para o novo aluno
6	Informa a matrícula e o nome do aluno	
7	Seleciona a opção “Criar Turma”	
8		Grava turma no banco de dados
9		Vincula alunos á turma criada
10		Redireciona a página atual para a página de “Gerenciamento de Turmas”

Tabela 11 – Fluxo de Interações de [UC04] – Visualização de Turma

Sequência	Usuário	Sistema
1	Seleciona a opção de “Visualizar” de uma turma	
2		Redireciona a página atual para a página de “Visualização de Turma”
3		Exibe uma lista em ordem alfabética com os alunos da turma

Tabela 12 – Fluxo de Interações de [UC04] – Edição de Turma

Sequência	Usuário	Sistema
1	Seleciona a opção de “Editar” uma turma	
2		Redireciona a página atual para a página de “Edição de Turma”
3		Exibe uma lista em ordem alfabética com os alunos da turma
4		Habilita os campos de matrícula e nome para edição
5	Altera o nome e a matricula de um aluno	
6	Seleciona a opção “Salvar Alterações”	
7		Atualiza a lista de alunos pertencentes à turma no banco de dados
8		Redireciona a página atual para página de “Gerenciamento de Turmas”

Tabela 13 – Fluxo de Interações de [UC04] – Exclusão de Turma

Sequência	Usuário	Sistema
1	Seleciona a opção de “Apagar” uma turma	
2		Exibe um alerta
3	Seleciona a opção “Sim” no alerta	
4		Apaga a turma do banco de dados
5		Atualiza a página de “Gerenciamento de Turmas”

* **[UC05] – Emite Relatório**

Descrição

Este caso de uso permite que o usuário visualize de forma gráfica os resultados e informações referentes a alocação feita, como a quantidade de alunos prejudicados pela alocação, e a lista de alunos que foram prejudicados.

Pré-condições

1. O caso de uso [UC01] ou o caso de uso [UC02] devem ter sido executados.

Pós-condições

1. O sistema deve informar a quantidade de conflitos gerados pela alocação.
2. O sistema deve informar os alunos prejudicados com duas provas alocadas em um mesmo horário.
3. O sistema deve permitir o “download” e/ou impressão do relatório através de arquivo PDF.

Tabela 14 – Fluxo Principal de Interações de [UC05]

Sequência	Usuário	Sistema
1	Conclui a alocação das turmas em seus respectivos horários para realizarem as provas.	
2		Exibe a quantidade de provas de segunda chamada necessárias com a alocação.
3		Exibe a alocação feita.
4		Fornece a opção de gerar PDF do relatório.
5		Fornece a opção de realizar nova alocação.

5.2.4 Diagrama de Atividades

O diagrama de atividades auxilia na criação e representação de um fluxo de utilização de um sistema por um usuário.

O diagrama 23 apresenta o comportamento do sistema a medida em que um usuário acessa suas funcionalidades, iniciando em sua primeira conexão ao sistema até a impressão do relatório contendo o resultado obtido com a alocação realizada.

Os diagramas de atividades são uma técnica para descrever lógica de procedimento, processo de negócio e fluxo de trabalho. De várias formas, eles desempenham um papel semelhante aos fluxogramas, mas a principal diferença entre eles e a notação de fluxograma é que os diagramas suportam comportamento paralelo. (FOWLER, 2007, p. 118)

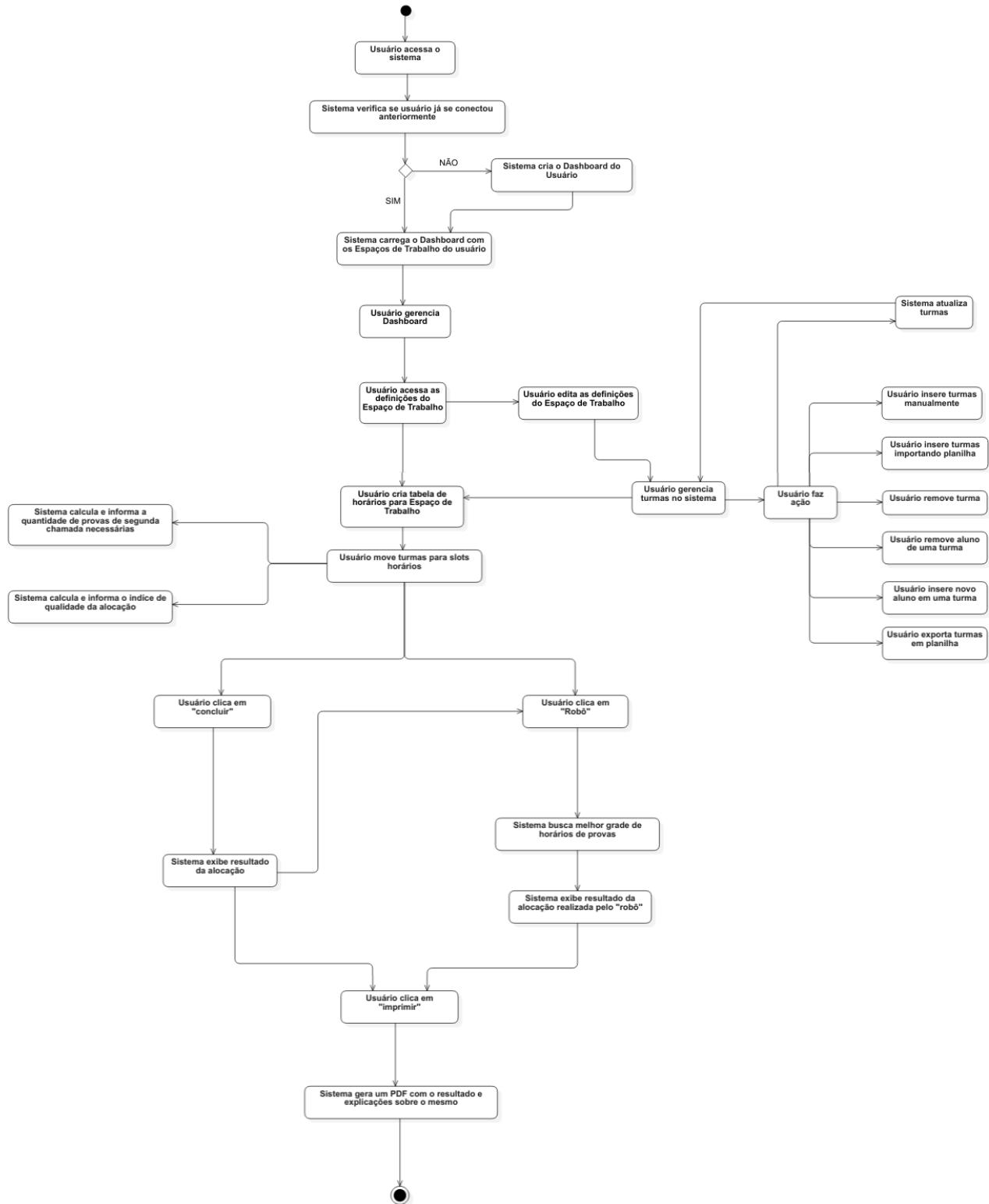


Figura 23 – Diagrama de Atividades

5.2.5 Diagrama de Classes

O diagrama de classes é um dos diagramas mais utilizados em UML, ele apresenta a forma pela qual o código está estruturado, permitindo dessa forma, que outros desenvolvedores possam compreender o funcionamento do “software”, facilitando dessa maneira a organização do código e sua manutenção ao longo do tempo.

Um diagrama de classes descreve os objetos presentes no sistema e os vários relacionamentos estáticos existentes entre eles. Os diagramas de classes também mostram as propriedades e as operações de uma classe e as restrições que se aplicam à maneira como os objetos estão conectados. A UML utiliza a palavra característica como um termo geral que cobre as propriedades e operações de uma classe. (FOWLER, 2007, p. 52)

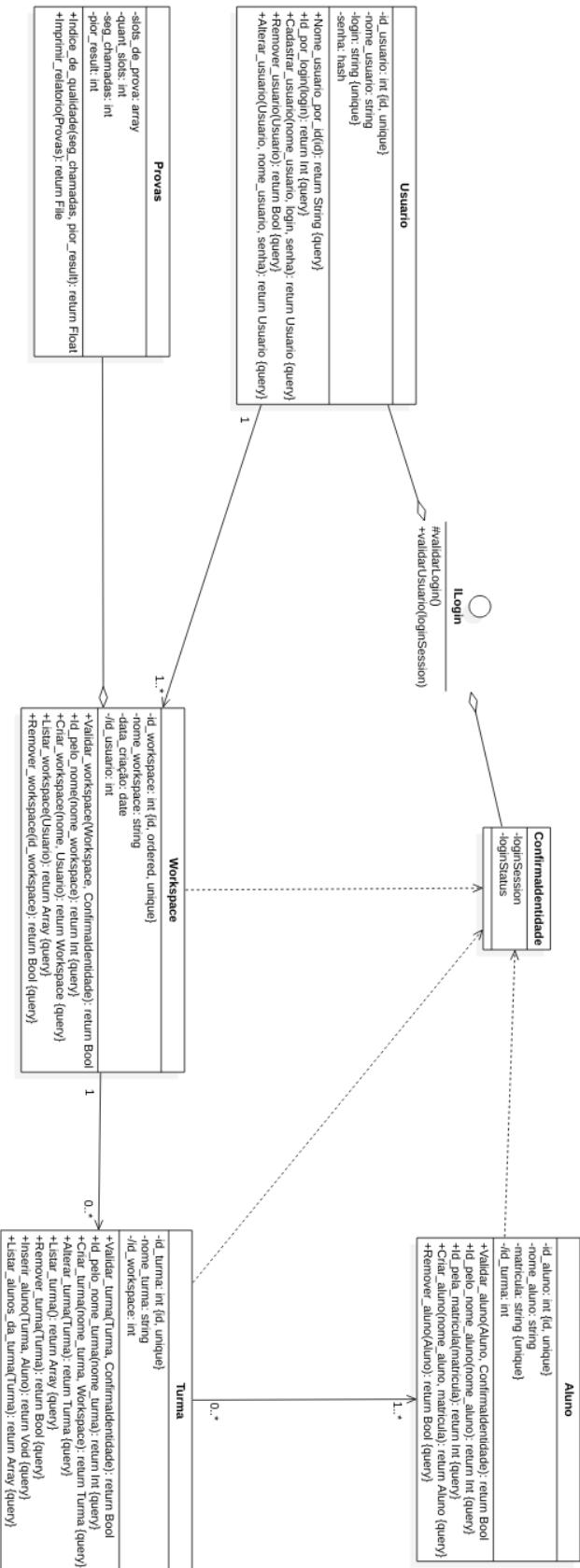


Figura 24 – Diagrama de Classes

5.2.6 Diagrama de Entidade-Relacionamento

O diagrama de entidade-relacionamento é utilizado para permitir a representação geométrica das relações entre diferentes entendidas abstratas, podendo essas serem pessoas, cargos, objetos, imóveis e quaisquer outros conjuntos que possuam atributos em comum que os permita serem diferenciados.

O diagrama neste trabalho apresenta as relações entre turmas e seus respectivos alunos, as turmas que compõem um espaço de trabalho (*workspace*), que representarão um conjunto de turmas, por exemplo uma instituição ou turno, e dessa forma a relação entre um usuário e todos os seus espaços de trabalho.

A técnica de modelagem de dados mais difundida e utilizada é a abordagem entidade-relacionamento (ER). Nesta técnica, o modelo de dados é representado através de um modelo entidade-relacionamento (ER). Usualmente, um modelo ER é representado graficamente, através de um diagrama entidade-relacionamento (DER). A abordagem ER foi criada em 1976 por Peter Chen. Ela pode ser considerada um padrão de fato para modelagem conceitual. (HEU-SER, 2009, p. 12)

5.2.7 Relatórios

O relatório gerado pelo sistema tem como propósito ser um documento para o usuário com as informações referentes a forma como as turmas foram distribuídas, possíveis conflitos causados pela alocação, e explicações sobre a construção e funcionamento do sistema.

Seu foco é: ser instrutivo, de modo a permitir que o usuário consiga replicar o resultado em uma universidade real. Ser didático, visando explicar as bases do problema matemático por trás do problema de horários de exames universitários e a proposta de solução usada no trabalho.

A partir do arquivo PDF gerado pelo sistema, é possível efetuar o “download” do relatório e salva-lo digitalmente, e também permite a impressão do mesmo, caso seja desejável ter uma versão física da solução.

Dessa forma o relatório serve tanto como uma instrução de como a grade de prova poderá ser realizada, como um documento com propósito explicativo que justifique a alocação e que antecipe os problemas que por ventura venham a acontecer, no caso os alunos com mais de uma prova em um mesmo horário.

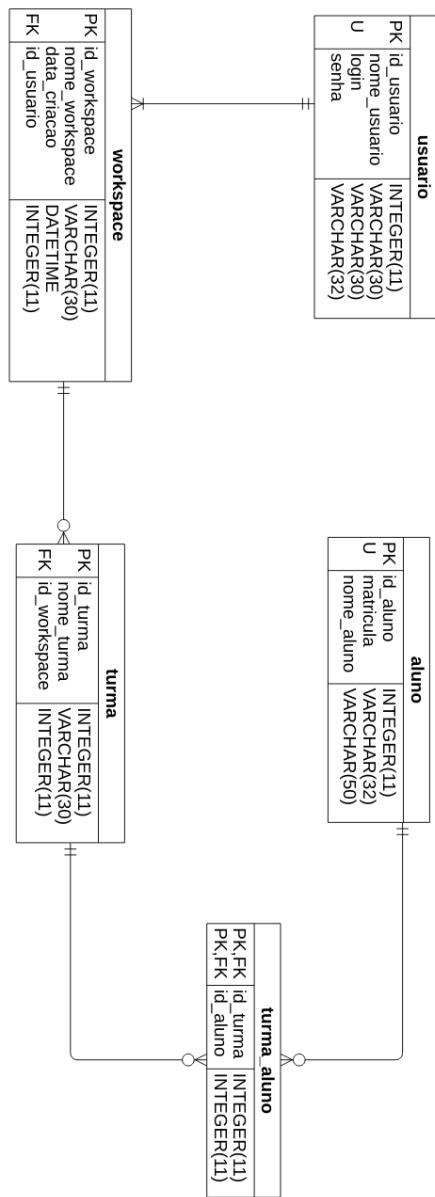


Figura 25 – Diagrama de Entidade Relacionamento

6 RESULTADOS OBTIDOS E TRABALHOS FUTUROS

Este trabalho visou obter o melhor resultado possível considerando as turmas, seus alunos e a quantidade de horários disponíveis na instituição para realização das provas. Dessa forma foi apresentado tanto conceitualmente quanto de forma prática, nos resultados da aplicação proposta, a melhor alocação e suas possíveis consequências. Permitindo dessa forma, que ajustes por parte do usuário sejam realizadas sobre as provas de segunda chamada naquele cenário em específico.

Apresentamos formalmente neste trabalho o problema de programação de horários de exames universitários com suas respectivas classificações e restrições, demonstramos também a modelagem matemática para o problema e propomos um algoritmo capaz de obter uma solução adequada para as condições estabelecidas, além de ofertar essa solução na forma de um sistema que permite a busca pela resolução de forma automática e amigável para qualquer usuário.

Com o desenvolvimento de uma solução computacional, foi possível automatizar o processo de busca por soluções, facilitando o trabalho das pessoas e principalmente, permitindo-as obter uma melhor solução com uma expressiva redução no tempo de busca, em comparação com métodos manuais.

Visando o desenvolvimento de um produto amigável para o usuário final (*user-friendly*), foi um objetivo desse trabalho tornar transparente a utilização de abstrações, estruturas e modelos matemáticos, atendendo aos fundamentos de simplicidade, compatibilidade, naveabilidade e reusabilidade a fim de permitir o emprego futuro deste trabalho em pesquisas de natureza similar.

