

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO  
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ALLAN WENDLAND KRETZMANN

**GERAÇÃO DE GRADES HORÁRIAS ESCOLARES: AUTOMAÇÃO  
COM BASE NA META-HEURÍSTICA SIMULATED ANNEALING**

PROPOSTA DE TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO  
2023

ALLAN WENDLAND KRETZMANN

## **GERAÇÃO DE GRADES HORÁRIAS ESCOLARES: AUTOMAÇÃO COM BASE NA META-HEURÍSTICA SIMULATED ANNEALING**

Proposta de Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: André Yoshiaki Kashiwabara  
Universidade Tecnológica Federal do Paraná

CORNÉLIO PROCÓPIO  
2023



4.0 Internacional

Esta é a mais restritiva das seis licenças principais Creative Commons. Permite apenas que outros façam download dos trabalhos licenciados e os compartilhem desde que atribuam crédito ao autor, mas sem que possam alterá-los de nenhuma forma ou utilizá-los para fins comerciais.

## RESUMO

KRETZMANN, Allan. Geração de Grades Horárias Escolares: Automação com base na meta-heurística Simulated Annealing. 2023. 35 f. Proposta de Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2023.

O *High School Timetabling Problem* consiste na tarefa de alocar aulas a horários letivos em instituições de ensino, considerando restrições relacionadas aos professores, salas e disciplinas. Apesar de todos os avanços tecnológicos que experienciamos na atualidade, esta tarefa ainda é realizada primordialmente de maneira manual nas escolas brasileiras, produzindo grades horárias subótimas e consequentemente trazendo frustrações a todos os envolvidos, sejam estes alunos, docentes ou diretores. Todas as dificuldades relacionadas ao problema, somadas a novos desafios como os Itinerários Formativos do Novo Ensino Médio, pedem uma solução melhor. Este trabalho teve como objetivo implementar um sistema de geração automatizada das grades horárias atendendo o maior número de objetivos possível, através da aplicação de um algoritmo de otimização baseado na meta-heurística *Simulated Annealing*, técnica de otimização inspirada no processo de recozimento de materiais na metalurgia. A solução desenvolvida é uma aplicação *web* que segue uma arquitetura cliente-servidor empregando JavaScript no *frontend* e *backend*, além do otimizador desenvolvido na linguagem C++. Espera-se que a aplicação desenvolvida passe a fazer parte do ferramental de instituições de ensino e auxilie estas a entregar horários escolares que superem os diversos desafios associados e providenciem uma experiência melhor para todos.

**Palavras-chave:** High School Timetabling Problem. Simulated Annealing. Otimização multi-objetivo.

## ABSTRACT

KRETZMANN, Allan. School Timetable Generation: Simulated Annealing based Generation. 2023. 35 f. Proposta de Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2023.

The High School Timetabling Problem involves the task of assigning classes to class periods in educational institutions while considering constraints related to teachers, classrooms, and subjects. Despite the technological advancements we experience today, this task is still primarily done manually in Brazilian schools, resulting in suboptimal timetables and frustrations for all stakeholders, including students, teachers, and administrators. The difficulties associated with the problem, combined with new challenges provided by the new high school model implemented in the country, call for a better solution. This study has aimed to implement an automated timetabling system that satisfies as many objectives as possible by applying an optimization algorithm based on the Simulated Annealing metaheuristic, a technique inspired by the annealing process in metallurgy. The developed solution is a web application following a client-server architecture, employing JavaScript for both the frontend and backend, in addition to the optimizer implemented in the C++ language. We hope that the developed application becomes part of the toolkit for educational institutions and helps them deliver school timetables that overcome the various associated challenges and provide a better experience for all parties involved.

**Keywords:** High School Timetabling Problem. Simulated Annealing. Multi-objective optimization.

## LISTA DE FIGURAS

Figura 1 – Aulas constantes . . . . .	8
Figura 2 – Restrições . . . . .	9
Figura 3 – Exemplo de região . . . . .	11
Figura 4 – Exemplo de grupo de alinhamento . . . . .	12
Figura 5 – Efeito de grupo de alinhamento . . . . .	13
Figura 6 – Modelo Entidade-Relacionamento . . . . .	17
Figura 7 – Consultas de validação da modelagem . . . . .	18
Figura 8 – Consulta SQL de grade horária com sete salas . . . . .	18
Figura 9 – Resultado da consulta de uma grade no banco de dados . . . . .	19
Figura 10 – Diagrama de Casos de Uso . . . . .	20
Figura 11 – Tela - Listagem de Configurações de Grade . . . . .	21
Figura 12 – Tela - Estrutura da Escola - Professores . . . . .	21
Figura 13 – Tela - Estrutura da Escola - Salas e Turnos . . . . .	22
Figura 14 – Tela - Aulas por professor . . . . .	22
Figura 15 – Tela - Restrições . . . . .	23
Figura 16 – Tela - Horários . . . . .	23
Figura 17 – Modelo Entidade-Relacionamento com adição da tabela de usuários . . . . .	24
Figura 18 – Tela de Acesso . . . . .	25
Figura 19 – Método de Cadastro . . . . .	26
Figura 20 – Método de Login . . . . .	27
Figura 21 – Middleware de autenticação . . . . .	28
Figura 22 – Middleware de validação da configuração . . . . .	28
Figura 23 – Modelo Entidade-Relacionamento com Matérias . . . . .	30
Figura 24 – Estrutura da Escola com Matérias . . . . .	30
Figura 25 – Tela de configuração de quantidades de aulas por matéria . . . . .	31
Figura 26 – Visualização de grade horária com matérias . . . . .	31

## SUMÁRIO

<b>1 – INTRODUÇÃO</b>	<b>1</b>
1.1 DELIMITAÇÃO DO TEMA	1
1.2 PROBLEMAS E PREMISSAS	1
1.3 OBJETIVOS	2
1.3.1 Objetivo Geral	2
1.3.2 Objetivos Específicos	2
1.4 ESTRUTURA ORGANIZACIONAL	2
<b>2 – REVISÃO DE LITERATURA</b>	<b>3</b>
2.1 TIMETABLING PROBLEM	3
2.1.1 Highschool Timetabling Problem	3
2.2 NOVO ENSINO MÉDIO	3
2.3 SIMULATED ANNEALING	4
<b>3 – DESENVOLVIMENTO</b>	<b>5</b>
3.1 OTIMIZADOR INICIAL	5
3.2 PESOS E RESTRIÇÕES	6
3.2.1 Agrupamento de aulas	6
3.2.2 Constantes	7
3.2.3 Restrições	8
3.2.4 Regiões	10
3.2.5 Grupos de Alinhamento	12
3.2.6 Janelas	14
3.2.7 Preferências	14
3.2.8 Armazenamento de soluções	14
3.3 SERVIDOR E BANCO DE DADOS	16
3.4 INTERFACE	18
3.5 USUÁRIOS E AUTENTICAÇÃO	24
3.5.1 Adaptação do banco de dados	24
3.5.2 Tela de login	24
3.5.3 Rotas de autenticação	25
3.5.4 Middlewares	27
3.6 ALOCAÇÃO DE MATÉRIAS	29
3.6.1 Alteração no banco de dados	29
3.6.2 Alteração de telas	30
3.6.3 Alteração de métodos no servidor	31
3.6.4 Matérias no otimizador	31
3.7 VALIDAÇÃO DE CONFIGURAÇÕES	32
3.8 EXPORTAÇÃO DE GRADES HORÁRIAS	32
<b>4 – CONSIDERAÇÕES FINAIS</b>	<b>34</b>
<b>Referências</b>	<b>35</b>

## 1 INTRODUÇÃO

A cada ano letivo, se faz necessária a definição de grades horárias que atendam inúmeros requisitos para o bom funcionamento das mais diversas instituições de ensino. Estas exigências dependem e impactam diretamente os envolvidos, sejam estes professores, alunos, gestores, ou ainda, os responsáveis pelo planejamento dos horários. Os requisitos citados anteriormente podem mudar durante o ano, requerendo constantes atualizações das grades definidas. Esta tarefa, segundo [Bardadym \(1996\)](#), requer grande esforço quando realizada manualmente, frequentemente resultando em soluções subótimas.

Para o desenvolvimento dos horários escolares, vários critérios devem ser considerados, como o número de aulas de cada professor, salas, turnos, além das restrições associadas a cada professor em particular. A combinação destes fatores para a produção de um horário otimizado é conhecida como *school timetabling problem* ([FONSECA; SANTOS; CARRANO, 2016](#)). Para a escrita deste trabalho, foi realizada a leitura de artigos que abordam o tema em questão, deste modo os autores [Fonseca, Santos e Carrano \(2016\)](#), [Tan et al. \(2021\)](#) e [Abramson \(1991\)](#) se mostraram imprescindíveis para a sua realização.

Ao decorrer da revisão bibliográfica, foi reafirmada a relevância desta pesquisa e da elaboração de um projeto que solucione o problema, dada a complexidade e recorrência deste. [Poulsen \(2012\)](#) estima que a maioria das instituições de ensino brasileiras efetua essa tarefa de maneira manual, o que implica em atrasos para as escolas, além de desgaste entre os docentes envolvidos nas negociações de restrições. Isto justifica novamente a necessidade de uma solução automatizada que entregue grades otimizadas em um tempo hábil.

Diante deste contexto, fica evidente a necessidade de solucionar o seguinte problema: como automatizar a geração de grades horárias que atendam as diversas demandas das instituições de ensino? Por ser um problema combinatório, consideram-se técnicas viáveis *Simulated Annealing*, Busca Tabu, Algoritmos Genéticos, entre outros ([TAN et al., 2021](#)).

