EVANDRO MURILO BRONSTRUP ALVES DA SILVA

IANA PELENTIR RAMOS

UMA ABORDAGEM INTERATIVA PARA O PROBLEMA DA PROGRAMAÇÃO DE HORÁRIOS EM ESCOLAS BRASILEIRAS

Ji-Paraná

2020

EVANDRO MURILO BRONSTRUP ALVES DA SILVA

IANA PELENTIR RAMOS

UMA ABORDAGEM INTERATIVA PARA O PROBLEMA DA PROGRAMAÇÃO DE HORÁRIOS EM ESCOLAS BRASILEIRAS

Monografia apresentada à Banca Examinadora do Centro Universitário São Lucas, como requisito para aprovação para obtenção de Título de Bacharel em Sistemas de Informação.

Orientador: Prof. Me. Thyago Bohrer Borges

Ji-Paraná

2020

Dados Internacionais de Catalogação na Publicação - CIP

S586a Silva, Evandro Murilo Bronstrup Alves da.

Uma abordagem interativa para o problema da programação de horários em escolas brasileiras. / Evandro Murilo Bronstrup Alves da Silva; Iana Pelentir Ramos. – Ji-Paraná, 2020.
42 p., il.

Monografia (Sistema de Informação) – Centro Universitário São Lucas, Ji-Paraná, 2020.

Orientador: Prof. Me. Thyago Bohrer Borges

1. Programação de horário escolar. 2. Modelagem de Software. 3. School Timetabling. 4. Software WEB. 5. Grade horária. I. Ramos, Iana Pelentir. II. Borges, Thyago Bohrer. III. Título.

CDU 004.4

Ficha Catalográfica Elaborada pelo Bibliotecário Giordani Nunes da Silva CRB 11/1125

EVANDRO MURILO BRONSTRUP ALVES DA SILVA

IANA PELENTIR RAMOS

UMA ABORDAGEM INTERATIVA PARA O PROBLEMA DA PROGRAMAÇÃO DE HORÁRIOS EM ESCOLAS BRASILEIRAS

Monografia apresentada à Banca Examinadora do Centro Universitário São Lucas, como requisito para aprovação para obtenção de Título de Bacharel em Sistemas de Informação.

Orientador: Prof. Me. Thyago Bohrer Borges

Ji-Paraná, 11 de dezembro de 2020.

Avaliação/Nota:

BANCA EXAMINADORA

_____ Centro Universitário São Lucas

Prof. Me. Ana Flávia Moreira Camargo

_____ Centro Universitário São Lucas

Prof. Me. Maigon Nacib Pontchuska

_____ Centro Universitário São Lucas

Prof. Me. Thyago Bohrer Borges

RESUMO

Neste trabalho é apresentado um *software* WEB para a resolução do problema da programação de horários em escolas brasileiras utilizando uma abordagem interativa, onde o usuário está em controle de todo o processo de agendamento, em vez de uma abordagem algorítmica, que é a mais comum, onde o papel do usuário geralmente está limitado à entrada de parâmetros iniciais. Todo o processo pode ser realizado com apenas cinco ações distintas e consistentes, utilizando teclado e *mouse*: selecionar um bloco de horário, remover a seleção, definir recursos agendados para o bloco de horário selecionado, clonar recursos de um bloco de horário para outro e trocar de lugar os recursos de dois blocos de horário. Descrevemos as estruturas de dados do *software*, implementadas no modelo relacional, a visualização dos dados num plano bidimensional, e a arquitetura do *software*, incluindo diagramas de comunicação que ilustram a estratégia de comunicação entre cliente e servidor. Estão disponíveis listagens de código que mostram implementações possíveis das consultas a nível de banco de dados e da transformação dos dados da agenda, de acordo com as ações do usuário, a nível de servidor.

Palavras-chave: Programação de horários. Software WEB. Modelagem de Software.

ABSTRACT

We present a WEB application as an interactive solution to the school timetabling problem, where the user is in control of the whole process, as opposed to the more usual algorithmic approach, where the user is responsible only for inserting the initial input parameters. This was made possible with only five distinct and consistent actions, based on keyboard and mouse usage: time block selection, deselection, resource definition for the selected time block, time block cloning and time block swapping. We describe the data structures, implemented in the relational model, the data visualization in a bidimensional plane, the software architecture and the communication strategy between client and server. Code listings with example implementations of database queries and application level data transformations are available.

Keywords: School timetabling. WEB Application. Software design.

LISTA DE FIGURAS

Figura 1 - Representação quadrimensional da *timetable* 18
Figura 2 - Representação tridimensional da *timetable* 19
Figura 3 - Representação tridimensional da *timetable* quando existem várias turmas 19
Figura 4 - Representação bidimensional da timetable, com o tempo contínuo 20
Figura 5 - Representação bidimensional da *timetable*, com o tempo discreto 21
Figura 6 - *Wireframe* da *timetable* com o painel lateral para gestão de recursos.....22
Figura 7 - Captura de tela do estado inicial da grade de horários 23
Figura 8 - Captura de tela da seleção de um bloco de horário.....24
Figura 9 - Captura de tela da definição de professor num bloco de horário.....24
Figura 10 - Captura de tela da clonagem de blocos de horário.....25
Figura 11 - Captura de tela com um conflito na agenda, indicado em vermelho.....25
Figura 12 - Captura de tela da ação de troca de blocos de horário.....26
Figura 13 - Modelo Entidade Relacionamento do banco de dados da aplicação.....28
Figura 14 - Diagrama com os três componentes do software 35
Figura 15 - Diagrama A, demonstrando a requisição de inicialização do sistema.....36
Figura 16 - Diagrama B, demonstrando a atualização da grade de horários 36

LISTAGENS DE CÓDIGO

Listagem 1 - Query para selecionar a agenda filtrando por professor e ordenando por dia e horário 29
Listagem 2 - Query para selecionar horários filtrando por professor e dia 29
Listagem 3 - Query para contar as linhas da agenda filtrando por local e disciplina .29
Listagem 4 - Query para trazer um valor booleano indicando se o filtro por dia, horário e professor retornou 0 resultados 30
Listagem 5 - Query para contar as linhas da agenda filtrando por turma e professor não definido.....30
Listagem 6 - Função com a lógica para definir um bloco de horário 32
Listagem 7- Exemplo de chamada da função "definir", no contexto Agenda 33
Listagem 8 - Função com a lógica para clonar um bloco de horário 33
Listagem 9 - Função com a lógica para trocar dois blocos de horário de lugar.....34

.