Em nossa pesquisa utilizamos a abordagem quantitativa, onde realizaríamos um teste de usabilidade a fim de validar nosso protótipo desenvolvido. Usaríamos como instrumento de coleta de dados um questionário, com 25 perguntas. Na elaboração das perguntas, procuramos investigar o conhecimento dos usuários sobre sua área de atuação profissional, a fim de detectar possíveis usuários finais, o conhecimento prévio do usuário sobre o tema, sua facilidade em identificar as funcionalidades do sistema e sua consequente utilização.

O instrumento não foi aplicado aos usuários na forma de um teste de usabilidade no ano de 2020, devido a limitações causadas devido as restrições distanciamento social impostas em meio a pandemia de Covid-19 causada pelo vírus Sars-Cov2, popularmente conhecido como Coronavírus. Os usuário responderiam ao questionário em forma de entrevista enquanto realizariam o teste do sistema de forma guiada, na qual deveriam seguir instruções recebidas durante a utilização do sistema.

Contudo, o roteiro desenvolvido para a aplicação dos testes de usabilidade está disponível em apênce do trabalho e poderá ser aplicado juntamente as adequações observadas em

Trabalhos Futuros.

6.1 Trabalhos Futuros

Segundo pesquisa realizada pelo Cetic.br (2019), 99% da população brasileira que possuía acesso à “internet” tinha um aparelho celular em 2019, sendo que, para 58% da população com acesso à “internet”, o aparelho celular era o único meio para se conectar a rede. Considerando informações desse gênero, fica evidente à necessidade de se criar futuramente adequações no protótipo visando atingir um público mais amplo.

Para que seja possível deixar o protótipo totalmente funcional em “smartphones”, será necessário realizar algumas adaptações na “interface” da aplicação, aumentando sua responsividade e adicionando mais ações para os eventos disparados ao pressionar sobre a tela do dispositivo, que são diferentes dos *cliques* com o cursor do mouse. Dessa forma, tornar o protótipo inicial em um produto multiplataforma, sem perder seu propósito e funcionamento.

Visando obter um melhor desempenho, é interessante a realização de uma pesquisa para implementar outros métodos Heurísticos propostos na literatura, tanto para o problema de programação de horários de exames universitários, quanto expandi-los para problemas de escalonamento de horários de aulas com devidas adaptações. Dessa forma pretendemos comparar os diferentes resultados e desempenhos obtidos com cada proposta, visando ao final, utilizar desse conhecimento para, quem sabe, propor uma nova heurística específica que resolva o problema apresentado neste trabalho.

Após a criação e implementação dessa heurística pretendemos a comparar seus resultados com demais heurísticas, caso ela se demonstre promissora, será um desejo utilizar instâncias de dados realistas para realizar testes. Para isso uma base de dados postulante seriam os dados do *International Competition Timetabling* ou ICT, que se trata de uma competição internacional para avaliar algoritmos heurísticos (em grande parte) no processo de resolução do problema clássico de *Timetabling*, que é alocar professores, turmas e salas em horários específicos dadas uma série de restrições utilizando uma base real de dados de uma determinada instituição universitária.

Apesar de o problema estudado no trabalho e o problema de alocação de professores em salas de aula, juntamente as suas turmas e horários serem similares em natureza, será necessário alterar sutilmente algumas informações, e acrescentar uma informação a base de dados, será necessário inicialmente retirar da modelagem os professores e suas restrições, as salas de aula e suas restrições, entre outras, e incluir o número máximo de horários de prova que aquele conjunto de turmas poderá fazer, para dessa forma ser possível utilizar uma base de dados real para submeter as heurísticas a testes e de acordo com seus resultados, buscar aplicá-las em problemas parecidos.

7 CONSIDERAÇÕES FINAIS

Ao produzir um sistema amigável utilizando a coloração de grafos de interseção para resolver o problema de programação de horário de exames em universidades, obtivemos sucesso em facilitar a obtenção do resultado de uma forma simplificada e organizada.

O trabalho trouxe uma abordagem em que se obteve o melhor resultado possível considerando as turmas, seus alunos e a quantidade de horários de provas disponíveis na instituição. Sendo apresentado conceitualmente e de forma prática na aplicação proposta, a melhor alocação e suas possíveis consequências, permitindo dessa forma, que fossem realizadas as aplicações das provas de segunda chamada da melhor forma possível.

Mesmo possuindo vasta quantidade e qualidade de referências e propostas de algoritmos eficientes que objetivassem solucionar o problema, não foi observado em nenhuma das pesquisas citadas uma aplicação voltada para o usuário final, considerando a possibilidade da utilização do “software” por um leigo em matemática ou computação.

No decorrer do trabalho, foi apresentado formalmente o problema de programação de horários de exames universitários com suas respectivas classificações e restrições, demonstrada a modelagem matemática para o problema e propondo um algoritmo capaz de obter o melhor resultado, além de apresentado um sistema que permite a realização da busca pela resolução de forma automática, contudo se mantendo amigável para qualquer usuário.

A documentação do processo, partindo da identificação de um problema de natureza combinatória até o desenvolvimento de um produto amigável para um usuário final, por meio da utilização de abstrações, estruturas e modelos matemáticos, não é comum em trabalhos de conclusão de curso da instituição, contudo, visando uma maior disseminação dessas informações no âmbito da instituição e possibilitar a (re)utilização futura de partes deste trabalho em pesquisas de problemas de natureza similar, optamos seguir por este caminho.

Buscando desenvolver um “software” fosse capaz de resolver o problema, mantendo sua simplicidade e sendo intuitivo ao máximo para os usuários finais, foi necessário que o problema estivesse bem estruturado, possibilitando a solução encontrada por meio da aplicação fosse completamente satisfatória tanto do ponto de vista de usabilidade, quanto do ponto de vista conceitual e matemático.

7.1 Limitações da Pesquisa

No decorrer da pesquisa, uma série de referências foram consultadas a fim de fundamentar de forma expressiva os métodos, ferramentas e expressões usadas no trabalho. No entanto devido ao excedente de informações relativas a diversos pontos, fez-se necessário que o trabalho

adotasse uma postura mais didática e por vezes superficial sobre alguns temas.

Devido a opção de se realizar um trabalho mais didático e prático, considerando ser uma pesquisa de graduação tecnóloga de Análise de Sistemas, a parte matemática do trabalho carece de maiores detalhes e aprofundamento. Contudo as referências citadas ao longo do trabalho possuem pesquisas com grande enfoque matemático, sendo de certa forma complementar ao apresentado.

A modelagem UML do sistema carece de uma documentação mais extensa ao se comparar com as recomendações de autores do tema, contudo a escolha por realizar uma documentação mais simples foca-se na flexibilização para possibilitar a reutilização da modelagem em futuras implementações. Contudo a opção deixa a documentação mais distante do sistema a ser desenvolvido de fato.

Na parte da análise dos resultados obtidos e observados com a utilização do protótipo através do teste de usabilidade não foi realizado, devido as restrições impostas pelo distanciamento social causado pela pandemia de Covid-19 causada pelo vírus Sars-Cov2, popularmente conhecido como Coronavírus, contudo o registro do questionário e roteiro permitirá a aplicação do experimento e a consequente validação do protótipo em oportunidades futuras.

7.2 Recomendações para Pesquisas Futuras

Os problemas de programação de horários são vastos na literatura e permitem a resolução de problemas de naturezas diferentes, assim como os problemas similares citados, de alocação de bandas em dias de um festival, escalas de equipes em uma companhia aérea e a distribuição de recursos militares. Existem ainda muitos outros problemas não enunciados que podem ser modelados da mesma forma e por fim serem solucionados pelos mesmos métodos.

Estudos de problemas similares podem focar tanto em explorar as abordagens da solução do modelo matemático, através de métodos determinísticos ou não-determinísticos, ou na criação de sistemas personalizados voltados para resolução de uma determinada demanda.

Trabalhos nas áreas da Teoria dos Grafos e de resolução de problemas de natureza combinatória por muitas vezes focam-se apenas na modelagem e solução matemática ou algorítmica do modelo, e muitas vezes não produz um sistema voltado para um usuário real, é recomendada a implementação de sistemas que apliquem essas as soluções, hoje abstratas, para muitos problemas.

Observa-se também que existe a possibilidade de se expandir as variáveis consideradas no problema, de forma que sejam ponderadas questões como quantidade máxima de alunos a realizarem provas em um mesmo horário, quantidade de salas disponíveis simultaneamente para realização dos exames, quantidade de professores ou aplicadores disponíveis em cada horário, minimização da quantidade de provas de segunda chamada para um mesmo aluno, entre muitos outros que poderiam ser inclusos.

Ao se prosseguir com pesquisas e desenvolvimento, no âmbito principalmente da graduação, de ferramentas voltadas para temas similares ao apresentado nesse trabalho, cada vez mais serão populares os métodos de modelagem e resolução de problemas de natureza combinatória, que estão entre os mais complexos da computação e talvez os que resolvam o maior número de problemas cotidianos considerados complicados. Também será aumentada a quantidade de aplicações automatizadas para problemas cotidianos, beneficiando dessa forma toda a sociedade, além proporcionar aumento considerável do acervo de conhecimento científico em língua portuguesa sobre esses problemas.

REFERÊNCIAS

- 4CIS SISTEMAS LTDA. *Cronos - Programa para fazer horários escolares*. 2020. Disponível em: <<https://www.cronostimetable.com/>>. Citado na página 18.
- BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. [S.l.]: Elsevier Rio de Janeiro, 2007. v. 2.
- BIAJOLI, Fabrício Lacerda; LORENA, Nogueira. **Novas Heurísticas para o problema de geração de escalas de jogos para torneios esportivos**. 2007.
- BONDY, J.; MURTY, U. **Graph Theory With Application**. 1978. Citado na página 17.
- BORGES, Suzan Kelly. Resolução de timetabling utilizando algoritmos genéticos e evolução cooperativa. **Universidade Federal do Paraná**, 2003.
- BROWN, Jason. **The complexity of generalized graph colorings**. 1996. 257-270 p. Citado na página 17.
- BROWN, Jason; CORNEIL, Derek. **On generalized graph coloring**. 1987. 87 - 99 p. Citado na página 16.
- BRUCKER, Peter. **Scheduling Algorithms**. Springer, 2007.
- BURKE, Edmund et al. **Hybrid variable neighbourhood approaches to university exam timetabling**. 2010. 46-53 p. Citado na página 16.
- BURKE, Edmund; RUDOVÁ, Hana. **Practice and Theory of Automated Timetabling VI: 6th International Conference, PATAT 2006 Brno, Czech Republic, August 30-September 1, 2006 Revised Selected Papers**. [S.l.]: Springer, 2007. v. 3867.
- BURKE, Edmund K et al. **Decomposition, reformulation, and diving in university course timetabling**. **Computers & Operations Research**, Elsevier, v. 37, n. 3, p. 582–597, 2010.
- BURKE, E. K.; NEWALL, J. P.; WEARE, R. F. **A Memetic Algorithm for University Exam Timetabling**. 1995. Descrição do Timetabling.
- BURKE, Edmund Kieran; PETROVIC, Sanja. **Recent research directions in automated timetabling**. **European Journal of Operational Research**, Elsevier, v. 140, n. 2, p. 266–280, 2002.

BURKE, E. K. et al. **Examination Timetabling in British Universities - A Survey**. [S.l.]: Spring, 1995.

CACCHIANI, Valentina; GALLI, Laura; TOTH, Paolo. **A tutorial on train timetabling and train platforming problems**. [S.l.]: Università di Pisa, 2013.

CARLSON, R.; NEMHAUSER, George. **Scheduling to Minimize Interaction Cost**. 1966. 52-58 p.

Citado na página 17.

CERIOLI, Márcia R; VIANA, Petrucio. **Minicurso de Combinatória de Contagem. II COLOQUIO DE MATEMÁTICA DA REGIAO SUL**, v. 2, 2012.

CETIC.BR, Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação. **Pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros - TIC Domicílios 2019**. [S.l.], 2019. Disponível em: <<https://cetic.br/pt/tics/domicilios/2019/individuos/C16A/>>.

Citado na página 63.

CORMEN, Thomas et al. **Algoritmos: Teoria e Prática**. [S.l.: s.n.], 2001.

CORMEN, Thomas H. **Desmistificando Algoritmos**. 1. ed. [S.l.]: Elsevier, 2014.

CSIMA, J.; GOTLIEB, C. C. **Tests on a Computer Method for Constructing School Timetables**. Commun. ACM, Association for Computing Machinery, New York, NY, USA, v. 7, n. 3, p. 160–163, mar. 1964. ISSN 0001-0782. Disponível em: <<https://doi.org/10.1145/363958.363986>>.

DEXTRA. **Requisito ou Regra de Negócio?** 2013. Disponível em: <<https://dextra.com.br/pt/requisito-ou-regra-de-negocio/>>.

Citado na página 49.

DIVERIO, Tiarajú A; MENEZES, Paulo B. **Teoria da Computação-UFRGS: Máquinas Universais e Computabilidade**. [S.l.]: Bookman Editora, 2009.

EISELT, Horst A; SANDBLOM, Carl-Louis. **Decision analysis, location models, and scheduling problems**. [S.l.]: Springer Science & Business Media, 2013.

ELMAGHRABY, Salah Eldin. **Symposium on the Theory of Scheduling and its Applications**. [S.l.]: Springer Science & Business Media, 2012. v. 86.

FEOFILOFF, Paulo. **Algoritmos para Grafos em C via Sedgewick**. Disponível em: <https://www.ime.usp.br/~pf/algoritmos_para_grafos/>. Acesso em: 26/01/2018.

_____. **Minicurso de Análise de Algoritmos.** 2011.

FEOFILOFF, Paulo; KOHAYAKAWA, Yoshiharu; WAKABAYASHI, Yoshiko. **Uma Introdução Sucinta à Teoria dos Grafos.** 2011. Disponível em: <<https://www.ime.usp.br/~pf/teoriadosgrafos/texto/TeoriaDosGrafos.pdf>>.

FERREIRA, Elcio; EIS, Diego. **HTML5: Curso W3C Escritório Brasil.** 2020. Disponível em: <<https://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf>>.

Citado na página 47.

FOWLER, Martin. **UML Essencial.** 3. ed. [S.l.]: Bookman, 2007.

Citado 3 vezes nas páginas 50, 56 e 58.

GAREY, Michael R.; JOHNSON, David S. **Computers and Intractability: A Guide to the Theory of NP-Completeness.** 1. ed. New York, NY, USA: W. H. Freeman & Co., 1979. ISBN 0716710447.

Citado na página 39.

GEHA SISTEMAS ESPECIALISTAS. **URÂNIA.** 2020. Disponível em: <<http://horario.com.br/>>.

Citado na página 18.

GOLDBARG, Marco Cesar; GOLDBARG, Elizabeth Ferreira Gouvêa. **Grafos: Conceitos, algoritmos e aplicações.** [S.l.]: Elsevier Editora Ltda, 2012. v. 1.

Citado na página 33.

GOLDBARG, Marco Cesar; LUNA, Henrique Pacca Loureiro; GOLDBARG, Elizabeth Ferreira Gouvêa. **Programação Linear e Fluxos em Redes.** [S.l.]: Elsevier Editora Ltda, 2015. v. 1.

GONÇALVES, Ariane. **O que é CSS? Guia Básico para Iniciantes.** 2019. Disponível em: <<https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/>>.

Citado na página 47.

GOTLIEB, C. C. **The construction of class-teacher timetables.** 1963. 73-77 p.

Citado na página 19.

GRUPO WPENSAR. **GridClass - Programa de Horário Escolar Online.** 2020. Disponível em: <<http://gridclass.com.br/>>.