Como critério metodológico, será aplicado o procedimento de levantamento de requisitos com os encarregados por planejar as grades horárias nas escolas, para embasar uma análise quantitativa que traga informações para o desenvolvimento do projeto. Os dados coletados, juntamente com os conhecimentos obtidos a partir da revisão bibliográfica viabilizarão identificar a melhor alternativa para a solução do problema. Por fim ocorrerá a implementação da aplicação.

### 1.1 DELIMITAÇÃO DO TEMA

O presente trabalho visa abordar o tema de otimização de grades horárias escolares, tendo em mente especificamente as necessidades de instituições brasileiras de ensino fundamental e médio.

### 1.2 PROBLEMAS E PREMISSAS

Como mencionado anteriormente, as instituições de ensino necessitam de constantes atualizações de suas grades horárias. Este processo envolve muitas variáveis, e impacta diretamente alunos, professores e gestores. Devido a esta complexidade, o problema já é conhecido na literatura como *school timetabling problem*, sendo dividido nos tipos *exam timetabling*, *course timetabling* e *high school timetabling* ([TAN et al., 2021](#)). Este trabalho tem como foco esta

última variação, que, segundo os mesmos autores, é definido em termos da disponibilidade dos professores, número de salas, número de aulas por professor em cada sala e restrições.

## 1.3 OBJETIVOS

### 1.3.1 Objetivo Geral

Desenvolver uma aplicação que automatize e simplifique o processo de criação de grades horárias escolares, otimizando diferentes características de qualidade destas, fazendo com que tais sejam livres de conflitos e atendam a restrições impostas pelas instituições de ensino.

### 1.3.2 Objetivos Específicos

1. Compreender quais são as reais necessidades das escolas quanto ao planejamento de grades horárias
2. Implementar métricas de qualidade para os horários, que refletem se as necessidades são atendidas com êxito
3. Aplicar um algoritmo de otimização na geração de grades que evite conflitos, atenda restrições de horários dos docentes, minimize a quantidade de janelas nas aulas dos professores, faça agrupamentos formando aulas duplas e evite excessos da mesma aula
4. Proporcionar uma experiência do usuário simples e agradável durante o uso da aplicação

## 1.4 ESTRUTURA ORGANIZACIONAL

O trabalho em questão está estruturado da seguinte forma: o capítulo 1 traz a introdução sobre o tema, uma visão geral sobre a pesquisa e seus objetivos; no capítulo 2 será desenvolvida a fundamentação teórica, listando todas as fontes de conhecimento consultadas para para o desenvolvimento do projeto; a forma de implementação do projeto, seus materiais e métodos serão discutidos no capítulo ??; o capítulo ?? apresentará o cronograma das atividades planejadas e por fim, o capítulo 4 trará as considerações finais.



## 2 REVISÃO DE LITERATURA

Este capítulo tem como objetivo explicar o problema de *timetabling* e suas especificidades no caso de grades horárias de ensino médio, assim como a técnica de otimização *Simulated Annealing* e sua aplicação ao problema.

### 2.1 TIMETABLING PROBLEM

O problema de *timetabling* ou alocação de horários é definido em termos de quatro conjuntos, sendo estes horários, recursos, encontros e restrições. O problema consiste em associar os encontros desejados aos horários, utilizando os recursos disponíveis e minimizando as violações das restrições.

É evidente que existem inúmeras tarefas e indústrias que dependem da alocação de horários, portanto é necessário especializar o problema. Uma das variações deste problema é o *school timetabling problem*, que segundo Tan et al. (2021) subdivide-se em *exam timetabling*, *course timetabling* e *high school timetabling*. Esta última subdivisão, relacionada a escolas de ensino médio, será o foco deste trabalho.

#### 2.1.1 Highschool Timetabling Problem

No caso específico da alocação de horários para escolas de ensino médio, o problema é definido em termos da disponibilidade dos professores, número de salas, número de aulas por professor em cada sala e restrições. Adequando estas necessidades à formulação do *timetabling problem*, os encontros que desejamos alocar são entre os professores e determinada sala de aula, dados os horários de aula em que a escola opera, visando a utilização dos recursos disponíveis, como salas de aula ou laboratórios, minimizando a violação de restrições associadas aos professores e recursos (TAN et al., 2021).

Segundo Abramson (1991), o problema envolve agendar aulas, professores e salas de aula de tal forma que nenhum professor, turma ou sala de aula seja utilizado mais de uma vez em determinado horário. Situações em que ocorra este agendamento de recursos duplicados serão tratadas como “conflitos” neste trabalho.

Levando em consideração a necessidade de agendar diversas aulas evitando conflitos, e o atendimento de restrições variadas, o *highschool timetabling problem* é um problema de otimização multiobjetivo, que segundo Cooper e Kingston (1995) é da classe NP-completo.

Considerando a natureza complexa do problema, este trabalho propõe desenvolver e analisar a eficiência de um algoritmo de *Simulated Annealing*, assim como implementar uma interface para interação com este.

### 2.2 NOVO ENSINO MÉDIO

O Novo Ensino Médio consiste na atualização das matrizes curriculares das salas desta etapa do sistema educacional (1º, 2º e 3º anos). Formulado através da Lei nº 13.415/2017 (BRASIL, 2017), o novo modelo traz ainda mais desafios para a tarefa de organização de grades horárias escolares.

O principal destes novos desafios no contexto da criação de grades horárias é o conceito dos Itinerários Formativos, conjuntos de atividades que os estudantes podem escolher realizar durante o ensino médio, como disciplinas, projetos, oficinas, entre outros.

No caso de grandes instituições de ensino, é possível que seja viável oferecer essas atividades optativas através da criação de turmas específicas, entretanto, em escolas de pequeno porte, a adição destas turmas e quantidade relativamente pequena de estudantes por turma pode facilmente desgastar os recursos da instituição.

Tendo estas limitações em mente, observou-se que uma estratégia aplicada por escolas de porte menor é a oferta de disciplinas optativas (Itinerários Formativos) em horários simultâneos em diferentes salas, nos períodos normais de ensino. No momento em que essas aulas acontecem, cada aluno das duas turmas se locomove para a sala adequada onde a aula que escolheu será ministrada, e com isso a escola não precisa alocar salas, professores ou horários adicionais.

Evidentemente, alocar horários simultâneos para determinadas aulas dificulta ainda mais o planejamento do horário escolar, sendo assim mais uma questão que pode ser auxiliada por um sistema de otimização da grade.

### 2.3 SIMULATED ANNEALING

Diversos problemas podem ser resolvidos imitando o que ocorre em fenômenos físicos ou naturais. Segundo [Laarhoven \(1987\)](#), *Simulated Annealing* é uma meta-heurística que se inspira no processo de recozimento na área da metalurgia. Este processo tem como objetivo reduzir as tensões internas de determinado material, da seguinte forma:

1. O material inicia com uma alta temperatura, com seus átomos desordenados e livres para vibrarem e se deslocarem
2. O sistema é resfriado gradualmente, fazendo com que os átomos encontrem posições cada vez mais estáveis. Em outras palavras, conforme a temperatura diminui, torna-se cada vez mais improvável um átomo se deslocar para uma posição menos estável
3. Por fim, atinge-se certa temperatura em que não ocorrem mais mudanças significativas no material.

Esta meta-heurística foi escolhida para o desenvolvimento deste trabalho considerando seu potencial verificado durante a revisão de literatura, e a clareza dos paralelos que podem ser realizados com o processo de têmpera real: os átomos representam os professores e aulas, que podem ser deslocados na grade horária até que atinjam uma posição estável, ou seja, que viole o mínimo possível de restrições do problema.

### 3 DESENVOLVIMENTO

Este capítulo tem como objetivo evidenciar todas as atividades executadas de acordo com os incrementos descritos na seção ???. As próximas seções serão dedicadas aos diferentes incrementos.

#### 3.1 OTIMIZADOR INICIAL

A primeira parte do projeto desenvolvida foi a versão inicial do otimizador, cuja tarefa era gerar uma grade horária válida, evitando conflitos, ou seja, professores alocados para mais uma turma ao mesmo tempo. Para realizar esta tarefa, aplicaram-se os conceitos de simulated annealing, originando o otimizador representado pelo algoritmo 1.

---

##### Algoritmo 1: Otimizador de grades inicial

---

**Input:** Lista de professores  $LP$ , lista de turmas  $LT$ , matriz de aulas por professor por turma  $MA$ , temperatura inicial  $TI$ , Taxa de resfriamento  $TR$

**Output:** Grade horária de professores otimizada

$temperatura \leftarrow TI$

$grade \leftarrow \text{CriaGradelInicial}(LP, LT, MA)$

$minConflitos \leftarrow \text{NumeroConflitos}(grade)$

**while** condição de parada não atingida **do**

**for** passo = 0 até numeroPassos **do**

$turma \leftarrow \text{EscolheTurmaAleatoria}()$

$linhas \leftarrow \text{EscolheHorariosAleatoriosValidos}(sala)$

$delta \leftarrow \text{CalculaDelta}(sala, linhas)$

$probabilidade \leftarrow e^{-delta/temperatura}$

$valorAceite \leftarrow \text{Aleatorio}(0, 1)$

**if**  $delta < 0$  ou  $probabilidade \geq valorAceite$  **then**

$\text{PermutaProfessores}(sala, linha1, linha2)$

**if**  $\text{NumeroConflitos}(grade) < minConflitos$  **then**

$\text{Imprime}(grade)$

$minConflitos \leftarrow \text{NumeroConflitos}(grade)$

**end**

**end**

**end**

$temperatura \leftarrow temperatura * TR$

**end**

---

No algoritmo 1, o valor da taxa de resfriamento é de 0,99, sendo a temperatura inicial e o número de passos por iteração escolhidos empiricamente. Além disso, durante o desenvolvimento deste primeiro incremento, utilizou-se como condição de parada o esgotamento da temperatura, ou seja, o algoritmo finaliza sua execução assim que a temperatura atinge um valor próximo de zero, quando não ocorrem mais permutações de professores.