SUMÁRIO

1 INTRODUÇÃO 8
1.1 OBJETIVO GERAL 9
1.2 OBJETIVOS ESPECÍFICOS 9
1.3 DELIMITAÇÃO DO ESTUDO 9
1.4 RELEVÂNCIA DO ESTUDO 10
2 O PROBLEMA DA PROGRAMAÇÃO DE HORÁRIOS 11
2.1 ABORDAGEM DE PROGRAMAÇÃO MATEMÁTICA 11
2.2 ABORDAGEM HEURÍSTICA 12
2.3 TIMETABLING INTERATIVA 13
3 GRADES DE HORÁRIOS NAS ESCOLAS DO BRASIL 15
4 DESENVOLVIMENTO DA SOLUÇÃO 17
4.1 REPRESENTAÇÃO VISUAL DA GRADE DE HORÁRIOS 17
4.2 AÇÕES E INTERAÇÃO 21
4.3 EXEMPLO DE WORKFLOW 23
4.4 REPRESENTAÇÃO DOS DADOS 26
4.4.1 Modelo ER do banco de dados 27
4.4.2 Queries principais 28
4.5 TRANSFORMAÇÕES DA AGENDA 30
4.5.1 Criação de uma sala de aula 30
4.5.2 Definição de um bloco de horário 31
4.5.3 Clonar bloco de horário 33
4.5.4 Trocar blocos de lugar 34
4.6 ARQUITETURA DO SOFTWARE 34
4.7 RENDERIZAÇÃO DA GRADE DE HORÁRIOS 36
5 CONCLUSÃO 38
REFERÊNCIAS 40

1 INTRODUÇÃO

Ano após ano notamos que as faculdades e escolas sofrem com o problema de fazer uma boa distribuição de salas e horários no começo dos períodos letivos. Existe uma grande dificuldade em coordenar as turmas, professores e horários diferentes com salas limitadas, que frequentemente contam com requisitos únicos que podem dificultar o bom alocamento.

Nas escolas de ensino fundamental e médio, não é tarefa trivial agendar os horários dos professores de diversas disciplinas para diversas turmas. Por exemplo, é comum que uma mesma professora lecione aula de Língua Portuguesa para todas as turmas do ensino fundamental de uma escola; neste caso, se existem dez turmas e cada uma delas possui carga horária semanal média de cinco horas, é preciso agendar os horários da professora em 50 blocos de aulas semanais. A professora não pode estar em mais de uma sala ao mesmo tempo, e nem pode ocupar o espaço de outros professores, que podem estar em situação semelhante.

Este problema, chamado de problema da programação de horários, por sua complexidade e importância, já é estudado há muito tempo – a referência mais antiga que encontramos data da década de 60 (GOTLIEB, 1963 apud WERRA, 1985) –, e é alvo frequente especialmente de cientistas da computação.

Como mostraremos a seguir, muitas abordagens diferentes já foram tentadas para transformar o conhecimento acumulado nas pesquisas em um *software* que resolva o problema na prática. No Brasil, nós encontramos poucos *softwares* deste tipo, e entre aqueles que tivemos acesso – por possuírem demonstrações gratuitas –, todos utilizam a mesma abordagem de gerar automaticamente a grade de horários utilizando longas configurações como parâmetros de entrada[1].

Neste trabalho, exploramos uma abordagem diferente ao problema, inspirada por Mathaisel e Comm (1991), que é a construção de uma ferramenta interativa, isto é, uma ferramenta na qual o usuário tenha controle de todo o processo de

[1] Sistemas GridClass (disponível em: http://gridclass.com.br/) e Cronos (disponível em: https://cronostimetable.com/).

agendamento. O usuário final planejado desta ferramenta é o funcionário responsável por realizar tal processo em escolas de ensino fundamental e médio brasileiras.

1.1 OBJETIVO GERAL

Desenvolver de forma exploratória uma ferramenta interativa para realizar a programação de horários, adequada para ser utilizada em escolas de ensino fundamental e médio brasileiras.

1.2 OBJETIVOS ESPECÍFICOS

- Definir os principais problemas e abordagens à programação de horários em escolas.
- Definir as particularidades das grades de horário em escolas de ensino fundamental e médio brasileiras.
- Descrever uma estrutura de dados, e método de armazenamento da mesma, que seja adequada ao problema.
- Descrever maneiras possíveis de apresentar os dados num plano bidimensional.
- Descrever o conjunto de interações necessárias para realizar o processo de programação de horários, utilizando teclado e *mouse*.
- Descrever uma arquitetura de *software* possível para a implementação da ferramenta, incluindo o modelo de comunicação entre os diversos componentes.
- Construir um *software* que possa ser utilizado a partir de um navegador de internet, com base nas etapas anteriores.

1.3 DELIMITAÇÃO DO ESTUDO

Modelaremos e desenvolveremos um software para resolver o problema da programação de horários em escolas de ensino fundamental e médio brasileiras utilizando uma abordagem interativa. O *software* deverá ser utilizável a partir de um navegador de internet, para satisfazer as demandas de portabilidade exigidas de *softwares* contemporâneos, e as interações serão baseadas no uso de um teclado e *mouse* (ou *touchpad*), que são as interfaces mais comuns de interação com *notebooks* e computadores de mesa – por limitação de escopo, portanto, interfaces *touchscreen* não serão presentemente consideradas.

Também não levaremos em consideração o ambiente universitário e nem o ensino anterior ao nível fundamental, que, embora parecidos em muitos aspectos, demandariam algumas adaptações no *software*, que estão fora do escopo deste trabalho.

1.4 RELEVÂNCIA DO ESTUDO

O Brasil conta com mais de 180000 escolas de educação básica (DEED, 2019). Cada uma delas oferece aulas de diversas disciplinas, lecionadas por diversos professores, que precisam ser adequadamente agendadas, o que não é um trabalho trivial. Um *software* que auxilie neste processo pode, portanto, ter grande efeito.

Além disso, o problema da programação de horários é estudado há bastante tempo, mas nós encontramos uma desproporção entre os estudos com uma abordagem algorítmica em relação aos estudos com uma abordagem interativa – o presente trabalho constitui também, assim, uma humilde contribuição a fim de diversificar os estudos deste assunto.

2 O PROBLEMA DA PROGRAMAÇÃO DE HORÁRIOS

O problema geral de alocação de horários é conhecido na literatura como *timetabling*, ou programação de horários, onde a resolução do problema é chamada de *timetable*, com duas variantes de especial interesse para o nosso trabalho: STC (*School Timetabling Problem*) e UTC (*University Timetabling Problem*).

Conforme explica Werra (1985), o *timetabling* é normalmente considerado em duas etapas. Na primeira, é definida a matriz curricular para cada turma de alunos e deve-se atribuir os vários recursos, como professores, às turmas. Na segunda, tenta-se criar uma *timetable* que seja compatível com todos os requisitos definidos na primeira etapa. Normalmente o papel de um *software* é resolver a segunda etapa.

Inicialmente, abordagens de programação matemática eram utilizadas para derivar *timetables* viáveis. Posteriormente foram desenvolvidas principalmente heurísticas (VEENSTRA e VIS, 2016).

2.1 ABORDAGEM DE PROGRAMAÇÃO MATEMÁTICA

Podemos citar como exemplo da primeira abordagem o trabalho de Hilton (1981), onde o autor prova dois teoremas para construir *timetables*, que ele define como uma matriz n x n onde cada célula é preenchida por um número de símbolos distintos de determinado conjunto de forma que cada símbolo ocorra exatamente uma vez em cada coluna e exatamente uma vez em cada linha. As linhas podem representar os professores, as colunas podem representar as aulas e os símbolos podem representar os períodos da semana, conforme a tabela a seguir (interpretação e solução nossa):

Quadro 1 - Representação do *school timetabling problem* de Hilton

	1º ano	**2º ano**	**3º ano**	**4º ano**
Carlos	seg, qua	ter	sex	qui
Maria	ter	seg, qui	qua	sex
Guilherme	qui	sex	seg	ter, qua
Flores	sex	qua	ter, qui	seg

Fonte: Os autores (2020), baseado em Hilton (1981).