Citado na página 18.

GSTI, Portal. **O que é Bootstrap?** 2020. Disponível em: <<https://www.portalgsti.com.br/bootstrap/sobre/>>.

Citado na página 47.

HEUSER, C.A. **Projeto de banco de dados : Volume 4 da Série Livros didáticos informática UFRGS.** Bookman, 2009. (Livros didáticos informática UFRGS). ISBN 9788577803828. Disponível em: <<https://books.google.com.br/books?id=pv0cPwAACAAJ>>.

Citado na página 60.

JUNIOR, Vilson Heck. **Programação para a Internet**. 2014. Disponível em: <http://docente.ifsc.edu.br/vilson.junior/pi/04_Introducao_JavaScript.pdf>.

Citado na página 47.

KANN, Viggo et al. **On the Hardness of Approximating MAX k-CUT and its Dual**. 1997.

Citado na página 17.

KIAER, Lynn; YELLEN, Jay. **Weighted graphs and university course timetable**. 1992. 59-67 p.

Citado na página 16.

KLEIN, S. **Algoritmos e Complexidade de Decomposição em Grafos**. 1994. Tese (Doutorado) — Doctoral Thesis (in portuguese), Progr. de Eng. de Sistemas e Computação . . . , 1994.

KNUTH, Donald Ervin; BRUIJN, NG DE. **Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms**. [S.l.]: American Mathematical Soc., 1997. v. 10.

KOSTUCH, Philipp; SOCHA, Krzysztof. Hardness prediction for the university course timetabling problem. In: SPRINGER. **European conference on evolutionary computation in combinatorial optimization**. [S.l.], 2004. p. 135–144.

LAI, Lien-Fu et al. **An Artificial Intelligence Approach to Course Timetabling**. 2008. 223-240 p.

Citado na página 16.

LARMAN, Craig. **Utilizando UML e Padrões**. 1. ed. [S.l.]: Bookman, 2000.

LEUNG, Joseph Y-T. **Handbook of Scheduling: Algorithms, Models, and Performance Analysis**. [S.l.: s.n.], 2004.

Citado na página 17.

LEVITIN, Anany. **Introduction to the design & analysis of algorithms**. [S.l.]: Boston: Pearson, 2012.

LIMA, Alex Reimann Cunha. **Classificação de cografos quanto ao índice cromático**. Universidade Federal da Fronteira Sul, 2014.

Citado na página 34.

MARTIN, Robert C. **Código limpo: habilidades práticas do Agile software**. [S.l.]: Alta Books, 2019.

MENEZES, Dimitri Carvalho. **Formulações exata e heurística para o problema de programação de horários da Universidade Federal de Sergipe**. DCOMP-Departamento de Computação–Ciência da Computação–São Cristóvão–Presencial, 2017.

MONTEIRO, Renan Costalonga; KAMPKE, Edmar Hell. **Heurísticas de Construção no Problema de Tabela-Horário de Universidades.** 2017.

MÜHLENTHALER, Moritz. **Fairness in academic course timetabling.** In: **Fairness in Academic Course Timetabling.** [S.l.]: Springer, 2015. p. 75–105.

NEUFELD, GA; TARTAR, John. **Graph coloring conditions for the existence of solutions to the timetable problem.** *Communications of the ACM*, ACM New York, NY, USA, v. 17, n. 8, p. 450–453, 1974.

OLIVEIRA, Marcos Paulo Martins et al. Timetabling, a complexidade na geração de horários em instituições de ensino. **RE3C-Revista Eletrônica Científica de Ciência da Computação**, v. 6, n. 1, 2011.

PAGE-JONES, Meilir. **Fundamentos do desenho orientado a objeto com UML.** 1. ed. [S.l.]: Makron Books, 2001.

PEREIRA, Renato Santos; LACRUZ, Adonai José. **Inserção de parâmetros controladores da aleatoriedade no método GRASP aplicado a um problema de programação de horários em escolas.** *Revista Gestão da Produção Operações e Sistemas*, v. 12, n. 3, p. 265, 2017.

PINEDO, Michael. **Planning and scheduling in manufacturing and services.** [S.l.]: Springer, 2005.

_____. **Scheduling.** [S.l.]: Springer, 2012. v. 29.

PINTO, José Wilson Coura. **Grafos ORTH[h, s, t].** 2018. Tese (Tese (Doutorado)) — UFRJ/-COPPE, 2018.

Citado 2 vezes nas páginas 23 e 26.

POULSEN, Camilo José Bornia. **Desenvolvimento de um Modelo para o School Timetabling Problem Baseado na Meta-Heurística Simulated Annealing.** 2012. Tese (Mestrado) — UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL, 2012.

POULSEN, Camilo José Bornia; BANDEIRA, Denise Lindstrom. **Desenvolvimento de um Modelo Matemático para Apoio à Decisão na Programação de Grades Horárias de Escolas de Excelência Pedagógica.** 2013.

PRESSMAN, Roger S. **Engenharia de Software, uma abordagem profissional.** 7. ed. [S.l.]: AMGH Editora, 2011.

Citado na página 46.

REIS, Joaquim. **Uma introdução ao scheduling.** 2006.

RICHARD, W CONWAY; WILLIAM, L MAXWELL; LOUIS, W MILLER. **Theory of Scheduling.** [S.l.]: Addison-Wesley, Reading, Massachusetts, 1967.

ROCHA, Wallace de Souza. **Algoritmo GRASP para o Problema de Tabela-horário de Universidades.** 2013. Tese (Mestrado) — Universidade Federal do Espírito Santo, 2013.
Citado 2 vezes nas páginas 19 e 39.

RODRIGUES, Antonio Gabriel; GÓMEZ, Arthur Tórgo. **Tratamento de um problema de escalonamento considerando datas de entrega, turnos de produção e trocas de ferramentas via Busca Tabu.** Production, SciELO Brasil, v. 18, n. 1, p. 64–75, 2008.

ROSS, P.; CORNE, D.; TERASHIMA-MARIN, H. **The Phase-Transition Niche for Evolutionary Algorithms in Timetabling.** 1995.

SANTOS, Haroldo Gambini; SOUZA, Marcone Jamilson Freitas. **Programação de Horários em Instituições Educacionais: Formulações e Algoritmos.** 2007.
Citado na página 19.

SCHAERF, A. **A Survey of Automated Timetabling.** 1999. 87-127 p.
Citado 2 vezes nas páginas 16 e 20.

SCHEEREN, Vanessa; PEREIRA, Elizangela; VAZ, Francieli. **PROGRAMAÇÃO DE HORÁRIOS DE EXAMES ATRAVÉS DA COLORAÇÃO DE GRAFOS.** 2018. 30-46 p.
Citado na página 16.

SCHMIDT, Gunther; STRÖHLEIN, Thomas. Timetable construction—an annotated bibliography. **The Computer Journal**, Oxford University Press, v. 23, n. 4, p. 307–316, 1980.

SILVA, Douglas José da; SILVA, Geiza Cristina da. **Heurísticas baseadas no algoritmo de coloração de grafos para o problema de alocação de salas em uma instituição de ensino superior.** Anais do XLII SBPO, p. 2839–2849, 2010.

SILVA, Italo Luis da; MACHADO, Pedro Henrique Mázala; CAMPOS, Saulo Cunha. **Algoritmo genético para resolução do problema de university course timetabling para instituições de ensino superior privadas: Um estudo de caso na FAGOC.** 2017.

SIPSER, Michael. **Introdução à teoria da computação.** [S.l.]: Thomson Learning, 2007.

SOCHA, Krzysztof; SAMPELS, Michael; MANFRIN, Max. **Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art.** 2003.
Citado na página 16.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. [S.l.]: Pearson Education, 2011.
Citado na página 48.

SOUTO, Mario. **O que é front-end e back-end?** 2019. Disponível em: <<https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end>>.

Citado na página 47.

SZWARCFITER, Jayme Luiz. **Teoria Computacional de Grafos**. first. [S.l.]: Elsevier Editora Ltda., 2018.

Citado na página 23.

SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estrutura de Dados e Seus Algoritmos**. 2. ed. [S.l.]: LTC, 1994.

SZWARCFITER, Jayme Luiz. **Grafos e Algoritmos Computacionais**. 2. ed. [S.l.]: Campus, 1986. 35-42, 169-195 p.

Citado na página 23.

VREDEVELD, Tjark. **Combinatorial Approximation Algorithms: Guaranteed versus experimental performance**. 2002. Tese (PhD) — Technische Universiteit Eindhoven, 2002.
Citado 3 vezes nas páginas 16, 37 e 39.

WAKABAYASHI, Yoshiko. **Notas de Aula de Teoria de Grafos**. 2012. Disponível em: <<https://www.ime.usp.br/~tassio/arquivo/2012-ii/grafoes/notas-grafoes.pdf>>. Acesso em: 05/05/2018.

WELSH, D. J. A.; POWELL, M. B. **An upper bound for the chromatic number of a graph and its application to timetabling problems**. 1967. 85-86 p. Disponível em: <<https://doi.org/10.1093/comjnl/10.1.85>>.

Citado 2 vezes nas páginas 40 e 42.

WERNER, Frank; BURTSEVA, Larysa; SOTSKOV, Yuri N. **Special Issue on Algorithms for Scheduling Problems**. [S.l.]: Multidisciplinary Digital Publishing Institute, 2018.

WERRA, D. de. **Some Combinatorial Models for Course Scheduling**. 1995.

WREN, A. **Scheduling, Timetabling and Rostering – A Special Relationship?** 1996.
Citado na página 19.

Apêndices

APÊNDICE A – ROTEIRO

A seguir está o roteiro para realização do teste de usabilidade do sistema desenvolvido, e de forma conjunta as perguntas referentes ao questionário de percepção do usuário.

I – Apresentação

Agradeço a participação no teste de usabilidade do protótipo do sistema de agendamento de provas.

II – Identificação do Usuário

Antes de iniciarmos é necessário informar os seguintes dados:

- Qual seu nome?
- Qual seu “e-mail”?
- Qual profissão ou ocupação?
- Já utilizou ou viu esse sistema anteriormente?
- Conhece o trabalho ao qual o sistema pertence?

Esse dados serão utilizados para permitir quantificar sua experiência com a usabilidade do sistema.

III – Identificação do Sistema

Você estará sendo orientado(a), conforme as ação você deverá tomar ao utilizar o sistema, e simultaneamente serão realizadas perguntas sobre sua experiência.

Neste momento do teste o usuário deverá ser conduzido até à página onde se encontra o sistema, e deverá ser realizado login utilizando um usuário predefinido que só é possuidor do espaço de trabalho padrão: “exemplo”.

Observe a página atual do sistema por 15 segundos e responda:

- Qual a finalidade dessa página para você?
- Você consegue dizer quais os recursos estão disponíveis nessa página?

Agora identifique onde está o menu principal do sistema e visualize suas informações.

Agora crie um novo **Espaço de Trabalho** com o título “Minha Escola”.

Acesse o **Espaço de Trabalho** “Minha Escola”.

Retorne até a página “**Dashboard**”, por favor.

Apague o **Espaço de Trabalho** “Minha Escola”.

Responda a seguir:

- Você teve alguma dificuldade em seguir alguma instrução?

IV – Espaço de Trabalho

Acesse as **Definições do Espaço de Trabalho** “exemplo”.

Observe por 15 segundos esta página e responda a seguir:

- Qual a finalidade dessa página para você?
- Você consegue dizer quais os recursos estão disponíveis nessa página?

V – Gerenciamento das Turmas

Acesse a página para **Gerenciar as Turmas** do “exemplo”.

Observe a página por 15 segundos e responda:

- Qual a finalidade dessa página para você?
- Você consegue dizer quais os recursos estão disponíveis nessa página?

Visualize os alunos de alguma das turmas.

Insira um novo aluno na turma.

Remova esse aluno da turma.

Retorne as **Definições do Espaço de Trabalho** “exemplo”.

- Você teve alguma dificuldade em seguir alguma instrução até esse momento?

VI – Criar Tabela de Horários

Crie uma **Tabela de Horário** para “exemplo”.

Visualize as opções no menu secundário da página.

Observe a página por 15 segundos e responda a seguir:

- Qual a finalidade dessa página para você?
- Você consegue dizer quais os recursos estão disponíveis nessa página?

Aumente a quantidade de horários de prova.

Defina o horário em que a primeira turma deverá realizar prova.

Defina o horário para as demais turmas de “exemplo” realizarem prova.

Observe a página por 10 segundos e responda a seguir:

- Você consegue dizer quais os recursos estão disponíveis nessa página?

Gere um **Relatório** com os horários de prova que você determinou.

VII – Relatório

Observe a página por 15 segundos e responda a seguir:

- Qual a finalidade dessa página para você?
- Você consegue dizer quais os recursos estão disponíveis nessa página?

Visualize os alunos que deverão realizar duas provas diferentes em um mesmo horário.

Utilize o **Robô** para encontrar uma nova tabela de horários de prova.

VIII – Robô

Observe a página por 15 segundos e responda a seguir:

- Existe alguma diferença para a página anterior?

Salve o relatório do resultado como um **PDF**.

IX – Avaliação Geral

Responda as perguntas a seguir para finalizar o teste.

- Você utilizaria esse sistema para gerar grades de horários de prova?
- Entre 1 e 5, qual seria sua avaliação sobre sua utilização do sistema?
- Você recomendaria esse sistema para uma pessoa que precisasse criar grades de horários de prova?
- Você conhece ou já utilizou algum sistema similar a esse?
- Se você precisasse definir o sistema em uma palavra, qual seria?

Muito obrigado por participar deste teste de usabilidade.

Anexos

ANEXO A – PROTÓTIPO

A.1 Instalação

A.1.1 Pré-requisitos do Projeto

Programas necessários para realizar a utilização do *software*:

- **Apache Server.**
- **PHP 7.0 ou superior.**
- **MariaDB 10 ou superior.**
- **GIT.**

Após verificar a existência dos programas citados e suas respectivas versões, pode ser iniciado o processo de instalação.

Para realizar a instalação deve-se entrar no diretório indicado pelo servidor **Apache** no seu sistema operacional através terminal (ou *console*), e após isso executar os comandos de acordo com o tutorial contido no repositório do protótipo <<http://github.com/FilipeRodrigues3003/TCC/agendador>>. Com isso você irá copiar os arquivos do protótipo para o diretório do seu servidor local.

A.2 Utilização

Após a instalação pode-se acessar o software a partir de <[//localhost/agendador](http://localhost/agendador)> para acessar o protótipo em caso de instalação seu próprio computador, ou através de <<http://ipdoseuservidor/agendador>> caso tenha feito a instalação em um servidor remoto.

Ao acessar pela primeira vez será exibida a página de **Login** conforme figura 26 abaixo, caso o usuário ainda não possua um *login*, este deverá clicar sobre o botão **Cadastre-se**.