Explicando melhor o algoritmo, o método "CriaGradelInicial" gera uma matriz com os as turmas e números de aulas de cada professor alocados corretamente. Esta grade inicial provavelmente possui inúmeros conflitos, portanto são aplicados os passos de otimização.

Para cada passo de otimização, são escolhidas aleatoriamente uma turma e duas linhas (posições) da grade horária. Com estas informações, é calculada a variação do número de conflitos que a permutação dos professores nas linhas escolhidas ocasionaria. De acordo com este valor de variação, é determinado se a troca dos professores deve ou não ser realizada: uma troca que diminua o número de conflitos sempre é aceita, enquanto uma troca que aumenta o número de conflitos pode ser aceita probabilisticamente, de acordo com o valor da temperatura na iteração atual.

### 3.2 PESOS E RESTRIÇÕES

Apesar a importância da resolução de conflitos, existem diversas outras nuances durante o planejamento das grades horárias que precisam ser levadas em conta para que as grades produzidas sejam aplicáveis na prática.

Para possibilitar a otimização simultânea de várias características das grades, optou-se por utilizar um sistema de métricas, cada qual mensura quantitativamente determinada característica do horário, e contém um peso que define a sua importância para o horário como um todo. Com estas métricas, para obter o custo de determinada solução, ou seja, a representação da quantidade de problemas que apresenta, basta tomar a média ponderada das métricas.

Adicionalmente, as métricas podem ser rígidas ou suaves. As métricas rígidas precisam ser perfeitamente atendidas para que a grade horária seja considerada viável, enquanto as métricas suaves promovem a melhoria da qualidade, mas não necessariamente precisam ser perfeitamente atendidas.

As próximas seções terão como objetivo explicar cada uma destas métricas e os desafios associados. A [Subseção 3.2.8](#) entrará em mais detalhes sobre como as grades horárias passaram a ser salvas após a implementação das métricas.

#### 3.2.1 Agrupamento de aulas

Observando grades horárias escolares, é possível notar que existe um desejo de realizar agrupamentos, formando aulas duplas. Isso aumenta a produtividade das aulas, à medida que diminui a troca e deslocamento de professores entre as salas. Para produzir estes agrupamentos, foram adicionadas algumas métricas que fazem o otimizador penalizar:

1. Aulas separadas;
2. Aulas desagrupadas;
3. Excessos de aulas iguais para determinada turma em um dia;
4. Dias com todas aulas planejadas distintas para determinada turma;

Quadro 1 – Exemplo de dia com aulas separadas.

Aula	Professor	Matéria
1	Marcos	Matemática
2	Fábio	Física
3	Marcos	Matemática
4	Luciana	Português
5	Luciana	Português
6	Luciana	Português

Fonte: Autoria própria

Observando o quadro 1, é possível notar a presença das situações comentadas:

- Aulas separadas: 1, 2 e 3, visto que não estão em nenhum agrupamento;
- Aulas desagrupadas: 1 e 3, pois consistem em múltiplas aulas iguais, no mesmo dia, que não foram agrupadas;
- Excessos de aulas: 4, 5, 6, pois consiste em uma aula tripla, considerada anti-didática no Ensino Médio

Por fim, a situação de dias com todas aulas diferentes, que serão referidos como "Dias fragmentados" neste trabalho, pode ser observada no quadro 2.

Quadro 2 – Exemplo de dia fragmentado.

Aula	Professor	Matéria
1	Marcos	Matemática
2	Fábio	Física
3	Luciana	Português
4	Roberto	Geografia
5	Renato	História


Fonte: Autoria própria

### 3.2.2 Constantes

Durante o desenvolvimento, concebeu-se o conceito de "Aulas constantes", como aulas que absolutamente devem ser alocadas em determinada posição da grade horária. Isto é útil para guiar o otimizador rumo a uma solução desejada, quando já são conhecidas algumas aulas que devem ser fixas.

Como exemplo de caso de uso, uma escola com seis aulas diárias pode ter alguns dias da semana com menos aulas, e pode ser interessante definir explicitamente quais dias devem ter a última aula da grade horária não alocada (vazia). A figura 1 mostra um exemplo de configuração de aulas constantes para determinada grade horária.

Figura 1 – Aulas constantes

Constantes 						
Dia	Aula	6 Ano	1 EM	7 Ano	8 Ano	9 Ano
Segunda-feira	1	Verônica				
	2	Verônica				
	3					
	4					
	5					
	6					
Terça-feira	1	Luciana	Adriana			
	2	Luciana	Adriana			
	3					
	4					
	5					
	6					
Quarta-feira	1					
	2					
	3					
	4					
	5					
	6					
Quinta-feira	1					
	2					
	3					
	4					
	5					
	6					
Sexta-feira	1					
	2					
	3					
	4					
	5					
	6					

Fonte: Autor

Considerando a natureza absoluta das aulas constantes, estas não são consideradas um métrica de qualidade, e não possuem um peso próprio. Em vez disso, no método "GeraGadelnicial" do algoritmo 1, o otimizador já aloca as aulas constantes nas posições desejadas, e a função "EscolheHorariosAleatoriosValidos" não seleciona posições que estejam alocadas com aulas constantes.

### 3.2.3 Restrições

As restrições representam o oposto das aulas constantes: posições em que determinadas aulas não devem ser alocadas. Implementaram-se no otimizador restrições suaves e rígidas, cada uma podendo também receber um peso customizado. Dessa forma, é possível configurar

o otimizador para nunca alocar determinada aula em certa posição da grade, ou simplesmente evitar isso.

Como exemplo de uso dessa funcionalidade, a figura 2 demonstra uma configuração de restrições para determinado professor que não pode ser alocado nas duas últimas aulas de qualquer dia da grade horária.

Figura 2 – Restrições

**Restrições** ☒ Proibir ☐ Evitar

Dia	Aula	6 Ano	1 EM	7 Ano	8 Ano	9 Ano
Segunda-feira	1					
	2					
	3					
	4					
	5					
	6					
Terça-feira	1					
	2					
	3					
	4					
	5					
	6					
Quarta-feira	1					
	2					
	3					
	4					
	5					
	6					
Quinta-feira	1					
	2					
	3					
	4					
	5					
	6					
Sexta-feira	1					
	2					
	3					
	4					
	5					
	6					

Fonte: Autor

No caso das restrições, a métrica associada mensura a quantidade de restrições violadas, ou seja, posições em que determinada aula foi alocada, mas não deveria ter sido.

### 3.2.4 Regiões

O conceito de regiões foi concebido como uma forma de proporcionar ainda mais controle ao usuário, sobre o posicionamento das aulas na grade horária. As regiões consistem em um grupo arbitrário de posições da grade horária (uma região), associado a uma regra relacionada a uma quantidade de aulas. Com as regiões, é possível determinar mínimos, máximos ou quantidades exatas de aulas que devem ser alocadas em certas posições da grade horária.

Como exemplo de uso das regiões, a figura 3 demonstra uma configuração utilizada para assegurar que em todos os dias da grade horária, o professor tenha alguma aula alocada no primeiro horário.



Figura 3 – Exemplo de região

Região (Verônica - Exatamente 5 aulas) 🗑

Dia	Aula	6 Ano	1 EM	7 Ano	8 Ano	9 Ano
Segunda-feira	1					
	2					
	3					
	4					
	5					
	6					
Terça-feira	1					
	2					
	3					
	4					
	5					
	6					
Quarta-feira	1					
	2					
	3					
	4					
	5					
	6					
Quinta-feira	1					
	2					
	3					
	4					
	5					
	6					
Sexta-feira	1					
	2					
	3					
	4					
	5					
	6					

Fonte: Autor

A região configurada na 3 pode ser interpretada da seguinte forma: "Verônica" deve ter exatamente cinco aulas alocadas dentro das posições marcadas pela cor azul. Como o otimizador não permite conflitos, cada uma das cinco aulas deverá ser alocada em um dos diferentes dias, garantindo que "Verônica" terá uma aula agendada no primeiro horário de todos os dias.

Neste caso, a métrica associada mensura o erro das regiões, ou seja, a diferença entre



O efeito da configuração do grupo de alinhamento da figura 4, pode ser observado na grade horária final gerada na figura 5, com as aulas corretamente alocadas em horários simultâneos.