Outro exemplo é o trabalho de Tripathy (1984), que formula um problema mais complexo de *timetabling* – num programa de graduação com mais de 900 disciplinas –, como um problema de programação linear.

2.2 ABORDAGEM HEURÍSTICA

Como exemplo da segunda abordagem, podemos citar o trabalho de Fonseca (2014), que desenvolveu um algoritmo vencedor da *Third International Timetabling Competition* (ITC2011). São utilizadas técnicas heurísticas para refinar iterativamente uma solução inicial até se alcançar a solução final, que leva em consideração diversas restrições como:

- limitação do número de horários ociosos
- limitação do número de aulas consecutivas do mesmo assunto
- preferência de determinada aula por determinados recursos

Outro exemplo brasileiro dessa abordagem é o trabalho de Poulsen (2012). Ambos os trabalhos utilizam de alguma maneira a metaheurística *simulated annealing*, que funciona através da perturbação do espaço de soluções; inicialmente, quando a "temperatura" está alta, elas são maiores, e com o tempo, com o "esfriamento", elas passam a ser menores, convergindo para uma aproximação da solução ótima global (KIRKPATRICK, 1983). No caso do *school timetabling problem*, a perturbação envolve, por exemplo, trocar dois agendamentos de lugar.

Uma antiga abordagem heurística que merece ser mencionada é a simulação do posicionamento manual (BARRACLOUGH, 1965), que é basicamente um algoritmo de construção automática de *timetables* baseado na forma como as pessoas

realizam a tarefa manualmente. Uma das justificativas para essa abordagem era "sua habilidade comprovada de produzir uma *timetable*". Hoje em dia, embora o problema não esteja completamente solucionado, podemos dizer que existem outras formas comprovadas de se encontrar uma *timetable* viável, visto o progresso na área, delineado nos parágrafos anteriores.

2.3 TIMETABLING INTERATIVA

Bardadym (1995) pesquisou mais de 700 publicações sobre o problema do *timetabling* e observou que, devido a limitações de software e interfaces primitivas, as ferramentas de *timetabling* inicialmente assumiam a geração automática da *timetable*, o que levou ao desenvolvimento de representações de dados e modelos matemáticos adequados. Acrescentamos que mesmo abordagens heurísticas mais recentes, como as mencionadas na seção anterior, continuam assumindo a mesma coisa.

Enquanto a solução completamente automática instigue muito interesse de pesquisadores e possa fazer sentido para grandes escolas e universidades, ela possui algumas desvantagens para uma pequena escola média brasileira. Primeiro vem a necessidade de definir claramente o problema com antecedência, cadastrando meticulosamente todos os recursos que devem ser distribuídos, e depois a definição dos parâmetros da função que avalia a qualidade da *timetable*, que o responsável muitas vezes não vai entender ou ter claramente em mente – quanto menor e mais pessoal o ambiente, mais subjetivas são as restrições e preferências.

Além disso, Veenstra e Vis (2016) observam bem que, tipicamente, esses métodos são feitos com o objetivo de criar uma *timetable* do zero – no início do ano letivo, por exemplo –, entretanto, o usuário da *timetable* precisa ser capaz de realizar pequenos ajustes rapidamente após a publicação, devido a perturbações como transferência de professores, atividades extracurriculares, mudanças na matriz curricular ou adoecimento de professores. Esse processo de adaptação da *timetable*, na prática, é feito manualmente.

Entre as características do que Bardadym chamou de "nova onda", está o surgimento do *timetabling* interativo. De especial interesse para nós são os editores

especialistas de *timetables*, e o que ele indicou como a próxima geração de ferramentas, as com interface orientada a objetos. Mathaisel e Comm (1991), citados como exemplo, já haviam experimentado um método de composição manual utilizando uma pilha de "cartões" que podiam ser arrastados com o mouse e movidos até o local adequado. Considerando as dificuldades apresentadas nos dois parágrafos anteriores, pensamos ser natural a observação contida em Mathaisel e Comm, que diz que frequentemente o usuário trabalha somente com as ferramentas interativas e rejeita os resolvedores automáticos.

Pelos motivos acima expostos, é esse tipo de ferramenta que queremos criar neste trabalho. Um sistema interativo que auxilie o processo manual de criação de *timetables*, e não a automação completa do processo.

3 GRADES DE HORÁRIOS NAS ESCOLAS DO BRASIL

Santos e Souza (2007) descrevem da seguinte maneira o problema de programação de horários nas escolas brasileiras

> No Brasil, o problema corresponde ao problema de programação de horários encontrado em escolas (5º a 8º séries) de ensino fundamental e ensino médio. Nesse problema, cada professor deve lecionar um determinado número de aulas para cada turma em um conjunto de períodos. As alocações devem ser feitas considerando-se a não existência de conflitos, ou seja, cada professor e turma deve se envolver com, no máximo, uma atividade letiva por período.

Em nossa experiência, a grade de horários é sempre semanal, e o mais comum é que existam quatro horários por dia para cada turma.

A LDB (Lei de Diretrizes e Bases da Educação Nacional) estabelece que a carga horária mínima anual é de oitocentas horas tanto para o ensino fundamental quanto para o ensino médio, e devem ser distribuídas por um mínimo de duzentos dias de trabalho escolar (Art. 24, I) (BRASIL, 1996). Está previsto que tal carga horária seja gradualmente ampliada, no ensino médio, para mil e quatrocentas horas (Art. 24, VII, §1º).

A Base Nacional Comum Curricular (BNCC) define 8 componentes curriculares para o ensino fundamental (MEC, 2020):

- Língua portuguesa
- Arte
- Educação física
- Matemática
- Ciências
- Geografia
- História
- Ensino religioso

Aos anos finais do ensino fundamental, acrescenta-se o componente curricular "Língua inglesa".

A Lei nº 13.415/2017 alterou a LDB, estabelecendo que o currículo do ensino médio será composto pela BNCC e por itinerários formativos (BRASIL, 2017), o que efetivamente substituiu o modelo único do currículo do ensino médio por um modelo mais flexível. Para os fins deste trabalho, podemos usar como referência, sem prejuízo, um currículo baseado no currículo único antigo, com os seguintes componentes:

- Língua portuguesa
- Matemática
- Arte
- Química
- Física
- Biologia
- Geografia
- História
- Filosofia
- Sociologia
- Educação física
- Inglês
- Espanhol

Assim, considerando a divisão semanal da grade, com uma semana de cinco dias letivos e quatro blocos de horário por dia, temos vinte blocos de horário por semana, para os quais devem ser alocados nove tipos de componente curricular para o ensino fundamental, e, no nosso exemplo, treze para o ensino médio. Para uma única turma, não é uma programação de horários particularmente difícil de se realizar. Se a escola, entretanto, possuir duas turmas para cada série do 6º ano do ensino fundamental ao 3º do ensino médio, totalizando quatorze, teremos 280 blocos de horários que precisam ser agendados de forma que não ocorram conflitos entre as agendas dos professores, e de forma que cada turma tenha sua agenda completamente preenchida com a quantidade correta de horários para cada componente curricular.