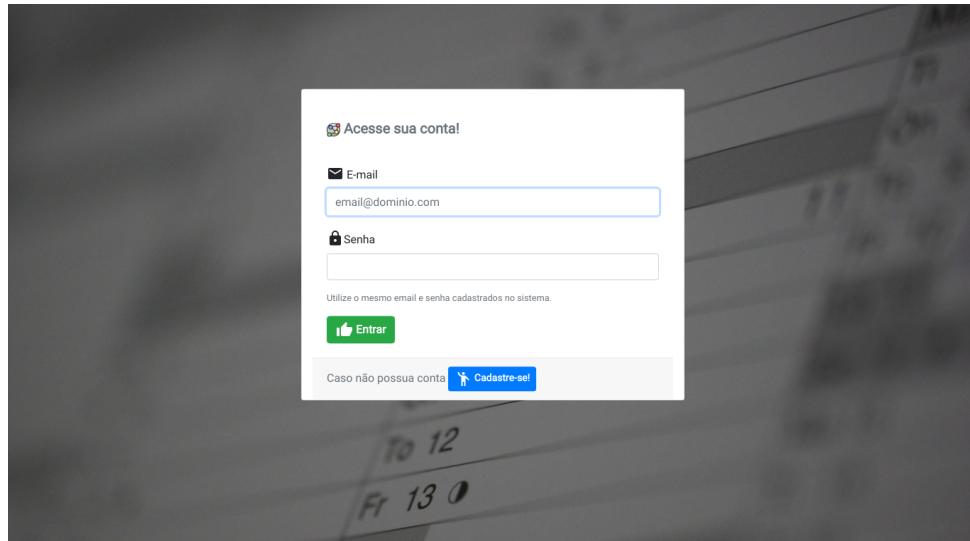


Figura 26 – Tela de Login

A.2.1 Cadastro de Usuários

Para que seja possível utilizar o sistema, é necessário que o usuário possua um *login* no mesmo, e para isso o usuário deverá realizar um pequeno cadastro na página apresentada na figura 27.

O usuário deverá informar obrigatoriamente: seu nome completo, endereço de *e-mail* válido e uma senha.

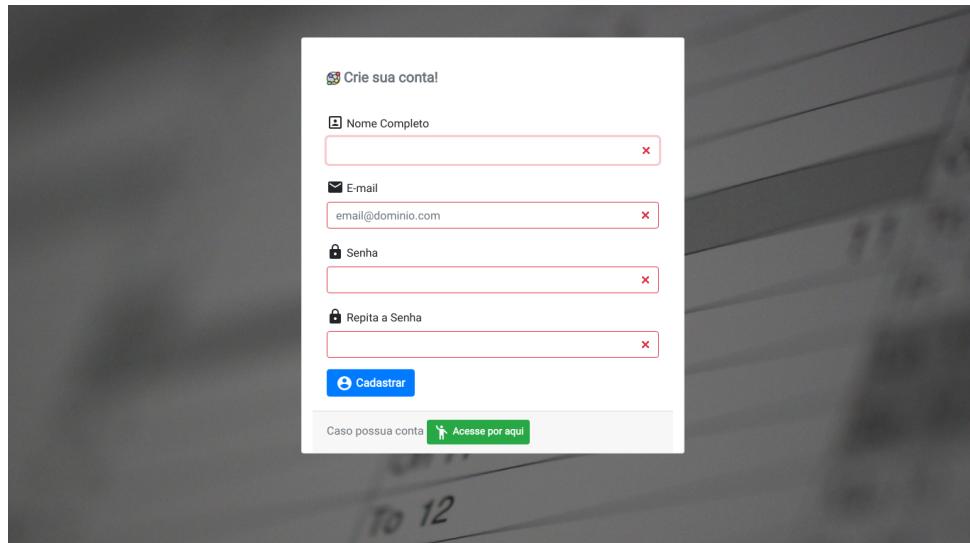


Figura 27 – Tela de Cadastro de Usuário

A partir do momento em que o usuário realizar seu cadastro, será possível efetuar *login* no sistema. Para isso o usuário deverá submeter seu *e-mail* e senha (conforme informado no cadastro).

A.2.2 Dashboard

Após efetuar o *login*, o usuário estará na tela de **Dashboard** do sistema, na qual visualizará seus **espaços de trabalho**, o **menu de navegação principal**, no topo da página, além da opção de **criar novos espaços de trabalho**.

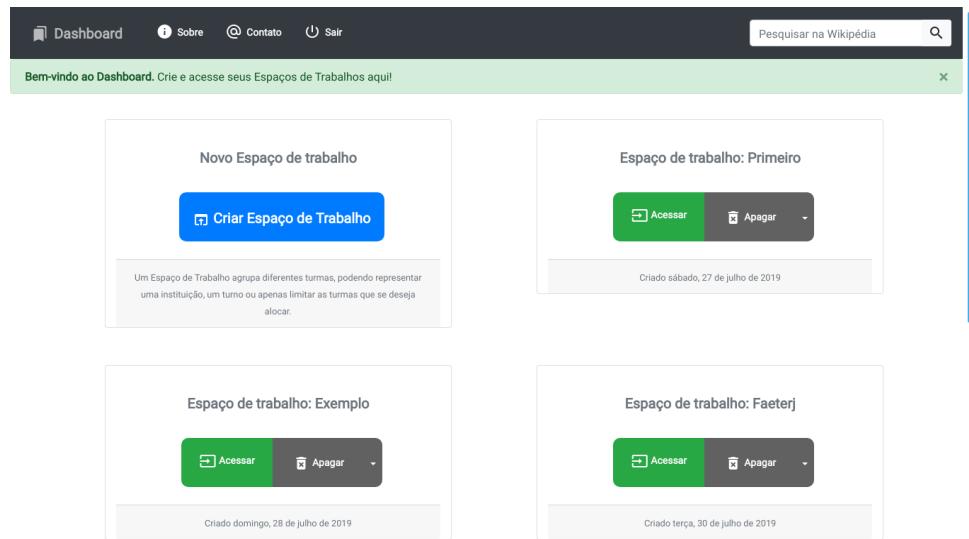


Figura 28 – Tela de Dashboard

Um **espaço de trabalho** aqui é análogo a uma instituição, turno ou qualquer outra forma pela qual o usuário deseja separar as turmas que deverão ter as grades de prova geradas.

Para se **criar um novo espaço de trabalho** deve-se clicar no botão azul escrito "Criar Espaço de Trabalho" no **Dashboard**, e informar o nome que o espaço de trabalho deverá ter, como mostrado na figura 29.

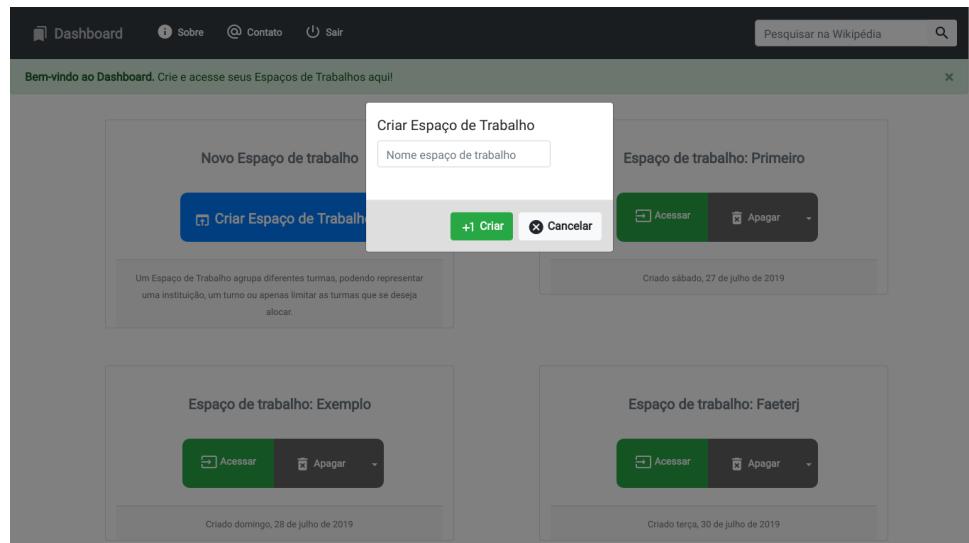


Figura 29 – Tela de Criação de Novo Espaço de Trabalho

No canto direito do **menu de navegação principal** temos uma barra de pesquisa, que serve para auxiliar o usuário em caso de dúvidas com termos técnicos, fazendo consultas diretamente na enciclopédia online <[Wikipédia.org](https://pt.wikipedia.org)> observado na figura 30, possibilitando dessa forma esclarecer as dúvidas sem ser necessário deixar a tela do sistema como apresentado na figura 31.

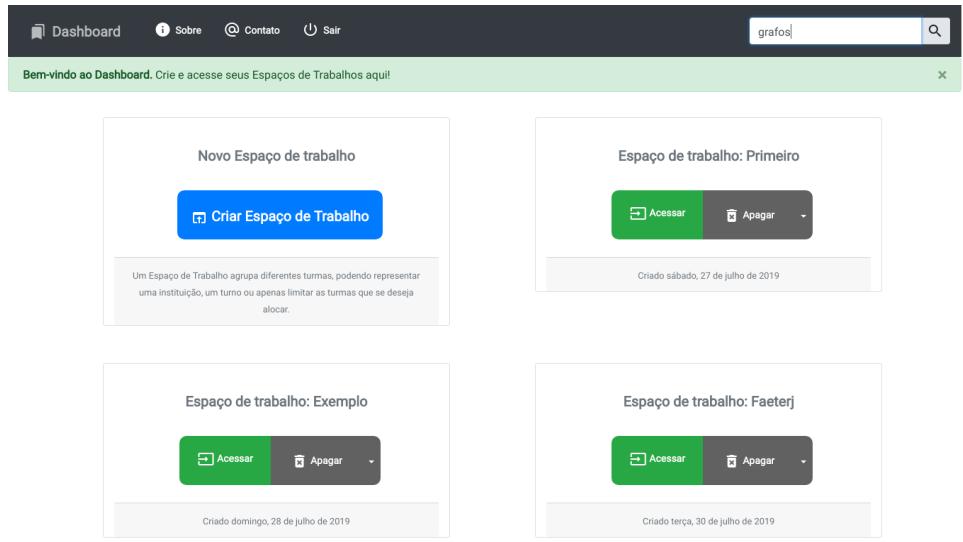


Figura 30 – Tela de Campo de Busca na Wikipédia



Figura 31 – Tela de Campo de Busca na Wikipédia Pop-up

Ainda no **menu de navegação principal**, da esquerda para direita observamos as seguintes opções:

Em **Sobre** que apresentará informações sobre o sistema, como o autor, os objetivos do projeto e como foi realizado o desenvolvimento do protótipo, assim como vemos na figura 32



Figura 32 – Tela Sobre o Sistema

Em sequencia temos a opção **Contato**, que permite o envio de um *e-mail*, conforme visto na figura 33, enviando-o para o desenvolvedor do sistema, para informar de possíveis problemas ou possibilitar o contato por qualquer que seja o motivo.

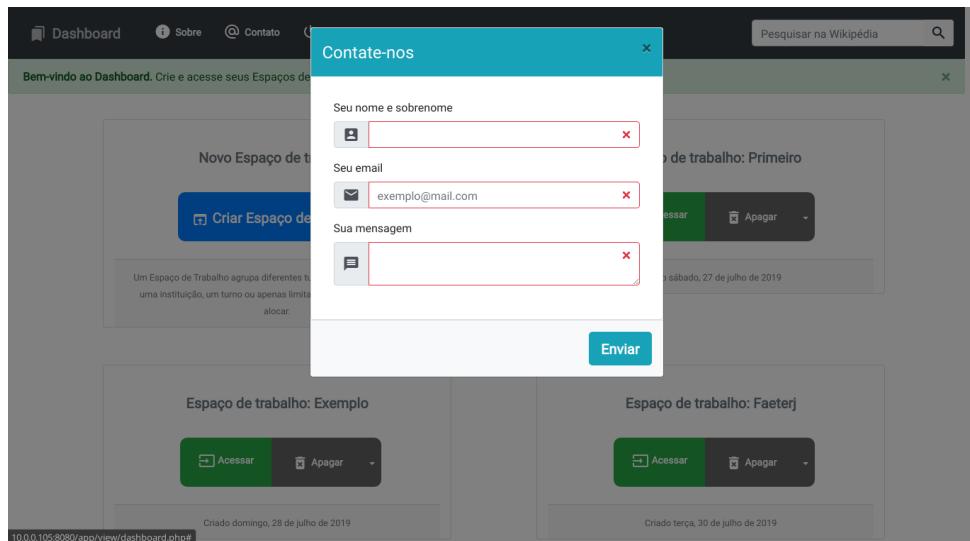


Figura 33 – Tela de Contato

No **Dashboard** temos ainda em cada espaço de trabalho as opções de acessar, apagar e um menu flutuante, como mostrada na figura 34, com opções adicionais, facilitando o acesso à algumas funcionalidades do espaço de trabalho como importar e exportar as turmas presentes sem precisar acessar o espaço de trabalho.

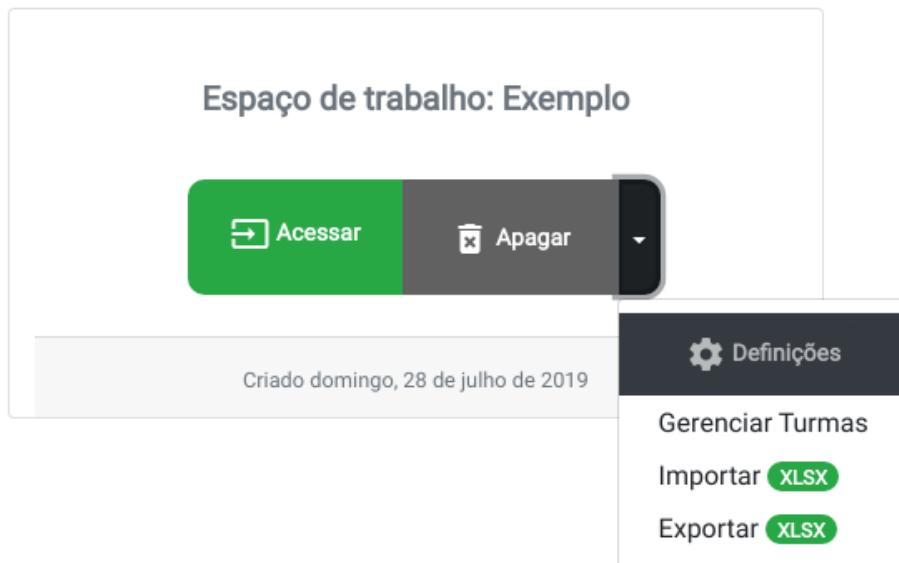


Figura 34 – Opções Rápidas do Espaço de Trabalho

A.2.3 Definições do Espaço de Trabalho

Ao acessar um espaço de trabalho, o usuário é redirecionado para a tela de **Definições do Espaço de Trabalho**, na qual é possível acessar as opções de configuração daquele espaço.

Dentre as suas opções (figura 35) temos a opção de criar uma tabela de horários, gerenciar as turmas pertencentes ao espaço de trabalho, adicionar turmas manualmente ou por arquivo (como visto na figura 36) ou exportar as turmas do espaço (figura 37), ainda temos a opção **Robô**, que demonstra uma explicação do que é o "Robô" usado no sistema (figura 38).