Figura 5 – Efeito de grupo de alinhamento

Dia	Aula	6 Ano		7 Ano	
Segunda-feira	1	Jack	Português	Renata	História
	2	Jack	Português	Renata	História
	3	Renata	História	Verônica	Geografia
	4	Verônica	Geografia	Adriana	Ciências
	5	Verônica	Geografia	Adriana	Ciências
	6				
Terça-feira	1	Jack	Português	Jéssica	Artes
	2	Jack	Português	Jéssica	Artes
	3	Jéssica	Inglês	Rosa	Matemática
	4	Rosa	Matemática	Luciana	Of. Texto
	5	Rosa	Matemática	Beatriz	ED. Física
	6	Beatriz	ED. Física		
Quarta-feira	1	Renata	História	Jack	Português
	2	Renata	História	Jack	Português
	3	Rosa	Matemática	Renata	História
	4	Luciana	Of. Texto	Rosa	Matemática
	5	Verônica	Geografia	Adriana	Ciências
	6				
Quinta-feira	1	Adriana	Ciências	Jack	Espanhol
	2	Adriana	Ciências	Jack	Português
	3	Jack	Português	Jéssica	Filosofia
	4	Jéssica	Inglês	Rosa	Matemática
	5	Jéssica	Filosofia	Verônica	Geografia
	6			Verônica	Geografia
Sexta-feira	1	Rosa	Matemática	Jéssica	Inglês
	2	Rosa	Matemática	Jéssica	Inglês
	3	Jack	Espanhol	Rosa	Matemática
	4	Adriana	Ciências	Rosa	Matemática
	5	Cristiane	Artes	Jack	Português
	6	Cristiane	Artes	Jack	Português

Fonte: Autor

Implementaram-se duas métricas de qualidade relacionadas aos grupos de alinhamento, uma que mensura a quantidade de grupos, que deve ser minimizada; e outra que mede a quantidade de aulas "desalinhadas", ou seja, aulas que deveriam ser agendadas no mesmo horário, mas que não foram.

### 3.2.6 Janelas

As janelas consistem em situações em que um professor tem um horário sem aulas agendadas em sua jornada de trabalho, ficando ocioso na escola. O quadro 3 exemplifica esta situação.

Quadro 3 – Exemplo de dia com janela.

<b>Aula</b>	<b>6ºAno</b>	<b>7ºAno</b>	<b>8ºAno</b>
1	Jéssica	Rosa	Adriana
2	Jéssica	Rosa	Adriana
3	Fábio	Jéssica	Rosa
4	Adriana	Jéssica	Rosa
5	Beatriz	Adriana	Jéssica

Fonte: Autoria própria

No quadro 3, a professora Adriana tem uma janela na terceira aula, visto que tem aulas alocadas antes e depois (aulas 1, 2, 4 e 5). Em contrapartida, apesar de não ter nenhuma aula planejada no quinto horário, Rosa não tem janelas neste exemplo, pois pode encerrar seu expediente na quarta aula.

A métrica de janelas simplesmente mede, para cada professor, o número de horários na grade em que existem janelas.

### 3.2.7 Preferências

### 3.2.8 Armazenamento de soluções

Com a adição das métricas, o critério para considerar uma grade horária como uma solução válida ficou mais estrito, e em muitos casos tornou-se impossível obter uma grade que atenda perfeitamente todas as nuances. Tendo isto em mente, foi necessário implementar uma funcionalidade de armazenamento de grades horárias um pouco mais detalhada. O resultado disso, pode ser visto no algoritmo 2.

**Algoritmo 2:** Otimizador com persistência de grades horárias

**Input:** Lista de professores  $LP$ , lista de turmas  $LT$ , matriz de aulas por professor por turma  $MA$ , temperatura inicial  $TI$ , Taxa de resfriamento  $TR$

**Output:** Grade horária de professores otimizada

$temperatura \leftarrow TI$

$grade \leftarrow \text{CriaGradeInicial}(LP, LT, MA)$

$iteracoesSemAlteracao \leftarrow 0$

$solucoes \leftarrow$  lista vazia

**while** condição de parada não atingida **do**

$deltaTotal \leftarrow 0$

**for** passo = 0 até numeroPassos **do**

$turma \leftarrow grade.EscolheTurmaAleatoria()$

$linhas \leftarrow grade.EscolheHorariosAleatoriosValidos(sala)$

$delta \leftarrow grade.CalculaDelta(sala, linhas)$

$probabilidade \leftarrow e^{-delta/temperatura}$

$valorAceite \leftarrow \text{Aleatorio}(0, 1)$

**if**  $delta < 0$  ou  $probabilidade \geq valorAceite$  **then**

$grade.PermutaProfessores(sala, linha1, linha2)$

$deltaTotal \leftarrow deltaTotal + delta$

**if** grade não existe na lista de soluções **then**

                insere grade na lista de soluções

                limita lista de soluções às 100 melhores grades

**end**

**end**

**end**

**if**  $delta = 0$  **then**

$iteracoesSemAlteracao \leftarrow iteracoesSemAlteracao + 1$

**else**

$iteracoesSemAlteracao \leftarrow 0$

**end**

**if**  $iteracoesSemAlteracao \geq 15$  **then**

$salvaGradesRelevantes()$

        apaga lista de soluções

$temperatura \leftarrow TI$

$iteracoesSemAlteracao \leftarrow 0$

**end**

$temperatura \leftarrow temperatura * TR$

**end**

Considerando as diversas métricas de qualidade que as grades horárias passaram a ter, o método "salvaGradesRelevantes" do algoritmo 2 salva, para cada métrica escolhida, as 5 grades que tiveram a melhor pontuação em cada métrica. A princípio, optou-se por utilizar as métricas de qualidade geral, janelas, agrupamento de aulas e número de preferências resolvidas, mas este método pode ser expandido para qualquer uma das métricas implementadas no sistema.

### 3.3 SERVIDOR E BANCO DE DADOS

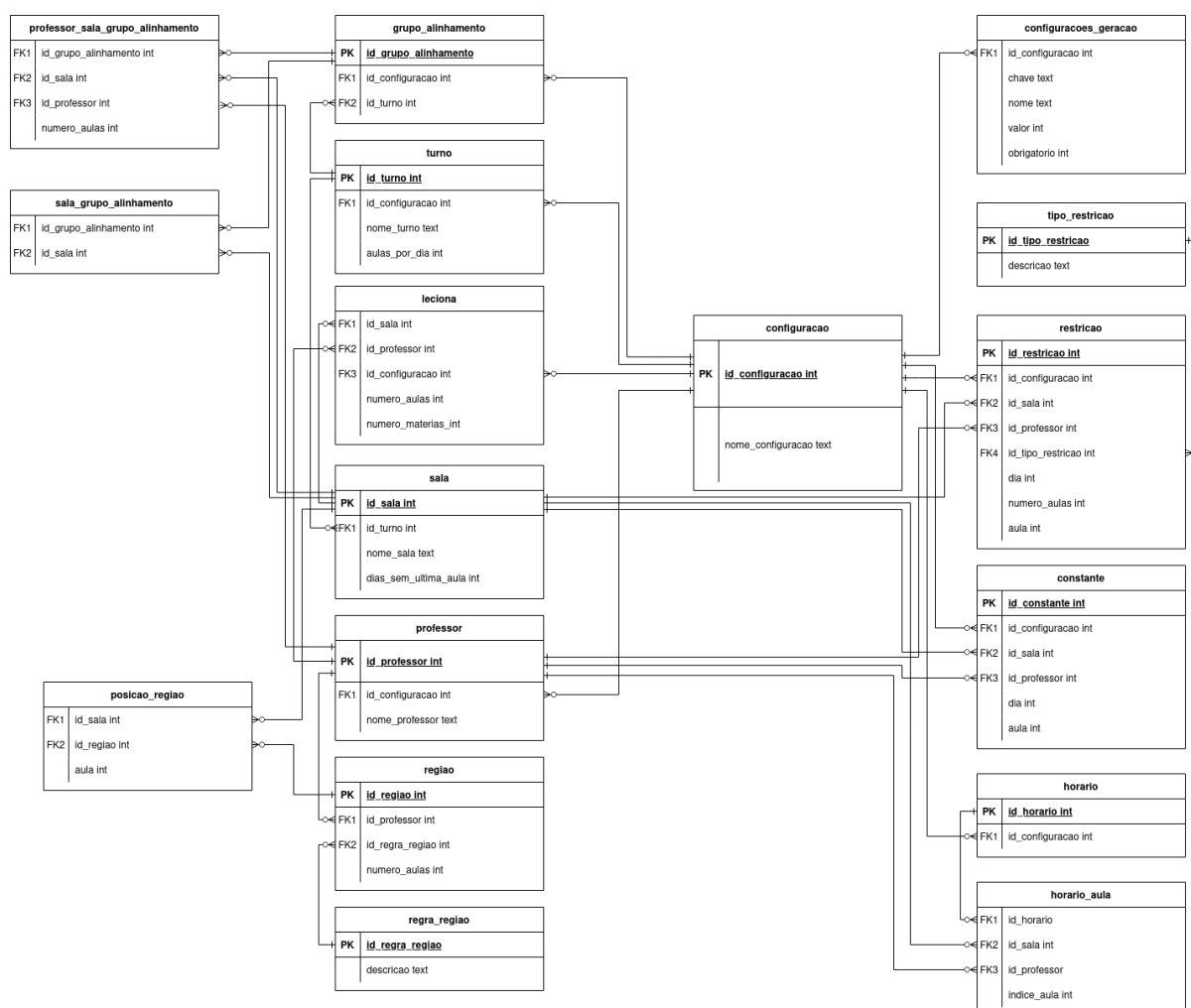
A aplicação do lado do servidor é responsável por receber as requisições da interface, persistir as configurações no banco de dados e realizar a comunicação com o otimizador, a fim de produzir e armazenar as grades horárias.

Para o desenvolvimento deste componente, optou-se pelo *framework Express.js*, a ser executado na plataforma *Node.js*, devido à simplicidade de implementação que estas tecnologias proporcionam. Em relação ao banco de dados, será utilizado o sistema de gerenciamento de banco de dados *PostgreSQL*, devido à sua robustez.

Conforme as premissas do problema sendo tratado, a modelagem do banco de dados é centrada na entidade "Configuração", que agrupa as configurações de determinada instituição de ensino para a geração de suas grades horárias. Cada uma dessas entidades tem turnos, salas, professores, e as configurações de quantas aulas cada professor deve ministrar em cada sala, e as respectivas restrições.