4 DESENVOLVIMENTO DA SOLUÇÃO

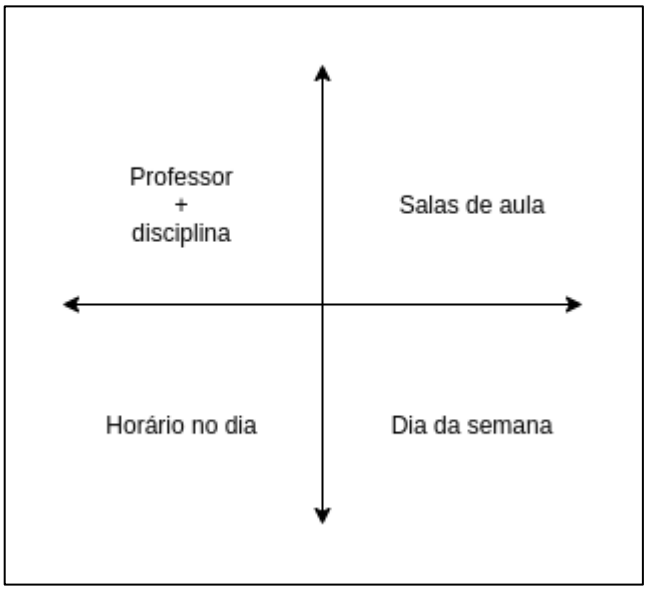
Nesta seção descreveremos o *software* criado para resolver o problema apresentado anteriormente, junto com todo o processo que levou a cada uma das decisões de *design* e modelagem de dados do sistema.

4.1 REPRESENTAÇÃO VISUAL DA GRADE DE HORÁRIOS

A solução do problema, como bem descreveram Comm e Mathaisel (1991), consiste na distribuição efetiva de recursos limitados no espaço-tempo. Se considerarmos as disciplinas, as salas de aula, os professores e o tempo como dimensões distintas, o problema fica definido num espaço quadrimensional. Se considerarmos o tempo em dois eixos, um para o dia da semana e outro para os horários do dia, como tradicionalmente é feito, temos ainda outra dimensão. Nosso primeiro objetivo é reduzir ao máximo o número de dimensões, a fim de facilitar o manuseio das informações: o computador pode muito bem lidar um número arbitrário de dimensões, mas elas precisam ser representadas para um humano, e são humanos que vão desenvolvê-la.

Duas das dimensões podem ser facilmente acopladas numa só: professores e disciplinas. Isso se dá porque as disciplinas não são recursos limitados que devem ser distribuídos – duas turmas podem cursar a mesma disciplina no mesmo horário sem qualquer problema. Quer dizer que, num determinado bloco da agenda, o professor e a disciplina podem formar uma unidade, cuja disponibilidade é ditada somente pela agenda do professor.

Figura 1 - Representação quadrimensional da *timetable*



Fonte: Os autores (2020).

Como essa unidade é o recurso principal a ser movimentado, já que as salas de aula e os horários são fixos, vamos considerá-la um ponto a ser posicionado num plano tridimensional. Entretanto, precisamos representar esse plano em duas dimensões somente, na tela do computador, porque manipular e interpretar um ambiente tridimensional simulado costuma ser complicado – imagine um cubo mágico onde cada pequeno bloco é um agendamento. Nosso primeiro movimento nesse sentido foi transformar a dimensão "salas de aula" em diversos planos bidimensionais, de forma que cada um pudesse ser representado como uma tabela ou planilha:

Figura 2 - Representação tridimensional da *timetable*

1º ano A					
	seg	ter	qua	qui	sex
1	Matemática (Carlos Augusto)				
2					
3					
4					

Fonte: Os autores (2020).

No exemplo acima ilustrado, o professor Carlos está agendado na segunda-feira, no primeiro horário, na sala de aula do "1º ano A", onde dará aula de matemática. O problema dessa abordagem aparece quando existe um conflito de agenda entre duas salas de aula diferentes:

Figura 3 - Representação tridimensional da *timetable* quando existem várias turmas

1º ano A					
	seg	ter	qua	qui	sex
1	Matemática (Carlos Augusto)				
2					
3					
4					

1º ano B					
	seg	ter	qua	qui	sex
1	Matemática (Carlos Augusto)				
2					
3					
4					

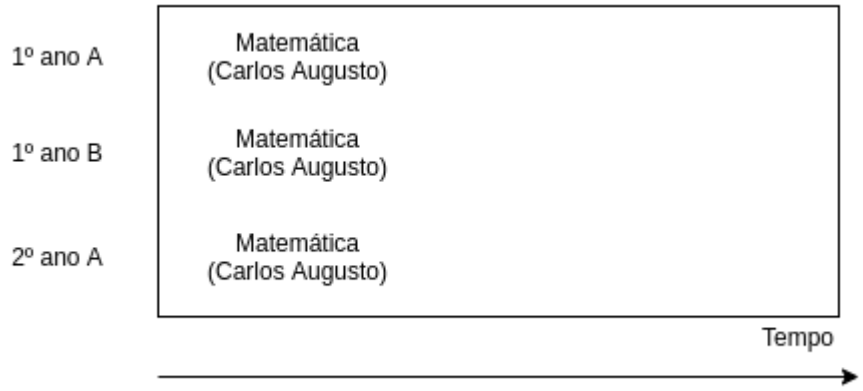
2º ano A					
	seg	ter	qua	qui	sex
1	Matemática (Carlos Augusto)				
2					
3					
4					

Fonte: Os autores (2020).

Não importa a forma como arranjemos as tabelas, identificar um conflito – ou ausência deste – envolve o trabalho de encontrar o mesmo ponto em N planos

diferentes, onde N é a quantidade de salas de aula disponíveis. Existe uma abordagem claramente superior, que adaptamos de um dos exemplos no trabalho de Comm e Mathaisel (1991), e envolve reduzir ainda outra dimensão do problema. No nosso caso, o mais óbvio é utilizar uma única dimensão contínua para o tempo, desfazendo-nos assim das dimensões "dia da semana" e "horário do dia". Desta forma, os conflitos se destacam naturalmente e podem ser identificados olhando uma única dimensão:

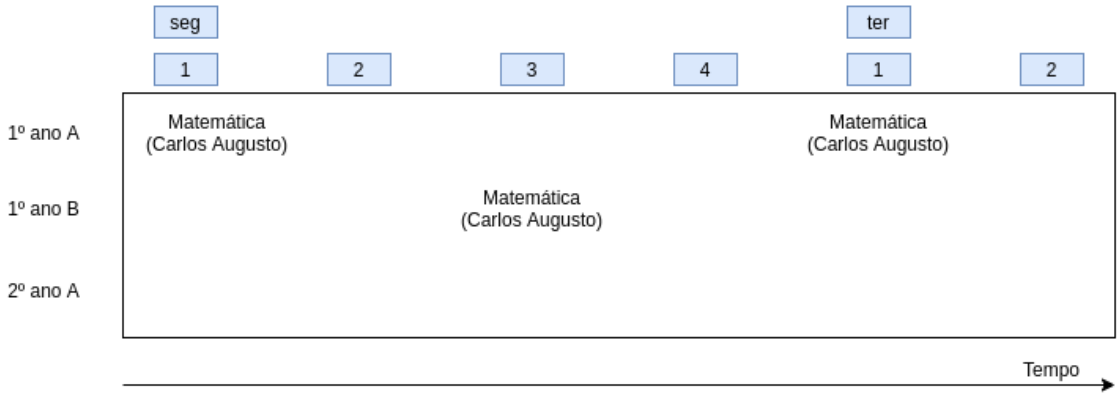
Figura 4 - Representação bidimensional da timetable, com o tempo contínuo



Fonte: Os autores (2020).