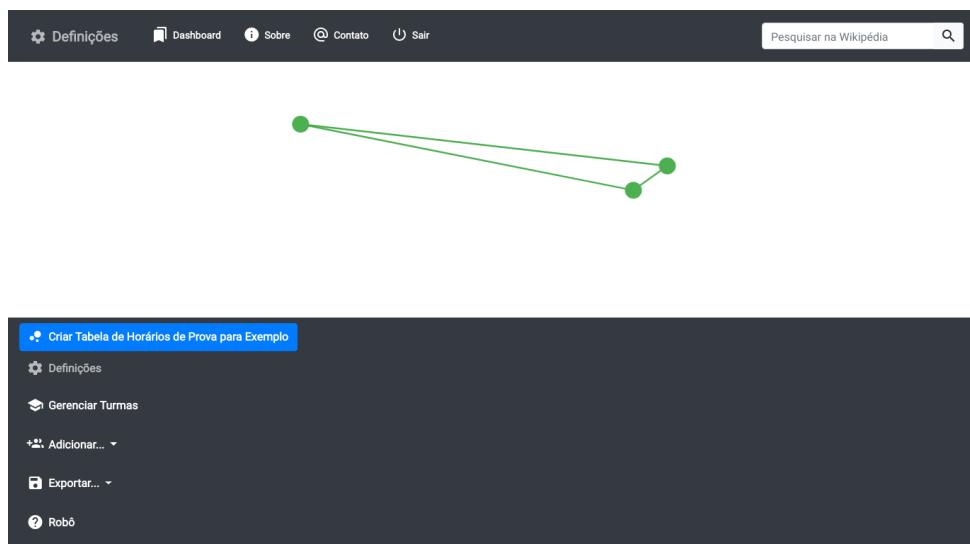


Figura 35 – Tela de Definições do Espaço de Trabalho

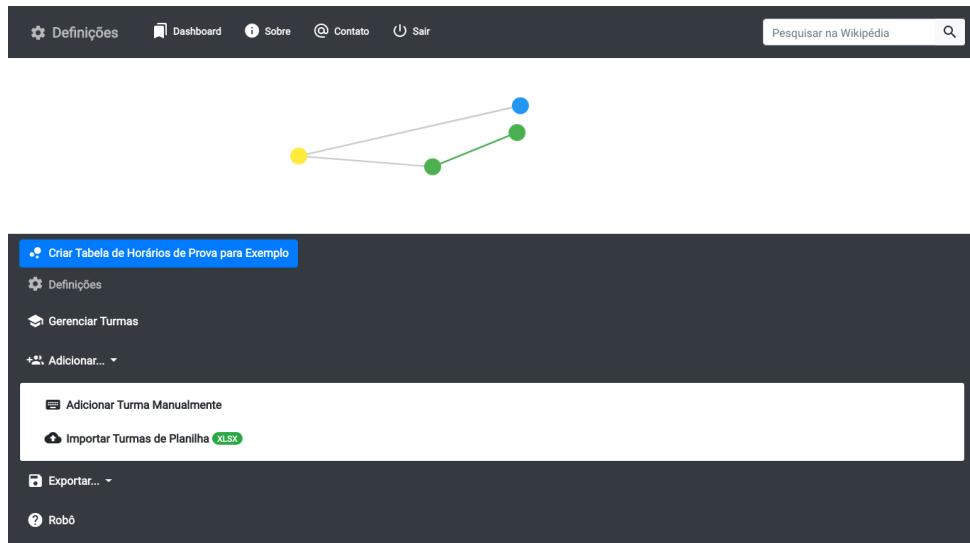


Figura 36 – Tela de Definições do Espaço de Trabalho - Caixa de Seleção para Adicionar Dados

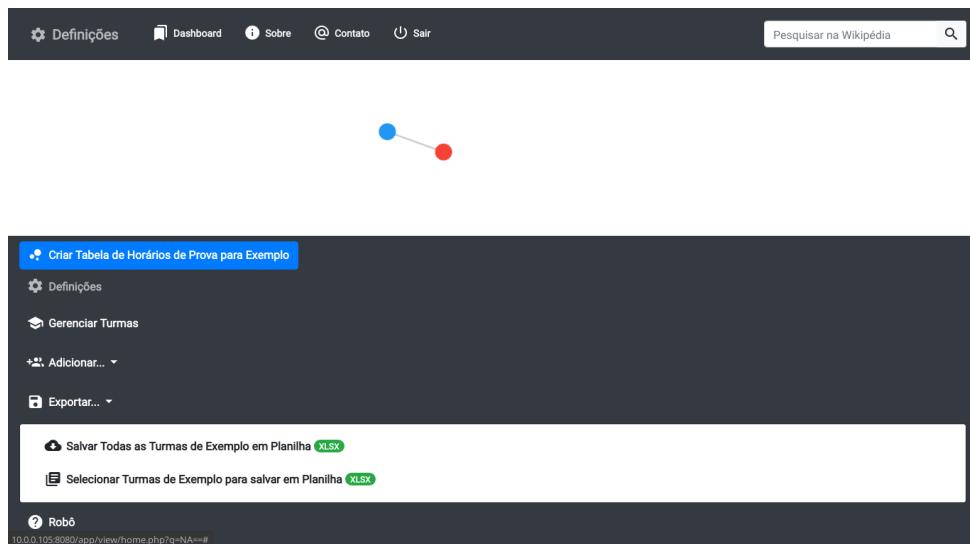


Figura 37 – Tela de Definições do Espaço de Trabalho - Caixa de Seleção para Exportar Dados

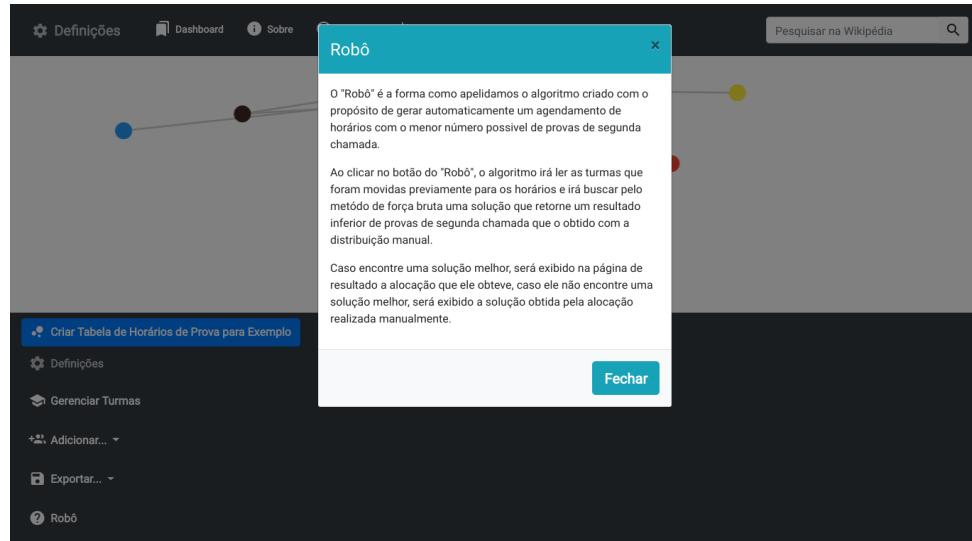


Figura 38 – Tela de Explicação da Solução através do uso do Robô

A.2.4 Gerenciar Turmas

Ao acessar a opção de **Gerenciar Turmas**, o usuário é direcionado para a página em que visualizará as turmas que estão presentes naquele espaço de trabalho (figura 39). Nesta página o usuário poderá **adicionar uma turma**, e caso já tenha adicionado turmas, terá as opções de visualizar, editar e apagar as turmas e seus alunos.

Quando selecionada a opção de **visualizar turma**, o usuário tem acesso a uma listagem de alunos que pertencem aquela turma (figura 40), a opção de retornar a tela de gerenciamento de turmas, e a de edição da turma (figura 41).

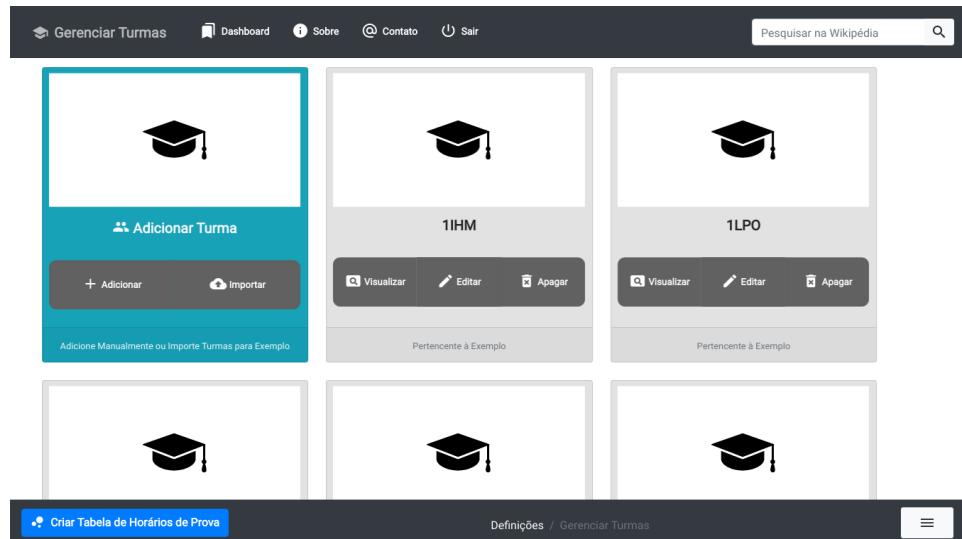


Figura 39 – Tela de Gerenciamento de Turmas

#	Matrícula	Nome
01	1910478300028	Ana Júlia Borges
02	1910478300015	Ana Júlia Duarte
03	1910478300072	Ana Laura Fernandes
04	1910478300066	Ana Luiza Miranda
05	1910478300011	Beatriz Dias Dos Santos
06	1910478300099	Beatriz Pereira De Souza
07	1910478300010	Benjamin Moreira Filho
08	1910478300002	Davi Lucca Nascimento Filho
09	1910478300008	Elisa Rodrigues Dos Santos

Definições / Gerenciar Turmas / Exibir Alunos

Figura 40 – Tela de Visualização da Turma

#	Matrícula	Nome
24	1910478300025	Liz Nascimento Dos Santos
25	1910478300022	Luiza Duarte De Andrade
26	1910478300029	Luiza Miranda Da Silva
27	1910478300026	Luiza Santos De Oliveira
28	1910478300007	Manuela Medeiros De Andrade
29	1910478300027	Marina Lima
30	1910478300060	Sarah Ferreira De Andrade
31	1910478300013	Sophia Carvalho De Andrade
32	1910478300005	Sophia Mendes Sá
33	1910478300023	Sophia Ramos De Oliveira
34	1910478300020	Vicente Barbosa De Andrade
35	1910478300014	Vicente Lima De Oliveira

← Voltar / Editar 1IHM

Definições / Gerenciar Turmas / Exibir Alunos

Figura 41 – Tela de Visualização da Turma – Cont.

Para se **editar uma turma**, o usuário será direcionado para a página de edição (figura 42), onde poderá modificar a matrícula ou nome de um aluno em específico, remove-lo da turma ou adicionar um novo aluno na turma (figura 43).

##	Matrícula	Nome	
01	1910478300028	Ana Júlia Borges	<button>Remover</button>
02	1910478300015	Ana Júlia Duarte	<button>Remover</button>
03	1910478300072	Ana Laura Fernandes	<button>Remover</button>
04	1910478300066	Ana Luiza Miranda	<button>Remover</button>
05	1910478300001	Beatriz Dias Dos Santos	<button>Remover</button>
06	1910478300099	Beatriz Pereira De Souza	<button>Remover</button>
07	1910478300010	Benjamin Moreira Filho	<button>Remover</button>
08	1910478300002	Davi Lucca Nascimento Filho	<button>Remover</button>
09	1910478300008	Elisa Rodrigues Dos Santos	<button>Remover</button>

Definições / Gerenciar Turmas / Editar Alunos

Figura 42 – Tela de Edição de Turmas

26	1910478300029	Luiza Miranda Da Silva	<button>Remover</button>
27	1910478300026	Luiza Santos De Oliveira	<button>Remover</button>
28	1910478300007	Manuela Medeiros De Andrade	<button>Remover</button>
29	1910478300027	Marina Lima	<button>Remover</button>
30	1910478300060	Sarah Ferreira De Andrade	<button>Remover</button>
31	1910478300013	Sophia Carvalho De Andrade	<button>Remover</button>
32	1910478300005	Sophia Mendes Sá	<button>Remover</button>
33	1910478300023	Sophia Ramos De Oliveira	<button>Remover</button>
34	1910478300020	Vicente Barbosa De Andrade	<button>Remover</button>
35	1910478300014	Vicente Lima De Oliveira	<button>Remover</button>

+ Adicionar Aluno

← Voltar Salvar Alterações

Definições / Gerenciar Turmas / Editar Alunos

Figura 43 – Tela de Edição de Turmas – Cont.

Para se **apagar uma turma**, uma mensagem irá informar que, ao apagar a turma ela não poderá ser recuperada posteriormente (figura 44).

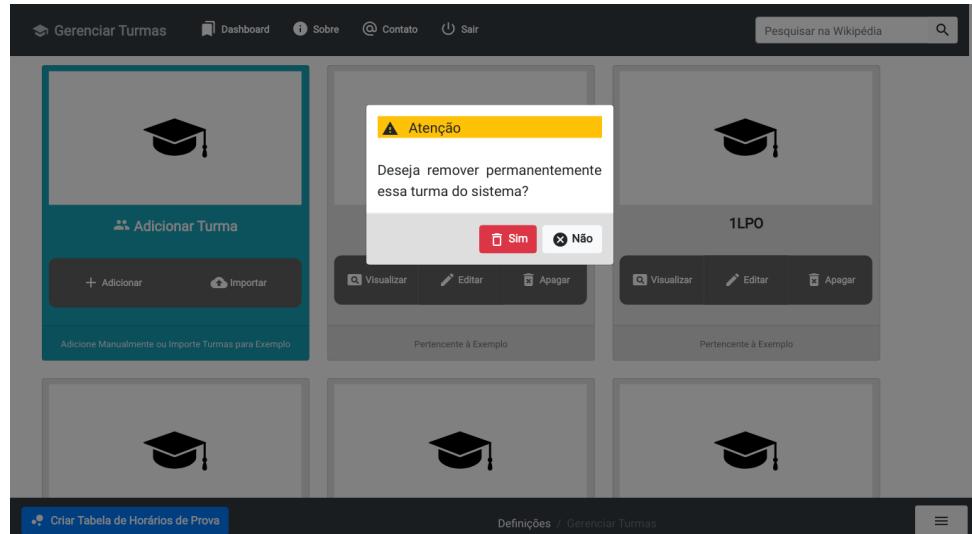


Figura 44 – Notificação ao Apagar Turma

Para criar uma nova turma o usuário deverá **Adicionar Turma** na tela de **Gerenciamento de Turmas** (conforme na figura 39), e para incluir a nova turma deverá informar o nome dessa turma (figura 45), e opcionalmente deverá informar os alunos que a compõe, informando a matrícula e o nome completo do aluno (figura 46).

Figura 45 – Tela de Criação de Nova Turma

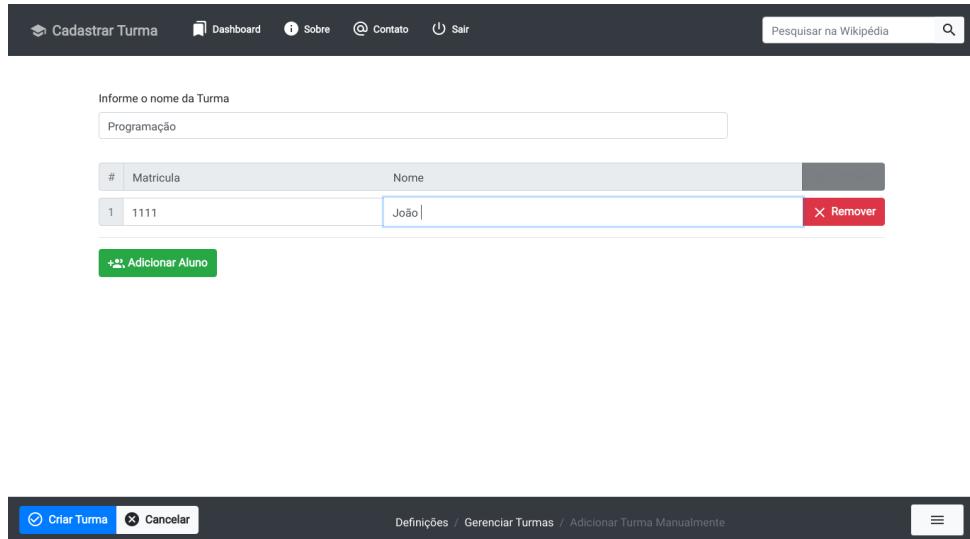


Figura 46 – Tela de Criação de Nova Turma – Cont.