A modelagem comentada está representada na [Figura 6](#):

Figura 6 – Modelo Entidade-Relacionamento



Fonte: Autor

Dentre as entidades mostradas no modelo entidade-relacionamento, vale ressaltar a importância da entidade “Grupo Alinhamento”. Esta será utilizada para configurar aulas que devam acontecer simultaneamente, a fim de resolver o desafio dos itinerários formativos do Novo Ensino Médio, conforme exposto na seção 2.2.

Para validar a modelagem do banco de dados, realizaram-se inserções de informações de exemplo nas diferentes tabelas. A Figura 7 mostra algumas dessas inserções e os vínculos instituídos pelos identificadores escolhidos.

Figura 7 – Consultas de validação da modelagem

```

INSERT INTO configuracao (id configuracao, nome configuracao) VALUES(1, 'Leme');
INSERT INTO professor (id professor, id configuracao, nome professor) VALUES(1, 1, 'Fabio');
INSERT INTO turno (id turno, id configuracao, nome turno, aulas por dia) VALUES(1, 1, 'Manhã', 6);
INSERT INTO sala (id sala, id turno, nome sala, dias sem ultima aula) VALUES(1, 1, '6 Ano', 3);
INSERT INTO leciona (id sala, id professor, id configuracao, numero aulas, numero materias) VALUES(1, 1, 1, 5, 1);
INSERT INTO horario (id configuracao, id horario) VALUES(1, 1);
INSERT INTO horario aula (id sala, id professor, id horario, index aula) VALUES(1, 1, 1, 0);
INSERT INTO constante (id constante, id sala, id professor, id configuracao, dia, aula) VALUES(1, 1, 1, 1, 0, 0);
INSERT INTO tipo_restricao (id tipo restricao, descricao, nome_xml) VALUES(0, 'Proibir', 'normal');
INSERT INTO restricao (id configuracao, id sala, id professor, tipo, dia, aula, num aulas) VALUES(1, 1, 1, 0, 0, 0, NULL);
INSERT INTO configuracoes_geracao (chave_xml, nome, valor, tipo, id configuracao, obrigatorio) VALUES('Conflitos', NULL, 60, 'setting', 1, 1);
INSERT INTO grupo_alinhamento (id grupo alinhamento, id configuracao, id turno) VALUES(1, 1, 1);
INSERT INTO sala_grupo_alinhamento (id sala, id grupo alinhamento) VALUES(1, 1);
INSERT INTO professor_sala_grupo_alinhamento (id professor, id sala, id grupo alinhamento, numero aulas) VALUES(1, 1, 1, 2);
INSERT INTO regiao (id regiao, id professor, num aulas, regra) VALUES(1, 1, 1, 0);
INSERT INTO posicao_regiao (id regiao, id sala, aula) VALUES(1, 1, 2);

```

Name	Value
Queries	16
Updated Rows	16
Execute time (ms)	41
Fetch time (ms)	0
Total time (ms)	41
Finish time	2023-05-20 15:13:53.843

Fonte: Autor

Validou-se também o armazenamento das grades horárias no banco de dados. A Figura 8 traz um exemplo de consulta de uma grade horária com sete salas:

Figura 8 – Consulta SQL de grade horária com sete salas

```

SELECT
CASE (a.n / 6)
WHEN 0 THEN 'Segunda'
WHEN 1 THEN 'Terça'
WHEN 2 THEN 'Quarta'
WHEN 3 THEN 'Quinta'
WHEN 4 THEN 'Sexta'
ELSE NULL
END AS "DiaSemana",
(a.n % 6) + 1 AS "Aula",
p1.nome_professor AS "6Ano",
p2.nome_professor AS "7Ano",
p3.nome_professor AS "8Ano",
p4.nome_professor AS "9Ano",
p5.nome_professor AS "1EM",
p6.nome_professor AS "2EM",
p7.nome_professor AS "3EM"
FROM generate_series(0, 29) AS a(n)
LEFT JOIN horario_aula ha1 ON ha1.id_sala = 2 AND ha1.index_aula = a.n AND ha1.id_horario = 10848
LEFT JOIN professor p1 ON p1.id_professor = ha1.id_professor
LEFT JOIN horario_aula ha2 ON ha2.id_sala = 3 AND ha2.index_aula = a.n AND ha2.id_horario = 10848
LEFT JOIN professor p2 ON p2.id_professor = ha2.id_professor
LEFT JOIN horario_aula ha3 ON ha3.id_sala = 4 AND ha3.index_aula = a.n AND ha3.id_horario = 10848
LEFT JOIN professor p3 ON p3.id_professor = ha3.id_professor
LEFT JOIN horario_aula ha4 ON ha4.id_sala = 5 AND ha4.index_aula = a.n AND ha4.id_horario = 10848
LEFT JOIN professor p4 ON p4.id_professor = ha4.id_professor
LEFT JOIN horario_aula ha5 ON ha5.id_sala = 6 AND ha5.index_aula = a.n AND ha5.id_horario = 10848
LEFT JOIN professor p5 ON p5.id_professor = ha5.id_professor
LEFT JOIN horario_aula ha6 ON ha6.id_sala = 7 AND ha6.index_aula = a.n AND ha6.id_horario = 10848
LEFT JOIN professor p6 ON p6.id_professor = ha6.id_professor
LEFT JOIN horario_aula ha7 ON ha7.id_sala = 8 AND ha7.index_aula = a.n AND ha7.id_horario = 10848
LEFT JOIN professor p7 ON p7.id_professor = ha7.id_professor
ORDER BY a.n;

```

Fonte: Autor

A consulta anterior traz como resultado a tabela visível na Figura 9, com linhas e colunas correspondentes a horários de aulas e salas respectivamente.

### 3.4 INTERFACE

A interface web será responsável pela interação do usuário final com a aplicação. Portanto, deverá implementar funcionalidades que possibilitem a configuração das restrições e



Figura 9 – Resultado da consulta de uma grade no banco de dados

DiaSemana	Aula	asc 6Ano	asc 7Ano	asc 8Ano	asc 9Ano	asc 1EM	asc 2EM	asc 3EM
Segunda	1	Renata	Rosa	Cristiane	Luciana	Adriana	Jéssica	Glaucia
Segunda	2	Renata	Rosa	Cristiane	Luciana	Glaucia	Jéssica	Adriana
Segunda	3	Beatriz	Renata	Rosa	Jéssica	Glaucia	Luciana	Adriana
Segunda	4	Cristiane	Renata	Rosa	Adriana	Jéssica	Glaucia	Luciana
Segunda	5	Cristiane	Adriana	Beatriz	Adriana	Jéssica	Glaucia	Luciana
Segunda	6	[NULL]	Beatriz	[NULL]	[NULL]	Luciana	Jéssica	Adriana
Terça	1	Rosa	Luciano	Luciana	Renata	Fabio	Adriana	Jéssica
Terça	2	Rosa	Adriana	Luciana	Fabio	Luciano	Glaucia	Jéssica
Terça	3	Luciano	Adriana	Renata	Fabio	Jéssica	Glaucia	Luciana
Terça	4	Luciano	Renata	Adriana	Jéssica	Fabio	Luciana	Glaucia
Terça	5	Adriana	Jéssica	Luciana	Beatriz	Fabio	Luciano	Glaucia
Terça	6	Adriana	[NULL]	[NULL]	[NULL]	Glaucia	Luciano	Jéssica
Quarta	1	Luciano	Jack	Rosa	Luciana	Adriana	Renata	Jéssica
Quarta	2	Rosa	Jack	Jéssica	Luciano	Adriana	Renata	Luciana
Quarta	3	Jack	Luciano	Jéssica	Fabio	Luciana	Adriana	Renata
Quarta	4	Jéssica	Luciano	Luciana	Fabio	Jack	Adriana	Renata
Quarta	5	Jéssica	Jack	Luciano	Luciana	Glaucia	Fabio	Adriana
Quarta	6	[NULL]	[NULL]	Luciano	Luciana	Glaucia	Jack	Adriana
Quinta	1	Luciana	Jack	Rosa	Renata	Jéssica	Fabio	Adriana
Quinta	2	Jéssica	Jack	Rosa	Renata	Adriana	Luciana	Fabio
Quinta	3	Jack	Rosa	Renata	Jéssica	Fabio	Adriana	Luciana
Quinta	4	Jack	Rosa	Renata	Jéssica	Luciana	Adriana	Fabio
Quinta	5	Adriana	Jéssica	Luciana	Jack	Luciano	Glaucia	Fabio
Quinta	6	[NULL]	[NULL]	Jack	Luciana	Luciano	Adriana	Glaucia
Sexta	1	Jack	Rosa	Luciana	Jéssica	Adriana	Renata	Luciano
Sexta	2	Renata	Jack	Jéssica	Fabio	Adriana	Luciana	Luciano
Sexta	3	Rosa	Jéssica	Luciano	Adriana	Renata	Luciana	Fabio
Sexta	4	Rosa	Luciana	Adriana	Luciano	Renata	Jéssica	Jack
Sexta	5	Jack	Jéssica	Adriana	Luciano	Luciana	Fabio	Glaucia
Sexta	6	Jack	Jéssica	[NULL]	[NULL]	Luciana	Fabio	Luciano