A desvantagem dessa abordagem é que ela não condiz com a forma como esse tipo de agenda é normalmente entendida, que é discreta, e não contínua. Para resolver isso, optamos por exibir a dimensão "tempo" como uma sequência discreta de blocos de horário, em vez de horários reais, com uma indicação do dia da semana na parte superior.

Figura 5 - Representação bidimensional da *timetable*, com o tempo discreto



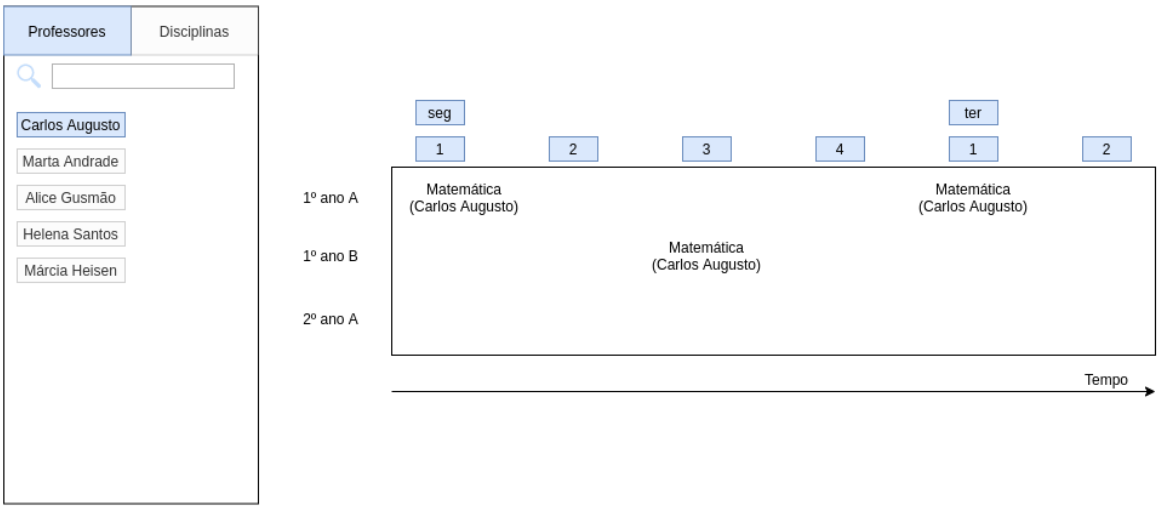
Fonte: Os autores (2020).

No exemplo acima, o professor Carlos Augusto lecionará a disciplina de matemática para o 1º ano A no primeiro horário da segunda-feira e no primeiro horário da terça-feira. O 1º ano B terá aula de matemática no terceiro horário da segunda-feira.

4.2 AÇÕES E INTERAÇÃO

Como dissemos, o recurso que deve ser distribuído no plano são os professores, e precisamos indicar quais disciplinas eles lecionarão. Assim, é importante que eles estejam sempre facilmente disponíveis para o usuário. Resolvemos isso com uma barra lateral fixa, sempre visível, de onde o usuário pode visualizar e adicionar os recursos que poderá distribuir.

Figura 6 - *Wireframe* da *timetable* com o painel lateral para gestão de recursos



Fonte: Os autores (2020).

A primeira possibilidade de interação que consideramos foi a ação de "arrastar e soltar". O usuário arrastava o professor para um lugar na grade de horários e ele era agendado. Também podia trocar dois blocos da agenda de lugar arrastando um para cima do outro. O problema é que a grade de horários representada desta forma pode ficar muito longa, e arrastar e soltar um bloco de uma extremidade para a outra se torna rapidamente uma tarefa fatigante.

Em vez disso, optamos pela seleção com um clique do mouse como a ação principal do sistema, ampliada por duas teclas modificadoras, *control* e *shift*. Essa pequena combinação de teclas foi o suficiente para desenvolvermos todas as interações necessárias no sistema, listadas a seguir:

- Um clique num bloco de horário não selecionado torna-o a seleção atual.
- Um clique num bloco de horário selecionado remove a seleção.
- Um clique no professor ou disciplina, na barra lateral, agenda o recurso para o bloco de horário selecionado, substituindo o antigo se um existir.
- Caso exista um bloco selecionado, um clique com a tecla modificadora *control* pressionada em outro lugar clona os recursos do primeiro para o segundo bloco, e seleciona o segundo.
- Caso exista um bloco selecionado, um clique com a tecla modificadora *shift* pressionada em outro lugar troca de lugar os recursos dos dois blocos, e remove a seleção.

Esse pequeno arsenal de ações consistentes foi o suficiente para oferecer uma experiência ágil e intuitiva para a montagem da grade de horários.

Por fim, a cada ação, caso algum conflito aconteça na agenda, os blocos conflitantes são imediatamente iluminados, tornando o problema claro para o usuário.

4.3 EXEMPLO DE WORKFLOW

Utilizaremos um breve exemplo prático, apoiado por capturas de tela do software desenvolvido para este trabalho, a fim de mostrar visualmente como funcionam as interações descritas na seção anterior.

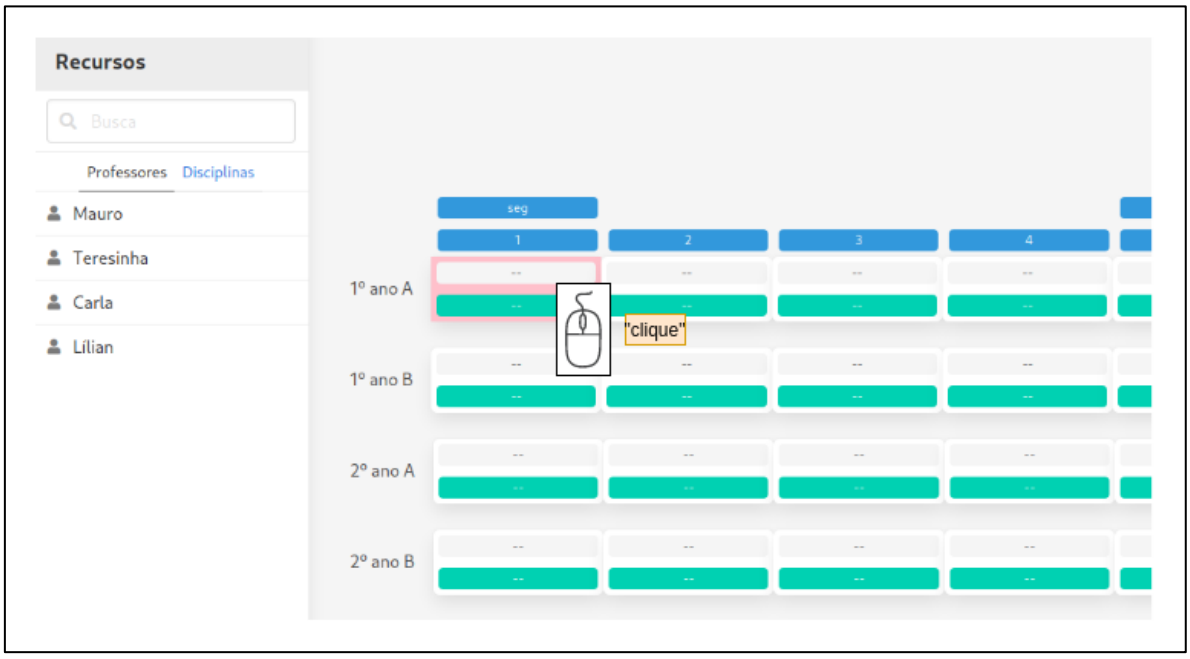
Num primeiro momento, o usuário tem a grade de horários vazia:

Figura 7 - Captura de tela do estado inicial da grade de horários

Fonte: Os autores (2020)

A fim de definir a agenda do primeiro horário da segunda-feira para a turma 1º ano A, ele utiliza a ação básica de selecionar um bloco de horário:

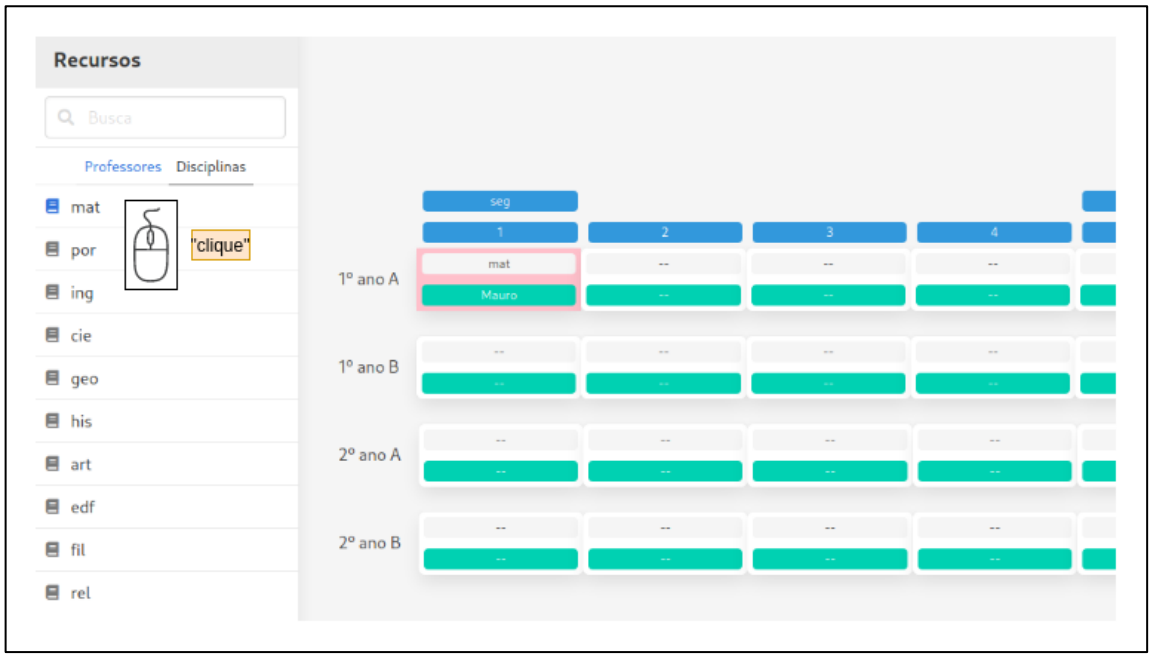
Figura 8 - Captura de tela da seleção de um bloco de horário



Fonte: Os autores (2020).

Com o bloco de horário selecionado, o usuário faz uso da barra lateral para definir o professor e a disciplina:

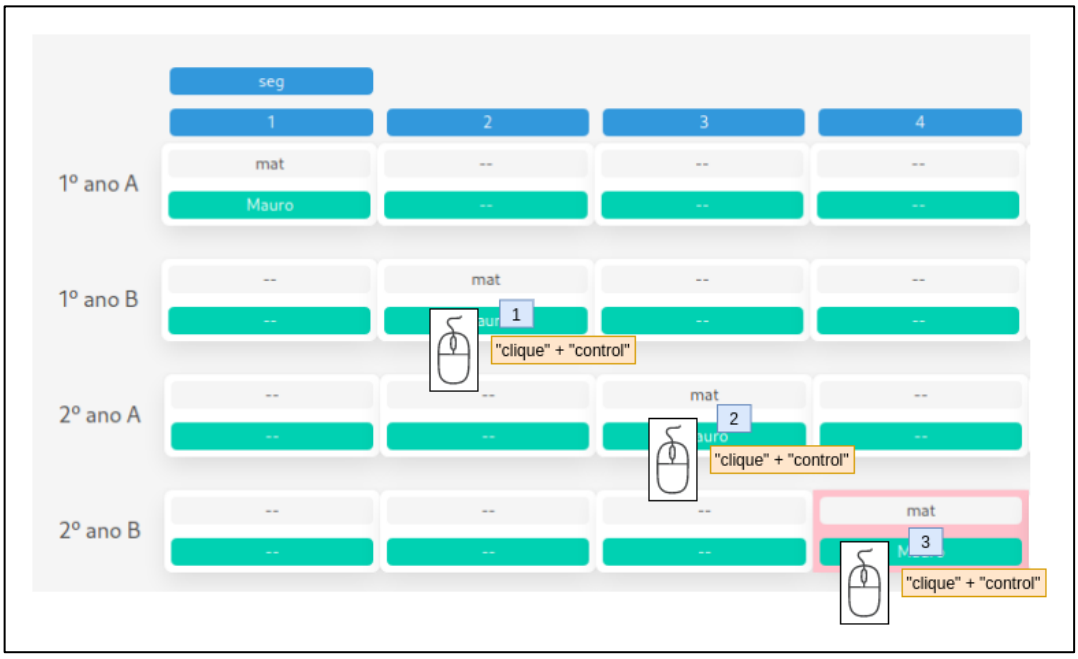
Figura 9 - Captura de tela da definição de professor num bloco de horário



Fonte: Os autores (2020).

Utilizando a ação de clonar, ele rapidamente define o restante da agenda do professor para aquele mesmo dia:

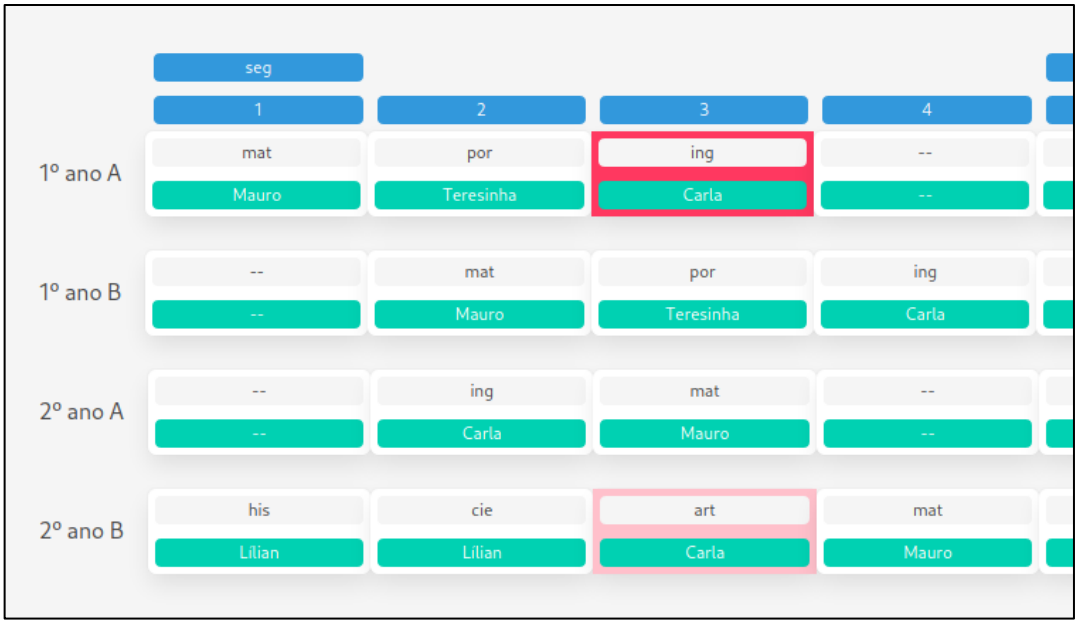
Figura 10 - Captura de tela da clonagem de blocos de horário



Fonte: Os autores (2020).