Outra opção na página de Gerenciamento de turmas é a **inclusão de turmas por importação de um arquivo XSLX** (planilha *MS Excel*), que deve estar formatada de acordo com o padrão do sistema e assim só será necessário realizar o *update* do arquivo clicando em enviar e as turmas serão importadas para o espaço de trabalho (figura 47).



Figura 47 – Tela de Importação de Turmas por Arquivo

A.2.5 Alocar Turmas em Horários de Prova

A tela onde iremos **alocar as turmas nos horários de prova** é a principal página do sistema, pois é aqui que criaremos as tabelas de horários de prova (figura 48). Devemos informar nessa página a quantidade de horários que dispomos para realizar as provas (na figura 49 foi definido que existem três horários disponíveis para realizar a prova).

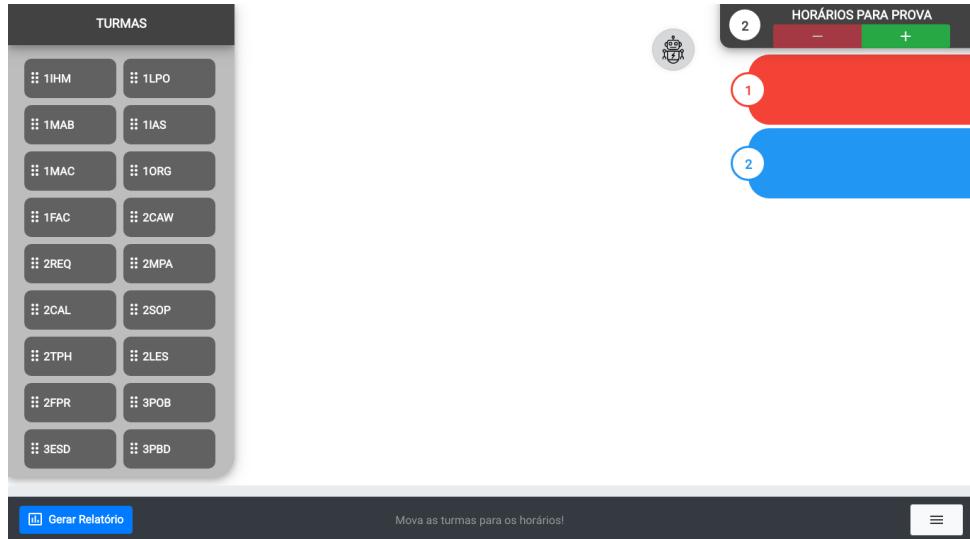


Figura 48 – Tela de Alocar Turmas em 2 Horários

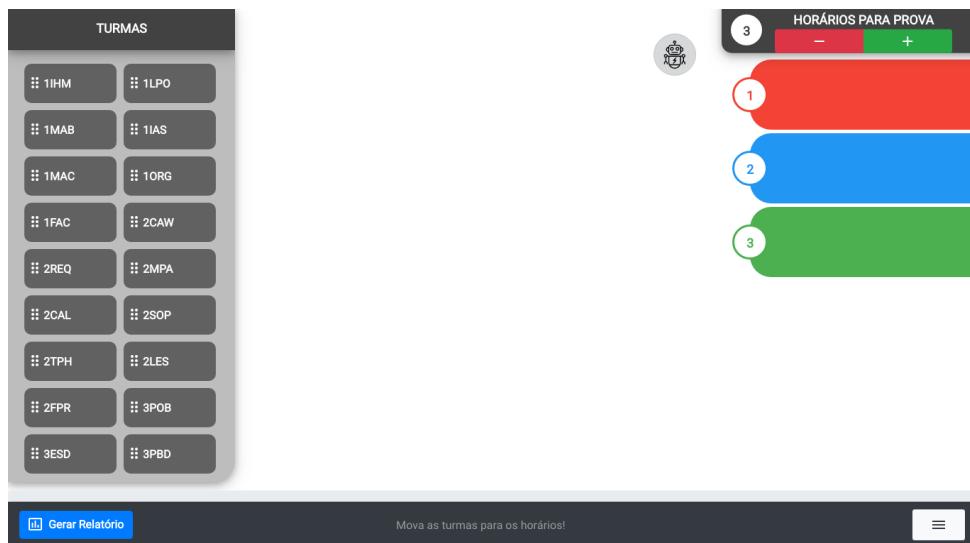


Figura 49 – Tela de Alocar Turmas em 3 Horários

Na barra situada na parte de baixo na página temos o menu secundário (figura 50), que apresenta as opções referentes ao espaço de trabalho e as turmas, assim como opções já apresentadas.

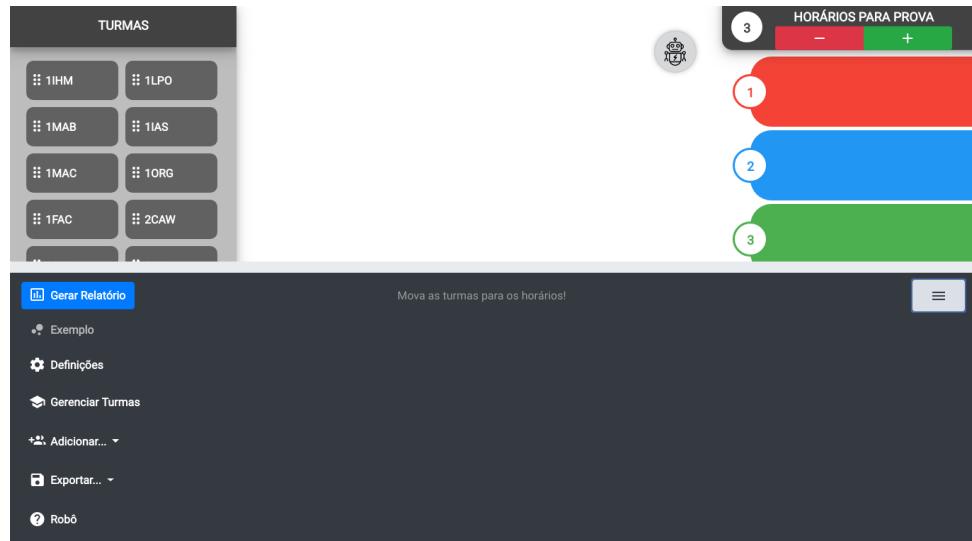


Figura 50 – Tela de Alocar Turmas em 3 Horários – Menu

Tabela 15 – Exemplos de alocações e seus resultados

Figura	Moveu as Turmas	Para os Horários	Conflitos	Taxa de Conflitos
51	1HM	1	0	0%
52	1IAS	2	0	0%
53	1MAB	3	0	0%
54	1LPO e 1IAS	2 e 1	8	15%
55	1MAC e 1ORG	2 e 3	26	19%
56	TODAS	1	140	100%
57	1LPO, 1MAB, 1IAS	2	57	41%

Para se alocar uma turma em um horário basta clicar na turma, manter pressionado e arrastar até o *slot* de horário desejado. Na tabela 15 temos descrita uma sequência de movimentações para exemplificar o funcionamento dessa tela.

Os conflitos são correspondentes ao número de provas de segunda chamada, necessárias para a realização daquela tabela de horários de prova.

A taxa de conflitos é obtida através do maior número de conflito possível para as turmas selecionadas, dividido pela quantidade de conflitos da solução atual multiplicado por 100.