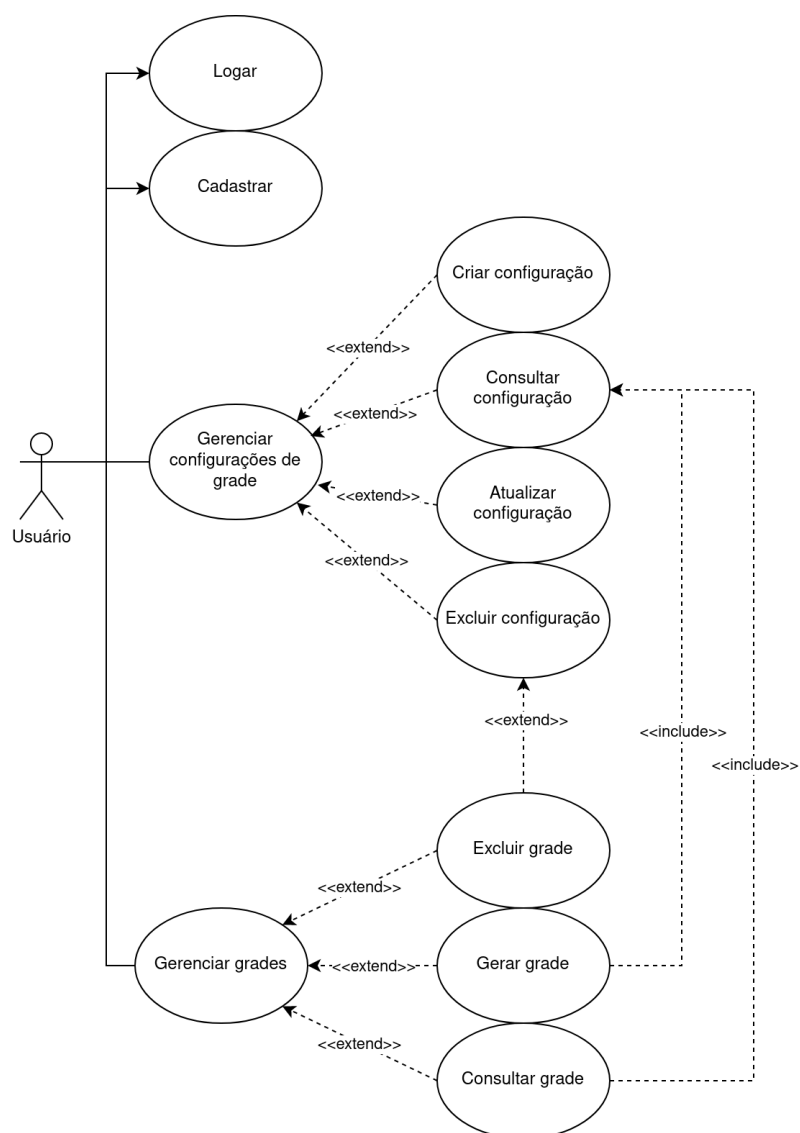
Fonte: Autor

características das grades horárias a serem geradas, além de mostrar ou exportar tais grades após a geração.

Para o desenvolvimento deste componente, optou-se pelo framework *Vue.js*, devido à riqueza de seu ecossistema de desenvolvimento *frontend* e familiaridade do graduando com este. Complementando este framework, será utilizado também o *Vuetify.js*, a fim de facilitar e padronizar o design da aplicação.

A Figura 10 mostra os casos de uso que a interface *web* será responsável por disponibilizar:

Figura 10 – Diagrama de Casos de Uso



Fonte: Autor

Desenvolveram-se alguns protótipos iniciais das telas necessárias na aplicação. Primeiramente, a [Figura 11](#) mostra a tela de listagem de configurações de grade. Como comentado anteriormente, alguns dos casos de uso da aplicação envolvem o gerenciamento de configurações de grades horárias, as quais são listadas nessa tela.

Figura 11 – Tela - Listagem de Configurações de Grade




Horário

CONFIGURAÇÕES

HORÁRIOS

Configurações de horário

NOVA CONFIGURAÇÃO

Nome	Ações
Teste	
Letra	  
LetraMultiTurno	

Fonte: Autor

A tela mostrada na [Figura 12](#) é responsável por permitir que o usuário cadastre os professores da escola. Nessa tela é possível notar que a aplicação foi estruturada seguindo uma noção de etapas de configuração até a geração da grade horária final. No topo da tela, a etapa "Estrutura da Escola" encontra-se selecionada.

Figura 12 – Tela - Estrutura da Escola - Professores

Horário

CONFIGURAÇÕESHORÁRIOS

1Estrutura da Escola

2Aulas/Matérias

3Aulas constantes

4Restrições























5Regras de preenchimento

6Mínimos/Máximos

7Horários

Professores

NOVO PROFESSOR

Nome	Ações
Fabio	 
Adriana	 
Luciana	 
Luciano	 
Jack	 
Glauçia	 
Renata	 
Beatriz	 
Rosa	 
Jéssica	 
Cristiane	 

Turnos

NOVO TURNO

Nome	Aulas por dia	Ações
------	---------------	-------

Fonte: Autor

Na Figura 13, tem-se a continuação da tela de estrutura da escola, mostrada na Figura 12. Esta parte da tela é responsável pela configuração das salas e turnos da escola.

Figura 13 – Tela - Estrutura da Escola - Salas e Turnos

Horário

CONFIGURAÇÕES HORÁRIOS

Cristiane

Turnos

NOVO TURNO

Nome	Aulas por dia	Ações
Manhã	5	

Informações do turno: Manhã

Salas

NOVA SALA

Nome	Sala sem última aula	Ações
6 Ano	3	
7 Ano	3	
8 Ano	3	
9 Ano	3	
1 EM	0	
2 EM	0	
3 EM	0	

SALVAR PRÓXIMA ETAPA GERAR DADOS EXEMPLO

Fonte: Autor

Na tela da Figura 14, o usuário pode realizar a configuração de número de aulas que cada professor deve ministrar em cada sala, informação fundamental para a geração das grades horárias.

Figura 14 – Tela - Aulas por professor

Horário

CONFIGURAÇÕES HORÁRIOS

1 Estrutura da Escola 2 Aulas/Matérias 3 Aulas constantes 4 Restrições 5 Regras de preenchimento 6 Mínimos/Máximos 7 Horários

Informações do turno: Manhã

Aulas por professor

Professor	6 Ano	7 Ano	8 Ano	9 Ano	1 EM	2 EM	3 EM	Total
Fabio	0	0	0	5	4	4	4	17
Adriana	4	3	3	3	6	6	6	31
Luciana	1	1	6	6	5	5	5	29
Luciano	3	3	3	3	3	2	3	20
Jack	6	6	1	1	1	1	1	17
Glauco	0	0	0	0	5	5	5	15
Renata	3	3	3	3	2	3	2	19
Betriz	1	1	1	1	0	0	0	4
Rosa	5	5	5	0	0	0	0	15
Jessica	3	5	3	5	4	4	4	28
Cristiane	2	0	2	0	0	0	0	4

Fonte: Autor

A tela da Figura 15 é responsável pela configuração das restrições. Nesta, o usuário pode configurar horários na grade que devem ser evitados ou proibidos para determinado docente.

Figura 15 – Tela - Restrições

Horário

CONFIGURAÇÕES HORÁRIOS

1 Estrutura da Escola 2 Aulas/Matérias 3 Aulas constantes 4 Restrições 5 Regras de preenchimento 6 Mínimos/Máximos 7 Horários

Restrições

Proibir Evitar

Restrições móveis

Manhã Tarde Noite

Salvar

Restrições móveis - Vínculos

NOVO VÍNCULO

Preferências

Fonte: Autor

A última etapa no fluxo da aplicação é representada pela tela da Figura 16. Nesta, o usuário pode requisitar a geração da grade horária utilizando as configurações realizadas nas etapas anteriores, e acessar as grades geradas anteriormente.

Figura 16 – Tela - Horários

Horário

CONFIGURAÇÕES HORÁRIOS

1 Estrutura da Escola 2 Aulas/Matérias 3 Aulas constantes 4 Restrições 5 Regras de preenchimento 6 Mínimos/Máximos 7 Horários

Horário (10832)

Manhã Tarde Noite

Horários Gerados

GERAR

Aulas separadas Restrições não resolvidas Restrições específicas não resolvidas Preferências não resolvidas Assinaturas de aulas digitais Espaços vazios Qualidade Ações

Linhas por página: 10 1/10 de 16

Fonte: Autor

### 3.5 USUÁRIOS E AUTENTICAÇÃO

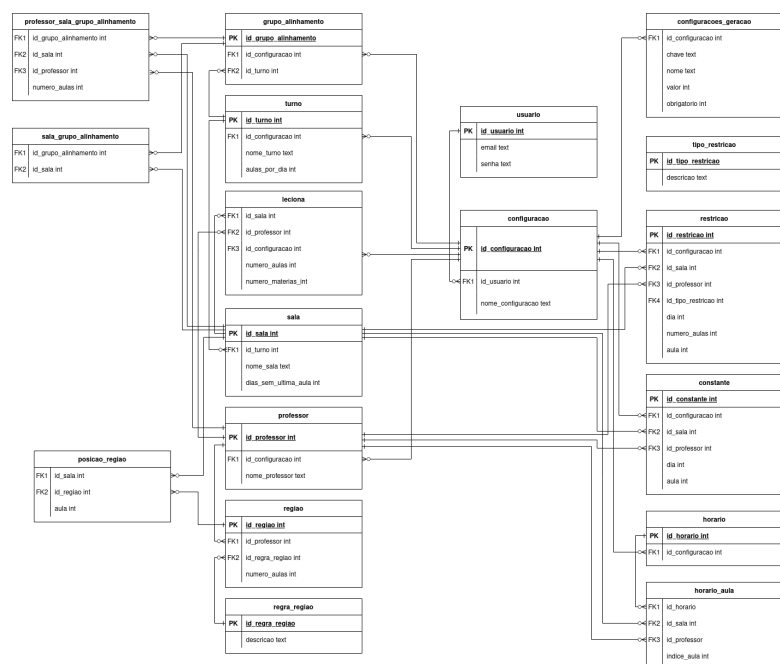
Para controlar a visibilidade de informações no sistema desenvolvido, implementou-se um sistema de usuários. A melhoria consistiu em:

1. Criação de tabela de usuários no banco de dados;
2. Associação da tabela "configuracao" com a tabela de usuários, para que seja possível armazenar o usuário responsável por determinada configuração;
3. Criação da tela de login na interface web;
4. Criação das rotas de cadastro e *login* no servidor
5. Criação do *middleware* de autenticação
6. Criação do *middleware* de validação da configuração

#### 3.5.1 Adaptação do banco de dados

A aplicação dos itens 1 e 2 foi realizada diretamente no banco de dados, através da criação da tabela mencionada e a chave estrangeira possibilitando a associação de cada usuário a múltiplas configurações. Após estas alterações, o diagrama entidade relacionamento do banco de dados passa a ser representado pela figura ??.