Eventualmente, ao agendar um professor, o sistema pode apontar um conflito de horários, iluminando os blocos de horário conflitantes com a cor vermelha:

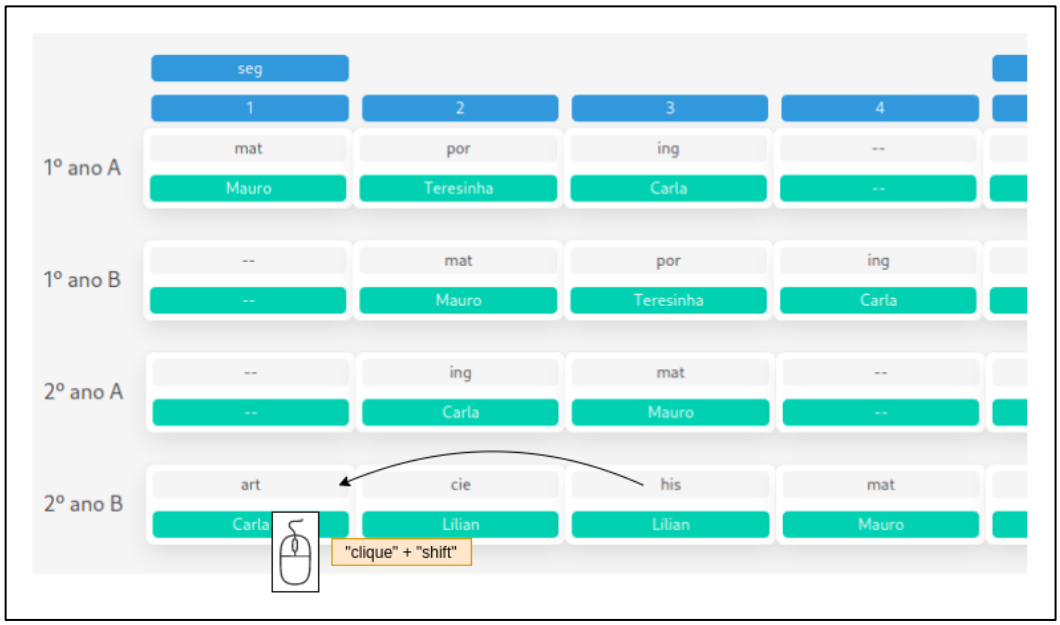
Figura 11 - Captura de tela com um conflito na agenda, indicado em vermelho



Fonte: Os autores (2020).

Neste caso, o conflito pôde ser resolvido com uma única ação de trocar blocos de horário:

Figura 12 - Captura de tela da ação de troca de blocos de horário



Fonte: Os autores (2020).

4.4 REPRESENTAÇÃO DOS DADOS

Para o armazenamento de dados, escolhemos utilizar um banco de dados relacional, porque nossos dados podem ser naturalmente dispostos numa tabela, porque são referenciados de diversas direções diferentes, e porque, com uma modelagem de dados apropriada, a linguagem SQL (*Structured Query Language*) provou-se muito conveniente para responder diversas perguntas pertinentes sobre a agenda, como demonstraremos posteriormente.

Nas seções anteriores, identificamos três tipos de recursos distintos – salas de aula, professores e disciplinas –, que convidam três tabelas correspondentes. São elas: locais, pessoas, eventos. Tais nomes foram escolhidos por representarem sucintamente o papel de cada recurso no problema do agendamento.

A grade de horários é representada por uma única tabela, chamada agenda. Cada bloco de horário é uma linha na tabela, e liga uma pessoa, um evento e um local a um horário. O horário em si é representado como as coordenadas de um ponto, desta forma podemos representar uma unidade contínua, que é o tempo, de forma

discreta. A coordenada X representa o dia da semana, enquanto a coordenada Y representa o horário do dia. A tabela a seguir traz alguns exemplos:

Quadro 2 - Exemplos da interpretação das coordenadas

Coordenada	**Significado**
(0, 0)	Primeiro horário da segunda-feira
(2, 3)	Quarto horário da quarta-feira
(4, 1)	Primeiro horário da sexta-feira

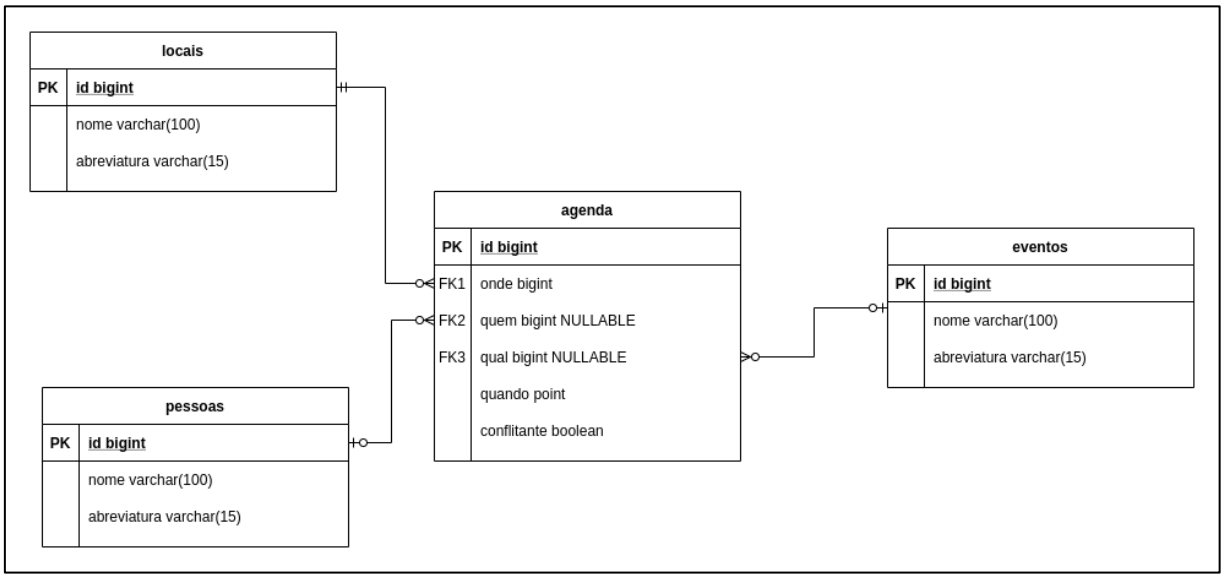
Fonte: Os autores (2020).