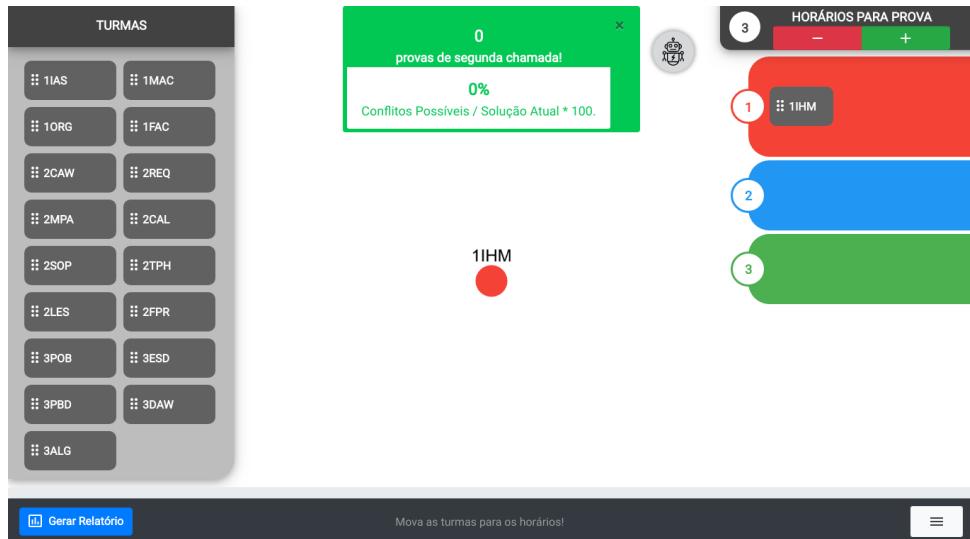


Figura 51 – Tela de Alocar Turmas – Exemplo 1

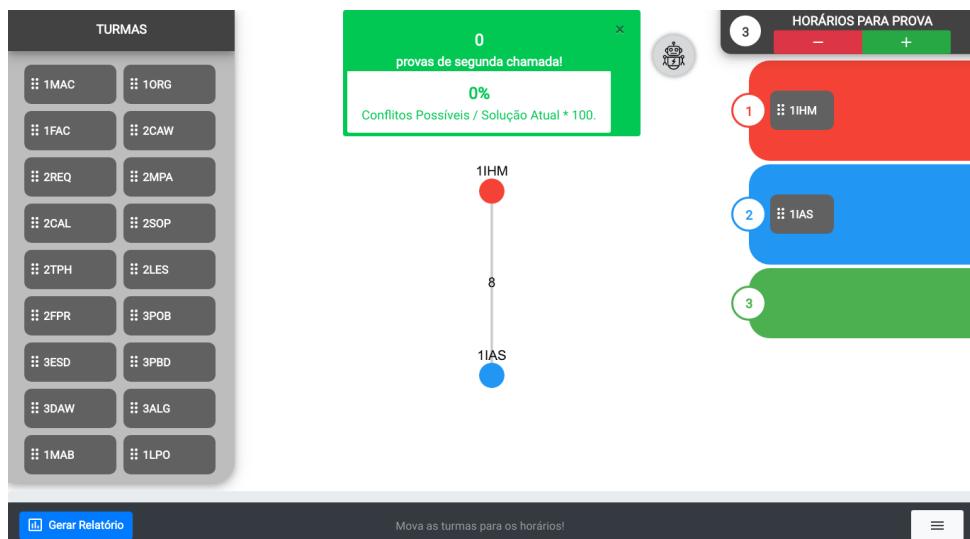


Figura 52 – Tela de Alocar Turmas – Exemplo 2

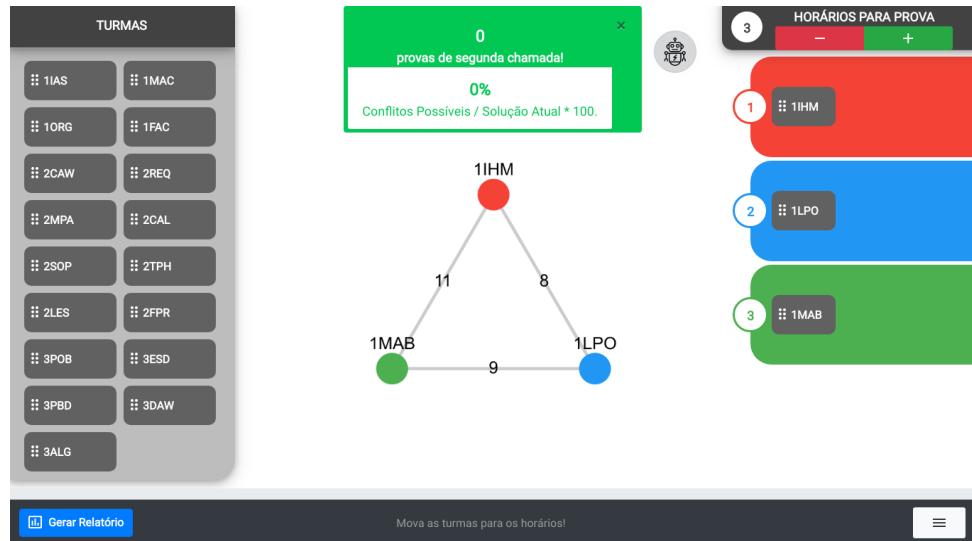


Figura 53 – Tela de Alocar Turmas – Exemplo 3

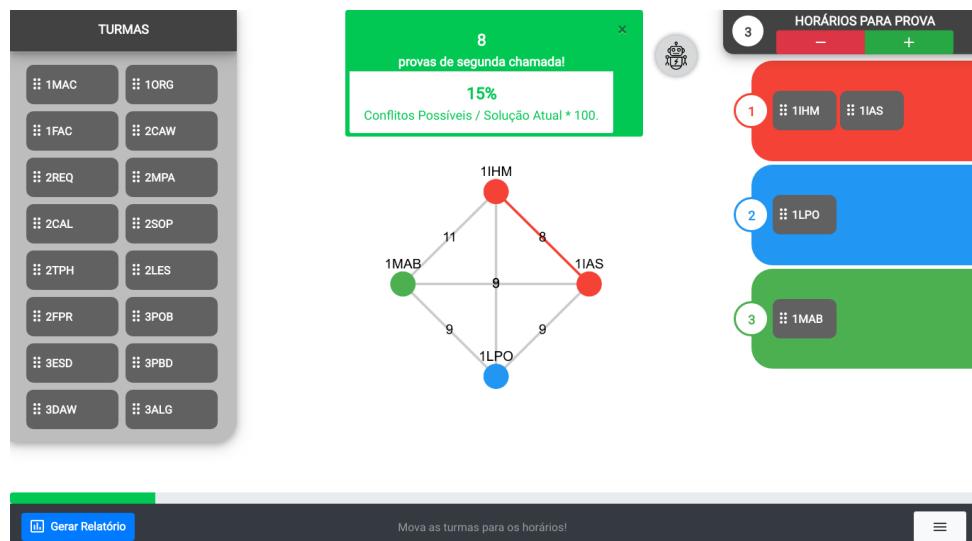


Figura 54 – Tela de Alocar Turmas – Exemplo 4

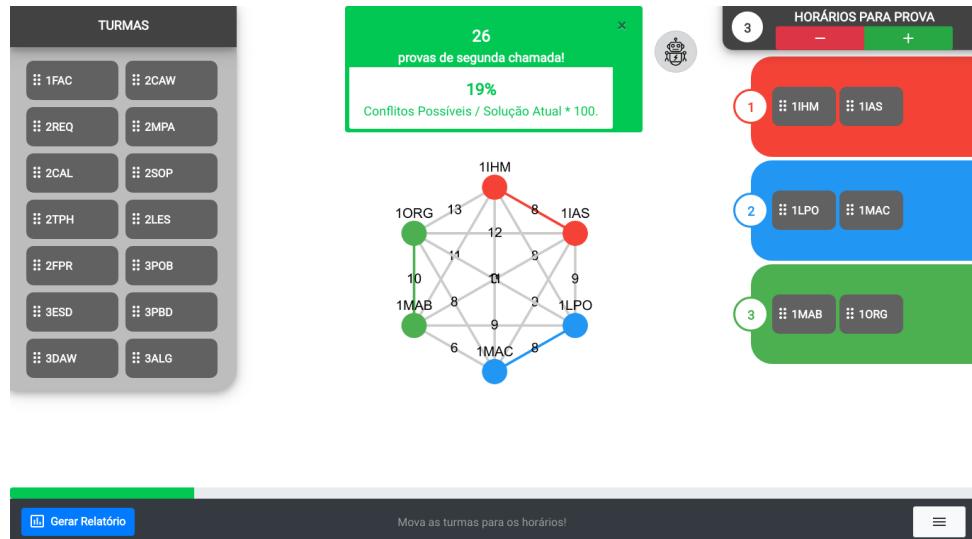


Figura 55 – Tela de Alocar Turmas – Exemplo 5

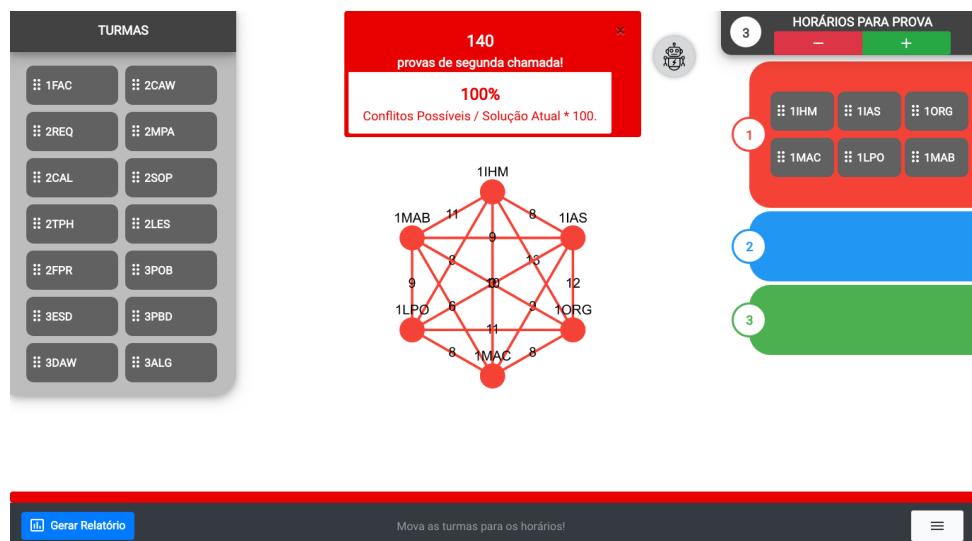


Figura 56 – Tela de Alocar Turmas – Exemplo 6

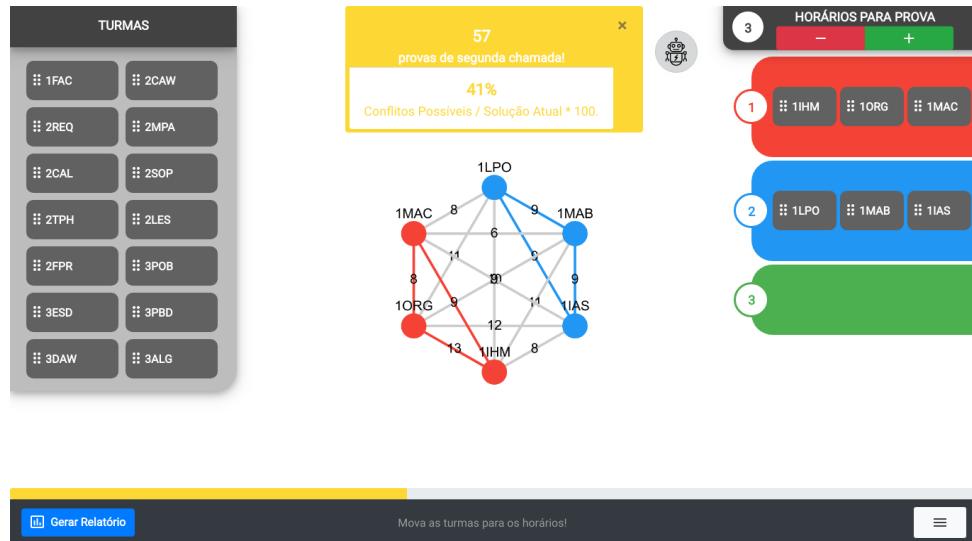


Figura 57 – Tela de Alocar Turmas – Exemplo 7

A.2.6 Gerar Relatório da Alocação

Agora vamos usar a tabela de horários de prova mostrada na figura 58 como a grade de horários que desejamos utilizar, dessa forma podemos clicar no botão **Gerar Relatório** para concluir a alocação das turmas e visualizar o resultado de forma detalhada como nas figuras 59 e 60.

Na figura 61 percebemos que em caso de clicarmos na linha da imagem (aresta do Grafo) será exibido os alunos que pertencem as duas turmas que a linha conecta. Já na figura 62 vemos que ao clicar na turma (no vértice do Grafo) temos a lista com todos os alunos pertencentes aquela turma.

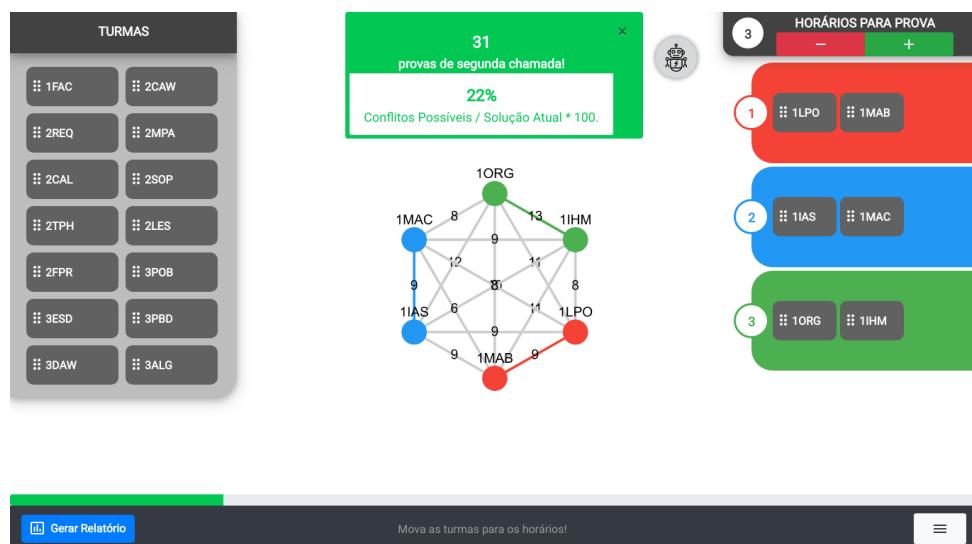


Figura 58 – Gerando uma Grade de Prova – pt. 1



Figura 59 – Gerando uma Grade de Prova – pt. 2

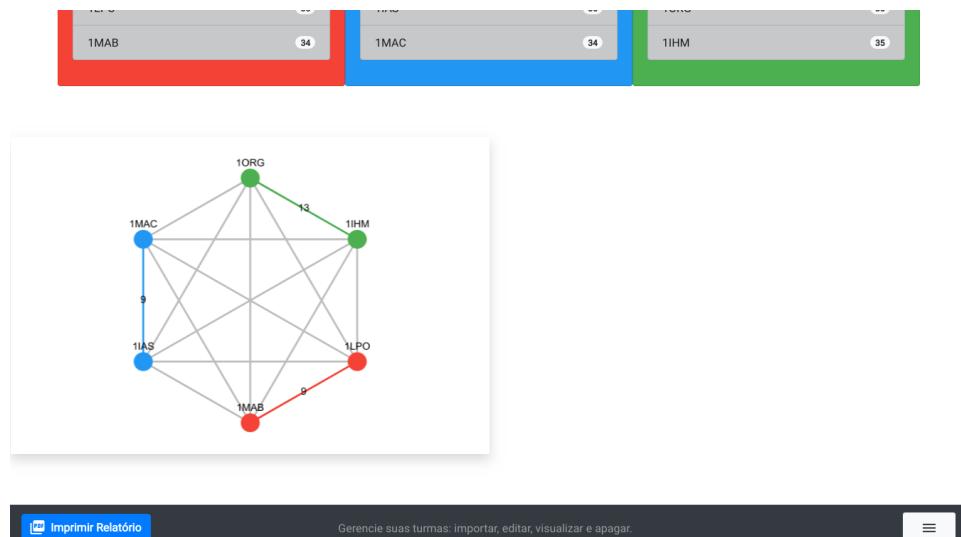


Figura 60 – Gerando uma Grade de Prova – pt. 3

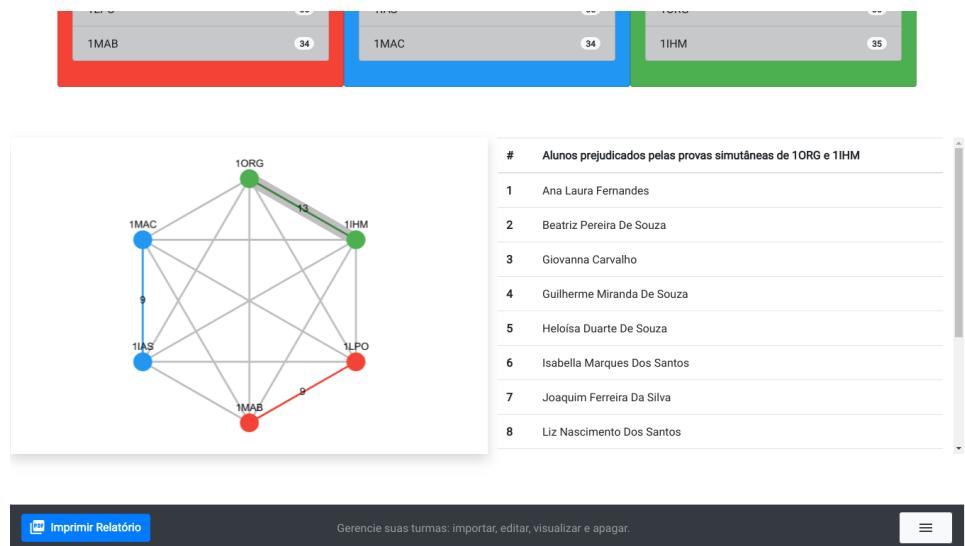


Figura 61 – Gerando uma Grade de Prova – pt. 4

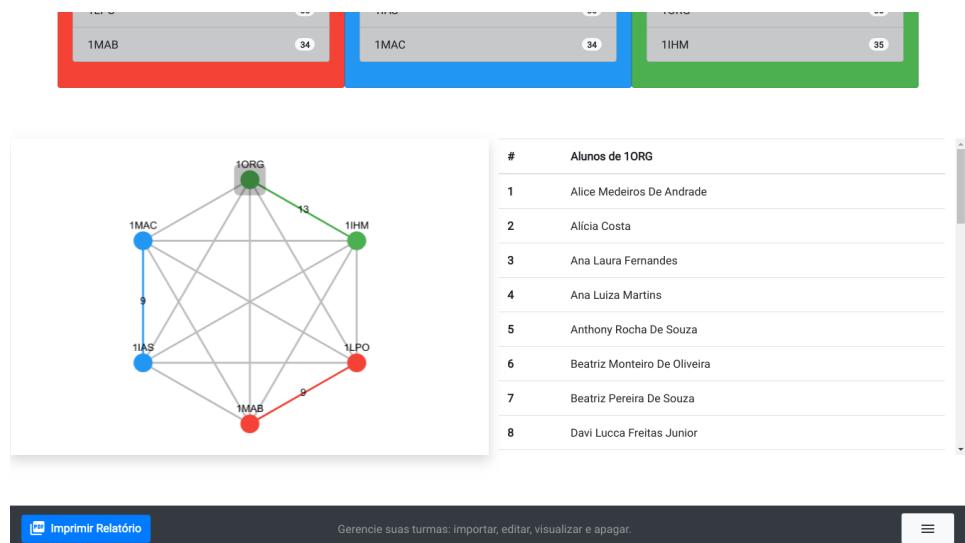


Figura 62 – Gerando uma Grade de Prova – pt. 5

Ao final da página de relatório temos uma opção descrita como "Deseja tentar encontrar um resultado melhor usando nosso robô?" (figura 63), essa opção irá utilizar o algoritmo criado para o sistema para buscar a melhor distribuição de turmas nos horários, isto é a alocação que vai gerar o menor número de provas de segunda chamada por conflitos de horário.

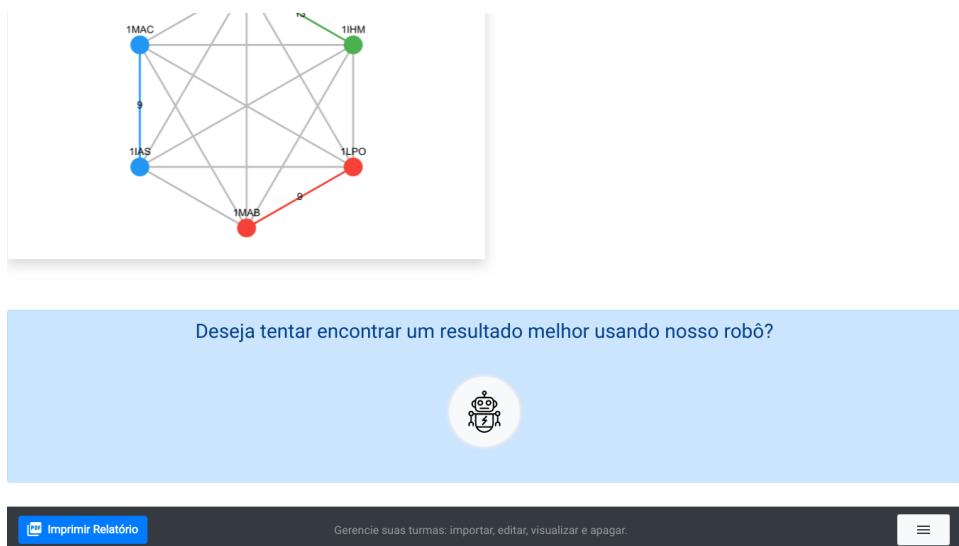


Figura 63 – Gerando uma Grade de Prova – pt. 6

Caso o usuário opte por essa opção ele será redirecionado para uma nova página de **relatório** (figura 64).

Vale dizer que caso o usuário clicasse no **botão do "Robô"** na página de **criação de tabela de horário** (figuras 48 e 49) ele seria direcionado para esse mesmo relatório.

Vemos na figura 64 que além de apresentar a quantidade de provas necessárias para realização das provas, esse relatório também irá apresentar a quantidade de provas necessárias na solução manual, que foi gerada anteriormente, e o quanto o "Robô" conseguiu melhorar o resultado manual.

Além das diferenças já citadas, a página vai apresentar as mesmas informações quanto aos horários em que as turmas foram alocadas, e uma figura interativa (de um Grafo) que representa a relação entre as turmas, contanto não haverá mais a opção de tentar uma solução melhor com o "Robô" já que essa se trata da melhor solução possível (figura 65).