Figura 17 – Modelo Entidade-Relacionamento com adição da tabela de usuários



Fonte: Autor

#### 3.5.2 Tela de login

Como pode ser visto na figura 18, desenvolveu-se uma tela simples de *login*, a qual também desempenha a função de cadastro de novos usuários.

Figura 18 – Tela de Acesso

Fonte: Autor

### 3.5.3 Rotas de autenticação

Para a implementação das rotas de cadastro e *login*, utilizou-se além do *framework ExpressJS*, os pacotes *JWT (Json Web Token)* e *bcrypt*. O pacote *JWT* é utilizado para gerar *tokens*, que são enviados para a interface web e podem ser utilizados para autenticar os usuários; já o *bcrypt* é utilizado para gerar e validar as *hashes* das senhas dos usuários, para que nunca sejam armazenadas senhas em texto pleno no banco de dados.

A função utilizada pela nova rota de cadastro pode vista na figura 19. A função "register" realiza validação dos parâmetros, e cria um usuário no banco de dados, caso já não exista outro com o mesmo e-mail. Além disso, a função retorna um *token JWT* para a interface web, para que seja possível verificar a autenticidade das próximas requisições realizadas pelo usuário.

Figura 19 – Método de Cadastro

```
const register = async (req, res) => {
  const { email, senha } = req.body;
  if (email == null || senha == null) {
    return res.status(400).json({ message: "E-mail e senha são obrigatórios" });
  }

  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  if (!emailRegex.test(email)) {
    return res.status(400).json({ message: "E-mail inválido" });
  }

  if (senha.length < 6) {
    return res
      .status(400)
      .json({ message: "A senha deve conter pelo menos 6 caracteres" });
  }

  const resUsuario = await pool.query(
    "select 1 from usuario where email = $1",
    [email]
  );
  if (resUsuario.rows.length > 0) {
    return res.status(400).json({
      message: "Este e-mail já está sendo utilizado por outro usuário",
    });
  }

  const saltRounds = 10;
  const hash = await bcrypt.hash(senha, saltRounds);

  const resUsuarioInserido = await pool.query(
    "insert into usuario (email, senha) values ($1, $2) returning id_usuario",
    [email, hash]
  );
  const id_usuario = resUsuarioInserido.rows[0].id_usuario;

  const secret = process.env.JWT_SECRET;
  const token = jwt.sign({ id_usuario }, secret);
  return res
    .status(200)
    .json({ token, message: "Usuário registrado com sucesso" });
};
```

Fonte: Autor

A função de *login*, visível na figura 20 é similar, realizando validação dos parâmetros, autenticação do usuário através da comparação de senhas utilizando o pacote *bcrypt* e geração de *token* JWT. Vale citar que em ambas as rotas de autenticação, é inserido no *payload* do *token* o identificador do usuário, que posteriormente pode ser utilizado pelos *middlewares* para a realização de validações.



Figura 20 – Método de Login

```
const login = async (req, res) => {
  const { email, senha } = req.body;
  if (email == null || senha == null) {
    return res.status(400).json({ message: "E-mail e senha são obrigatórios" });
  }

  const resUsuario = await pool.query(
    "select id_usuario, senha from usuario where email = $1",
    [email]
  );
  if (resUsuario.rows.length === 0) {
    return res.status(400).json({ message: "Credenciais inválidas" });
  }

  const { senha: hash, id_usuario } = resUsuario.rows[0];
  const correctPassword = await bcrypt.compare(senha, hash);

  if (!correctPassword) {
    return res.status(400).json({ message: "Credenciais inválidas" });
  }

  const secret = process.env.JWT_SECRET;
  const token = jwt.sign({ id_usuario }, secret);

  return res
    .status(200)
    .json({ token, message: "Login realizado com sucesso" });
};
```

Fonte: Autor

### 3.5.4 Middlewares

Foram criados duas funções *middleware* relacionadas aos usuários. A primeira é responsável por verificar que apenas usuários propriamente autenticados tenham acesso aos recursos protegidos do sistema. Como pode ser visto na figura 21, essa verificação é realizada através da verificação da presença e validade de um *token* JWT no *header* "authorization" da requisição.

Figura 21 – Middleware de autenticação

```
const jwt = require("jsonwebtoken");

function authenticateToken(req, res, next) {
  const authHeader = req.headers["authorization"];
  const token = authHeader && authHeader.split(" ")[1];

  if (token == null) return res.sendStatus(401);

  jwt.verify(token, process.env.JWT_SECRET, (err, data) => {
    if (err || data.id_usuario == null) return res.sendStatus(403);
    req.id_usuario = data.id_usuario;
    next();
  });
}

module.exports = authenticateToken;
```

Fonte: Autor

O outro middleware criado tem como objetivo assegurar que um usuário possa consultar apenas as configurações pelas quais seja responsável. Isso é feito utilizando o identificador do usuário, presente no *payload* do *token* JWT, conforme a figura 22. Caso o usuário não tenha um vínculo com a configuração que está tentando acessar, a requisição é bloqueada.

Figura 22 – Middleware de validação da configuração

```
const pool = require("../config/db");

async function validateConfigId(req, res, next) {
  const ids = [req.params.id, req.body.id_configuracao];
  if (ids.every((id) => id == null)) {
    return next();
  }
  const { id_usuario } = req;
  for (let id_configuracao of ids) {
    if (id_configuracao == null) {
      continue;
    }

    const result = await pool.query(
      "select 1 from configuracao where id_configuracao = $1 and id_usuario = $2",
      [id_configuracao, id_usuario]
    );
    if (result.rows.length == 0) {
      return res.sendStatus(401);
    }
  }
  return next();
}

module.exports = validateConfigId;
```

Fonte: Autor

### 3.6 ALOCAÇÃO DE MATÉRIAS

Durante os incrementos anteriores, o otimizador gerava grades horárias que continham apenas os nomes dos professores alocados para aula. Em outras palavras, as grades eram matrizes nas quais as linhas eram os horários disponíveis, as colunas eram as salas e cada posição na matriz representava qual professor deveria ministrar a aula naquele horário.

A alocação dos professores é muito importante para a resolução do problema, pois apresenta a maior parte das restrições e dificuldades relacionadas, como os conflitos, por exemplo. Entretanto, em termos de completude, as grades horárias devem ter também a alocação de matérias em cada horário de aula, visto que um mesmo professor pode ministrar mais de uma matéria. As alterações necessárias para possibilitar isso foram:

1. Alterar a modelagem do banco de dados para comportar informações relacionadas às matérias;
2. Alterar telas da interface web para que fosse possível configurar as matérias;
3. Alterar rotas do servidor para persistir as matérias;
4. Alterar código do otimizador para produzir grades horárias com matérias

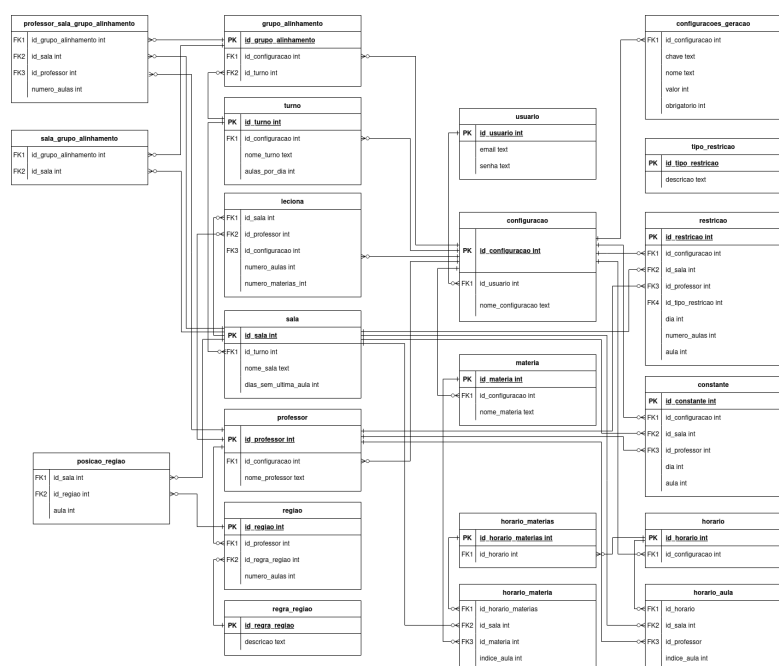
#### 3.6.1 Alteração no banco de dados

Para armazenar informações relacionadas às matérias, a modelagem do banco de dados foi alterada com a criação de três tabelas:

- **matéria**
- **horario-materias**: representa uma grade horária de matérias, que tem vínculo com a entidade principal da grade horária
- **horario-materia**: representa uma matéria alocada em determinada posição de uma grade horária

Com esta modelagem atualizada, presente na figura 23, cada grade horária de professores, pode ter diferentes opções de grades de matérias.

Figura 23 – Modelo Entidade-Relacionamento com Matérias

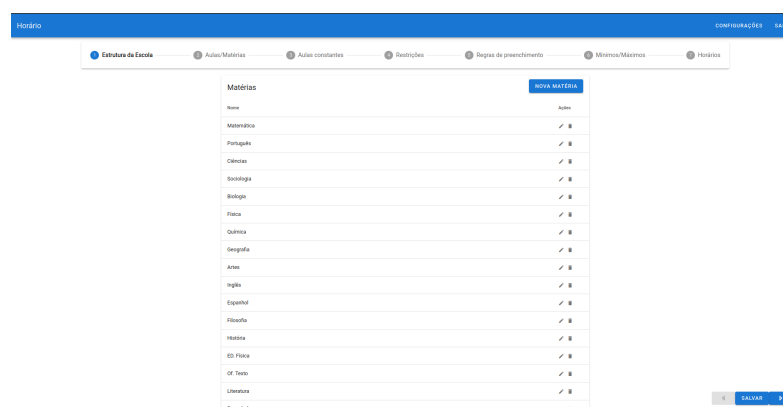


Fonte: Autor

### 3.6.2 Alteração de telas

Com a adição do conceito das matérias ao sistema, algumas telas da interface precisaram ser modificadas. A primeira destas, foi a tela inicial da configuração, ou seja, a tela de "Estrutura da Escola", cuja alteração pode ser vista na figura 24 consistiu na adição de uma seção para cadastro e visualização de matérias, semelhante ao componente de cadastro de professores.

Figura 24 – Estrutura da Escola com Matérias



Fonte: Autor

Além da tela de estrutura, a segunda aba da configuração ("Aulas/Matérias") foi alterada para que fosse possível vincular matérias aos professores, e configurar a quantidade de aulas de cada matéria deve ser ministrada semanalmente, conforme a figura 25.

Figura 25 – Tela de configuração de quantidades de aulas por matéria

Fonte: Autor

Por fim, na tela final da configuração, representável por exibir as grades horárias geradas, foi necessário alterar o componente da grade para incluir, além dos nomes dos professores, os nomes das matérias alocadas para cada horário, como pode ser visto na figura 26.

Figura 26 – Visualização de grade horária com matérias

Fonte: Autor

### 3.6.3 Alteração de métodos no servidor

A adição das matérias também envolveu algumas alterações no servidor. As rotas alteradas para acomodar a melhoria foram rotas de armazenamento e consulta da estrutura da escola, aulas e grades horárias.

### 3.6.4 Matérias no otimizador

Após as alterações na interface, servidor e banco de dados, o sistema estava pronto para lidar com as informações relacionadas às matérias, faltando apenas a incorporação destas

na geração de grades horárias por parte do otimizador. Para realizar isso, implementou-se um método análogo ao [Algoritmo 2](#), porém voltado à geração de grades de matérias.

A diferença é que o novo método implementado obtém determinada grade horária gerada pelo [Algoritmo 2](#), e realiza as operações necessárias para preencher a grade com matérias. Vale ressaltar que essas operações são semelhantes à geração da grade de aulas dos professores: preenchimento da grade inicial, cálculo da variação de custo, realização de trocas de posições e armazenamento das soluções.

O método de geração de grades foi aplicado no otimizador de forma que cada vez que uma grade horária de professores é salva no banco de dados, esta também já tem a sua grade horária de matérias gerada, completando assim o processo de otimização da grade, conforme o [Algoritmo 3](#).

### 3.7 VALIDAÇÃO DE CONFIGURAÇÕES

### 3.8 EXPORTAÇÃO DE GRADES HORÁRIAS

**Algoritmo 3:** Otimizador com matérias

**Input:** Lista de professores  $LP$ , lista de turmas  $LT$ , matriz de aulas e matérias por professor por turma  $MA$ , temperatura inicial  $TI$ , Taxa de resfriamento  $TR$

**Output:** Grade horária de matérias e professores otimizada

$temperatura \leftarrow TI$

$grade \leftarrow \text{CriaGradeInicial}(LP, LT, MA)$

$iteracoesSemAlteracao \leftarrow 0$

$solucoes \leftarrow$  lista vazia

**while** condição de parada não atingida **do**

$deltaTotal \leftarrow 0$

**for** passo = 0 até numeroPassos **do**

$turma \leftarrow grade.EscolheTurmaAleatoria()$

$linhas \leftarrow grade.EscolheHorariosAleatoriosValidos(sala)$

$delta \leftarrow grade.CalculaDelta(sala, linhas)$

$probabilidade \leftarrow e^{-delta/temperatura}$

$valorAceite \leftarrow \text{Aleatorio}(0, 1)$

**if**  $delta < 0$  ou  $probabilidade \geq valorAceite$  **then**

$grade.PermutaProfessores(sala, linha1, linha2)$

$deltaTotal \leftarrow deltaTotal + delta$

**if** grade não existe na lista de soluções **then**

                insere grade na lista de soluções

                limita lista de soluções às 100 melhores grades

**end**

**end**

**end**

**if**  $delta = 0$  **then**

$iteracoesSemAlteracao \leftarrow iteracoesSemAlteracao + 1$

**else**

$iteracoesSemAlteracao \leftarrow 0$

**end**

**if**  $iteracoesSemAlteracao \geq 15$  **then**

**for** gradeProfessores em EscolherGradesRelevantes(solucoes) **do**

$\text{SalvaGradeProfessores}(gradeProfessores)$

$gradesMaterias \leftarrow \text{GeraGradesMaterias}(gradeProfessores)$

$\text{SalvarMelhorGradeMaterias}(gradesMaterias)$

**end**

        apaga lista de soluções

$temperatura \leftarrow TI$

$iteracoesSemAlteracao \leftarrow 0$

**end**

$temperatura \leftarrow temperatura * TR$

**end**

## 4 CONSIDERAÇÕES FINAIS

Em conclusão, este trabalho abordou o *High School Timetabling Problem*, e a demanda associada por um software de otimização de grades horárias. Verificou-se durante a revisão de literatura a dificuldade que a tarefa de planejamento de grades horárias representa, e que grande parte das instituições de ensino do país ainda realiza essa tarefa manualmente.

O presente trabalho trouxe também algumas demonstrações do estado atual do protótipo da aplicação desenvolvida, assim como melhorias sugeridas para a continuação da pesquisa. Com o desenvolvimento dos primeiros incrementos da aplicação, foi possível verificar a viabilidade do projeto aplicando a técnica de *Simulated Annealing* para produzir e otimizar grades horárias escolares, atendendo múltiplos objetivos tradicionais do problema, assim como a nova preocupação trazida pelo Novo Ensino Médio, os Itinerários Formativos.

Resta portanto como atividade de crucial importância a coleta de dados de potenciais usuários futuros da aplicação, para que seja possível verificar a necessidade de adições e melhorias ao otimizador, e a execução dos incrementos do software a fim de cumprir os objetivos propostos.



## Referências

ABRAMSON, D. Constructing school timetables using simulated annealing: Sequential and parallel algorithms. **Management Science**, v. 37, n. 1, p. 98–113, 1991. Disponível em: <<https://EconPapers.repec.org/RePEc:inm:ormnsc:v:37:y:1991:i:1:p:98-113>>. Citado 2 vezes nas páginas 1 e 3.

BARDADY, V. A. Computer-aided school and university timetabling: The new wave. In: BURKE, E.; ROSS, P. (Ed.). **Practice and Theory of Automated Timetabling**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. p. 22–45. ISBN 978-3-540-70682-3. Citado na página 1.

BRASIL. Lei nº 13.415, de 16 de fevereiro de 2017. **Diário Oficial [da] República Federativa do Brasil**, Brasília, DF, 2017. ISSN 1677-7042. Disponível em: <[http://www.planalto.gov.br/ccivil\\_03/\\_ato2015-2018/2017/lei/l13415.htm](http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2017/lei/l13415.htm)>. Citado na página 3.

COOPER, T. B.; KINGSTON, J. H. The complexity of timetable construction problems. In: **International Conference on the Practice and Theory of Automated Timetabling**. [S.l.: s.n.], 1995. Citado na página 3.

FONSECA, G. H.; SANTOS, H. G.; CARRANO, E. G. Integrating matheuristics and metaheuristics for timetabling. **Computers And Operations Research**, v. 74, p. 108–117, 2016. ISSN 0305-0548. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0305054816300879>>. Citado na página 1.

LAARHOVEN, P. J. **Simulated annealing theory and applications**. [S.l.]: Kluwer, 1987. Citado na página 4.

POULSEN, C. J. B. **Desenvolvimento de um Modelo para o School Timetabling Problem Baseado na Meta-Heurística Simulated Annealing**. 141 p. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2012. Citado na página 1.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software - 8ª Edição**. [s.n.], 2016. ISBN 9788580555349. Disponível em: <<https://books.google.com.br/books?id=wexzCwAAQBAJ>>. Nenhuma citação no texto.

TAN, J. S. et al. A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. **Expert Systems with Applications**, v. 165, p. 113943, 2021. ISSN 0957-4174. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0957417420307314>>. Citado 2 vezes nas páginas 1 e 3.