Figura 64 – Gerando uma Grade de Prova – pt. 7

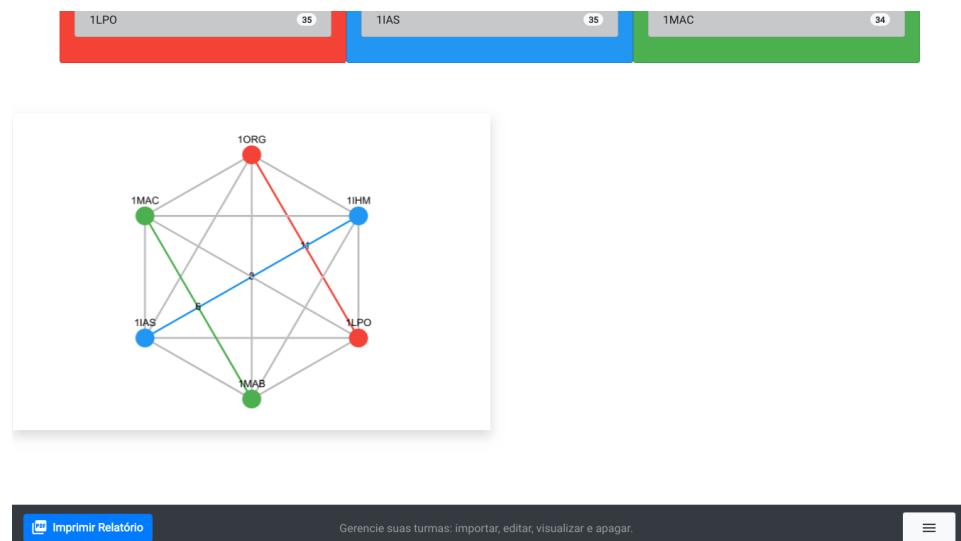


Figura 65 – Gerando uma Grade de Prova – pt. 8

A.2.7 Imprimir Relatório

Na base da tela de relatório temos o menu secundário com algumas opções semelhantes as que já foram apresentadas (figura 66), e além dessas existe a opção de **Imprimir Relatório**, a qual possibilita gerar um arquivo para impressão ou para download (figura 67).

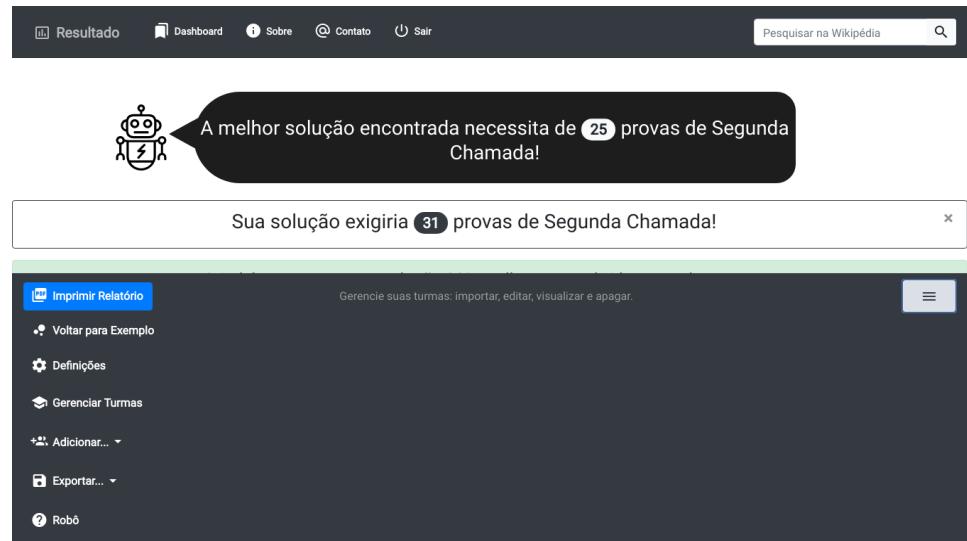


Figura 66 – Gerando uma Grade de Prova – pt. 9

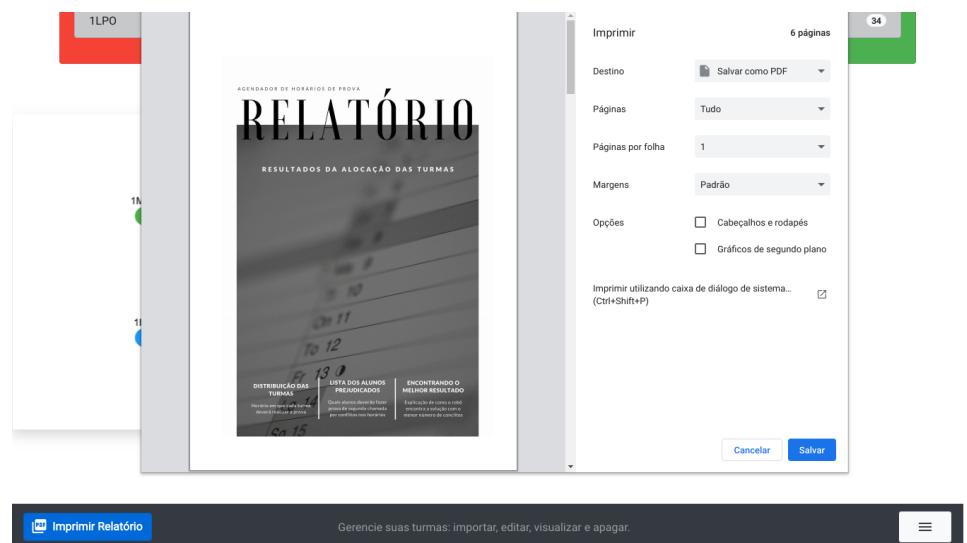


Figura 67 – Gerando Relatório

ANEXO B – RELATÓRIO

AGENDADOR DE HORÁRIOS DE PROVA

RELATÓRIO

RESULTADOS DA ALOCAÇÃO DAS TURMAS

DISTRIBUIÇÃO DAS TURMAS

Horário em que cada turma deverá realizar a prova

LISTA DOS ALUNOS PREJUDICADOS

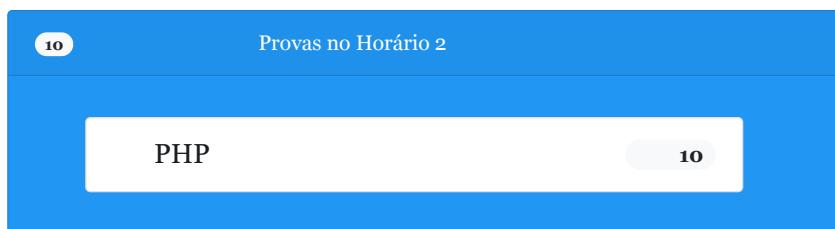
Quais alunos deverão fazer prova de segunda chamada por conflitos nos horários

ENCONTRANDO O MELHOR RESULTADO

Explicação de como o robô encontra a solução com o menor número de conflitos

Resultados obtidos para realização das provas

Para aplicar as provas com sua solução, serão necessárias 3 provas de Segunda Chamada.



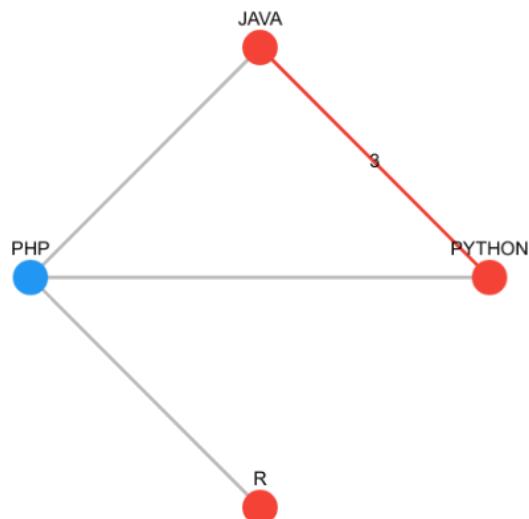
Horários para realização dos Exames

Cada horário, representado por um retângulo de uma cor específica, possui as turmas que deverão realizar os exames em um mesmo horário.

Sobreposto à informação da turma, está a quantidade de alunos matriculados, dessa forma podemos saber o exato número de provas que cada disciplina precisará. O número total de provas necessárias em cada horário é apresentado no topo do retângulo, realizando o somatório das provas por turma, sem o decréscimo das avaliações para os alunos prejudicados, caso existam.

Resultado em forma de Grafo.

Com a visualização do resultado na forma de um grafo pode ser observado graficamente quais as turmas possuem alunos em comum e quais dessas turmas acabaram sendo alocadas em um mesmo horário.



Provas de segunda chamada

A lista a seguir apresenta as turmas que possuem conflitos entre alunos, e os alunos que necessitarão de provas de segunda chamada.

Alunos matriculados em JAVA e PYTHON

#	Nome	Matrícula
1	Ana	6
2	Betina	3
3	Ulisses	9

Método de Solução Utilizado

Visando permitir o uso do poder computacional na busca de uma solução para o problema, foi necessário modelar o problema de forma matemática.

Dessa forma foi optado o uso da teoria dos grafos como estrutura matemática visando modelar o problema.

O que é Teoria dos Grafos?

Um **grafo** é definido por Szwarcfiter (1986) como sendo uma estrutura matemática composta por um conjunto finito não vazio de vértices V e um conjunto E de pares, não ordenados, de diferentes vértices denominados arestas.

Um grafo é composto por uma série de elementos entre os quais se destacam:

- **Vértices:** um vértice também é chamado de V e representado graficamente através de um ponto, a quantidade dos vértices de um grafo é expressa por $|V|$, onde um grafo com $|V| = 1$ é classificado como um grafo trivial;
- **Arestas:** uma aresta e é formada por um par de vértices, com isso $e = (v, w)$. Dessa forma os vértices v e w estão conectados as extremidades da aresta e dessa forma são denominados adjacentes. Já uma aresta é denominada incidente aos vértices em seus extremos v e w ;

Como transformamos as turmas em Grafos?

Para utilizar a teoria dos grafos para modelar esse problema foi necessário a utilização dos grafos de interseção.

Pinto (2018) define um Grafo de Interseção, quando seja possível esse grafo $G(V, E)$ representar as relações entre elementos de um conjunto através das relações entre seus vértices $v \in V$.

Dessa forma iremos relacionar os alunos das turmas aos vértices do Grafo, e criaremos arestas que indique a presença de alunos que componham ambas as turmas. Com isso teremos a turma como um grafo, que inclusive pode ser visualizada em sua forma geométrica.

Definindo os horários de forma automática.

Após realizar a modelagem na forma de um problema de grafo, iremos utilizar essa estrutura para criar uma grade de horário de provas.

Para definir quais provas poderão ser aplicadas em um mesmo horário, iremos considerar os alunos pertencentes á ambas turmas. Com isso, caso duas turmas não possuam alunos em comum, elas poderão realizar seus exames em um mesmo horário.

Agora, para realizarmos a atribuição das turmas em horários aproveitaremos a forma geométrica do grafo, atribuindo uma cor a cada horário e pintando o vértice da cor correspondente ao horário.

A partir desse momento iremos atribuir á primeira turma o primeiro horário, dessa forma

podemos pintar o primeiro vértice da primeira cor (vermelho, por exemplo).

Ao pintar o próximo vértice, poderemos ter um caso onde ele possua alunos em comum com a turma anteriormente pintada, nesse caso iremos pinta-lo com nossa segunda cor (azul, por exemplo). Mas se as turmas não possuírem alunos em comum (ou seja, não sejam ligadas por uma aresta) poderemos usar a mesma cor para pinta-la.

Seguiremos repetindo esse procedimento de olhar em que horário estão as turmas que possuem alunos em comum para não colocarmos ambas as turmas no mesmo horário, o que causaria a necessidade de se aplicar uma prova de segunda chamada para os alunos prejudicados.

Esse processo de pintar os vértices do grafo, de forma que vértices adjacentes não recebam mesma cor, consiste em uma área muito estudada dentro da Teoria de Grafos, que é a Coloração de Grafos.

O que é Coloração de Grafos?

Colorir um grafo G propriamente é definido por Goldbarg e Goldbarg (2012) como, atribuir cores aos seus vértices de forma que vértices adjacentes recebam cores distintas.

O principal problema em se colorir um grafo propriamente consiste em determinar qual o menor número possível para se colorir o grafo.

Esse número é chamado de número cromático do grafo ou $\chi(G)$. É um problema classificado como NP-Completo, sendo um dos problemas cuja solução não pode ser obtida em tempo polinomial de execução.

Contudo, a solução do problema de exames universitários por meio da coloração própria de vértices, não resolverá todos os casos possíveis, já que é possível existir um caso onde a quantidade de horários disponíveis para realização da prova é menor que o número cromático do grafo.

Nesse caso, como uma coloração própria não será suficiente, adotaremos uma nova forma de solução.

Considerando que não será possível separar todas as turmas que possuem alunos em comum em horários de forma que não gere conflitos, iremos buscar uma solução que minimize a quantidade de provas de segunda chamada, ou seja, a número de alunos prejudicados.

Para sabermos quantos alunos estarão sendo prejudicados precisaremos incluir mais uma etapa em nossa modelagem.

Iremos atribuir às arestas do grafo um peso, correspondente a quantidade de alunos que pertencem às duas turmas. Dessa forma teremos um **Grafo Ponderado em Areias**.

Colorir o grafo agora consistirá em encontrar uma coloração não própria cuja soma dos pesos das arestas incidentes em vértices coloridos com mesma cor seja o menor possível.

Esse processo é chamado de Generalized Graph Coloring, onde efetuamos um tipo especial de coloração de modo a atingir algum objetivo pré-estabelecido.

Generalized Graph Coloring Problem?

Definido por Vredeveld (2002), o k-GGCP é um problema que consiste em um grafo $G = (V, E)$, uma função de peso $z : E \rightarrow Z$ nas arestas, e um número inteiro $k \geq 2$.

Em que deseja-se encontrar uma atribuição de cores $c : V \rightarrow \{1, \dots, k\}$ dos vértices que minimize o peso total das arestas monocromáticas do grafo, isto é, arestas que incidem em vértices com mesma cor.

Enfim o Resultado!

Ao buscar a distribuição que cause o menor número de conflitos chegaremos à coloração que será o resultado desejado para o problema. Assim como o obtido pelo robô no sistema.

O Software de Agendamento de Horários de Prova

Este sistema foi desenvolvido por **Filipe Rodrigues Cardoso da Silva**. A partir do ano de 2019, com objetivo de ser um protótipo funcional para o Trabalho de Conclusão de Curso de Análise de Sistemas Informatizados, intitulado de "**Utilizando Coloração de Grafos de Interseção para Resolver o Problema de Programação de Horários de Exames Universitários**" vinculado a Faculdade de Educação Tecnológica do Estado do Rio de Janeiro - FAETERJ-Rio.

O Agendador tem como finalidade criar horários de provas universitárias de forma que o menor número possível de alunos tenham que realizar provas de segunda chamada, em decorrência de ter duas ou mais provas de diferentes turmas num mesmo horário.

O sistema foi desenvolvido usando MariaDB para o banco de dados, PHP no backend e HTML, CSS e JavaScript no frontend. Utilizando o framework Bootstrap para criar uma interface mais coesa e simples, e a biblioteca Cytoscape.js para possibilitar gerar grafos interativos.

Todo o código do sistema, a integração do trabalho escrito e os slides da apresentação podem ser obtidos através do repositório oficial do trabalho no Github: <https://github.com/FilipeRodrigues3003/TCC>.

REFERÊNCIAS

GOLDBARG, Marco Cesar; GOLDBARG, Elizabeth Ferreira Gouvêa. **Grafos: Conceitos, algoritmos e aplicações.** [S.l.]: Elsevier Editora Ltda, 2012. v. 1.

PINTO, José Wilson Coura. **Grafos ORTH[h, s, t].** 2018. Tese (Tese (Doutorado)) – UFRJ/COPPE, 2018.

SZWARCIFTER, Jayme Luiz. **Grafos e Algoritmos Computacionais.** 2. ed. [S.l.]: Campus, 1986. 35-42, 169-195 p.

VREDEVELD, Tjark. **Combinatorial Approximation Algorithms: Guaranteed versus experimental performance.** 2002. Tese (PhD) – Technische Universiteit Eindhoven, 2002.

SILVA, Filipe Rodrigues Cardoso da. **Utilizando Coloração de Grafos de Interseção para Resolver o Problema de Programação de Horários de Exames Universitários.** 2020. (Trabalho de Conclusão de Curso - TCC) - Faculdade de Educação Tecnológica do Estado do Rio de Janeiro - FAETERJ-Rio, 2020.