Observe que a associação da coordenada X aos dias da semana é arbitrária: 0 poderia facilmente representar o domingo, por exemplo. Esse tipo de representação possui todas as conveniências das representações discretas – como a iteração –, e mantém a flexibilidade da representação contínua do tempo: podemos facilmente adicionar um bloco de horário ao dia, por exemplo, para que o sistema se adeque a escolas com 5 horários.

4.4.1 Modelo ER do banco de dados

Escolhemos o Postgresql como nosso RDBMS (*Relational Database Management System*), por ser robusto, ter código aberto e possuir bom suporte a tipos geométricos. A seguir apresentamos o modelo ER (Entidade Relacionamento) do banco de dados, com os tipos de dados do Postgresql, utilizando a notação *Crow's foot* para representar a cardinalidade.

Figura 13 - Modelo Entidade Relacionamento do banco de dados da aplicação



Fonte: Os autores (2020).

Nota: A notação utilizada é a *Crow's foot.*

Essencialmente, a tabela agenda poderia possuir, no lugar de uma chave primária simples, uma chave primária composta, constituída pelos campos onde e qual, já que cada turma possui apenas um bloco por horário.

Por fim, a tabela agenda possui uma coluna chamada conflitante, que serve como um cache da situação atual do bloco, para reduzir o tempo de computação de algumas operações.

4.4.2 Queries principais

Demonstraremos a adequação do modelo relacional ao problema proposto através de algumas queries que visam responder questões pertinentes sobre a agenda.

No Postgresql, uma coluna do tipo point pode ser manipulada através da notação de *array*. No nosso caso, quando[0] indica o dia da semana, e quando[1] indica o horário do dia. Além disso, uma coordenada inteira pode ser comparada utilizando o operador "~=". Os exemplos, a começar do primeiro, farão amplo uso dessa sintaxe.

Quando uma nova sala de aula é adicionada ao sistema, são inseridas no banco de dados todas as linhas que representam a sua agenda. Todas as operações posteriores apenas alteram os campos "quem", "qual" e "conflitante". A conveniência disso se tornará clara com os exemplos 2 e 5.

1. Qual é a agenda do professor Carlos (id: 1)?

Listagem 1 - Query para selecionar a agenda filtrando por professor e ordenando por dia e horário

```
SELECT * SELECT *

FROM agenda

WHERE quem = 1

ORDER BY quando[0], quando[1];
```

2 - Em que horários da terça-feira o professor Carlos está livre?

Listagem 2 - Query para selecionar horários filtrando por professor e dia

```
SELECT quando[1]

FROM agenda

WHERE quem != 1

  AND quando[0] = 1

ORDER BY quando[1];
```

3 - Quantas aulas de matemática (id: 1) o 1º ano A (id: 1) tem agendadas?

Listagem 3 - Query para contar as linhas da agenda filtrando por local e disciplina

```
SELECT COUNT(*)

FROM agenda

WHERE onde = 1

  AND qual = 1;
```

4 - O professor Carlos está livre no primeiro horário da quinta-feira?

Listagem 4 - Query para trazer um valor booleano indicando se o filtro por dia, horário e professor retornou 0 resultados

```
SELECT COUNT(*) = 0

FROM agenda

WHERE quando ~= point(0, 0)

  AND quem = 1;
```

5 - Quantos horários vagos o 1º ano A possui?

Listagem 5 - Query para contar as linhas da agenda filtrando por turma e professor não definido

```
SELECT COUNT(*)

FROM agenda

WHERE onde = 1

  AND quem IS NULL;
```

4.5 TRANSFORMAÇÕES DA AGENDA

A cada ação realizada no sistema, os dados da agenda são transformados de alguma maneira. A existência da agenda começa com o cadastro de uma sala de aula, e passa pelas definições de professores e disciplinas, além das trocas de horários.

4.5.1 Criação de uma sala de aula

Quando o usuário cria uma sala de aula, sua agenda é criada imediatamente e constitui-se de uma grade D x B, onde D é a quantidade de dias letivos na semana e B é a quantidade de blocos de horário por dia, representada por linhas da tabela, de acordo com o descrito na seção anterior.

Salvo por mudança nas variáveis D e B, esse é o único momento em que linhas são inseridas na tabela "agenda". Todas as outras ações meramente manipulam essas linhas já existentes. Além de facilitar algumas queries, isso permite que os blocos de horário sejam manipulados por uma interface única, sem precisar checar se um dos blocos envolvidos em determinada operação existe ou não. Selecionar um bloco é sempre selecionar um bloco que já existe, mesmo que ele esteja vazio.

4.5.2 Definição de um bloco de horário

Todas as demais ações são formas específicas de chamar uma mesma ação principal, que é a definição dos dados de um bloco de horário. Esta ação é uma função que aceita quatro parâmetros: onde, quando, quem e qual. "Onde" e "quando" identificam o bloco de horário a ser alterado, enquanto "quem" e "qual" são os valores que serão definidos na linha correspondente na tabela.

Se o parâmetro "quem" está presente, a função trata de buscar, através de uma query como a 4 da seção anterior, se um conflito está sendo criado ou resolvido, e o campo "conflitante" é atualizado adequadamente não só no bloco em questão, mas em todos os blocos relacionados. A função é curta e pode ser mostrada por inteira (em PHP):

Listagem 6 - Função com a lógica para definir um bloco de horário

```
function definir(
    int $onde,
    string $quando,
    int $quem = null,
    int $qual = null
)
{
    $quemAnterior = Agenda::obter($onde, $quando)->quem;

    DB::table('agenda')
        ->where('onde', $onde)
        ->whereRaw('quando ~= ?', $quando)
        ->update(compact('quem', 'qual'));

    if ($quemAnterior !== null) {
        Pessoa::atualizarConflitos($quemAnterior, $quando);
    }

    if ($quem !== null) {
        Pessoa::atualizarConflitos($quem, $quando);
    } else {
        Agenda::limparConflito($onde, $quando);
    }
}
```

Esta função, que está num contexto chamado Agenda, poderia ser chamada da seguinte maneira para definir o professor 2 e a disciplina 5 no terceiro horário do primeiro dia da semana na sala de aula 1:

Listagem 7- Exemplo de chamada da função "definir", no contexto Agenda

```
Agenda::definir(1, point(0, 2), 2, 5);
```

A função primeiro obtém o bloco do banco de dados e armazena o valor anterior do campo "quem" na variável "quemAnterior". A seguir, a agenda é atualizada no banco de dados.

A função "atualizarConflitos", do contexto "Pessoa", vai limpar ou marcar conflitos nos blocos do horário "quando" tanto para o professor anterior do bloco, quanto para o professor recém definido.

4.5.3 Clonar bloco de horário

A função para clonar um bloco de horário pode ser facilmente definida nos termos da função anterior:

Listagem 8 - Função com a lógica para clonar um bloco de horário

```
public static function clonar(
    int $ondeA,
    string $quandoA,
    int $ondeB,
    string $quandoB
)
{
    $a = Agenda::obter($ondeA, $quandoA);
    Agenda::definir($ondeB, $quandoB, $a->quem, $a->qual);
```

```
}
```

Primeiro obtemos o professor e a disciplina do bloco A, e depois os definimos no bloco B. Como a lógica de checagem de conflitos está encapsulada na função "definir", não precisamos repeti-la em nenhum outro lugar.

4.5.4 Trocar blocos de lugar

Assim como a função anterior, esta pode ser facilmente escrita nos termos da função "definir":

Listagem 9 - Função com a lógica para trocar dois blocos de horário de lugar

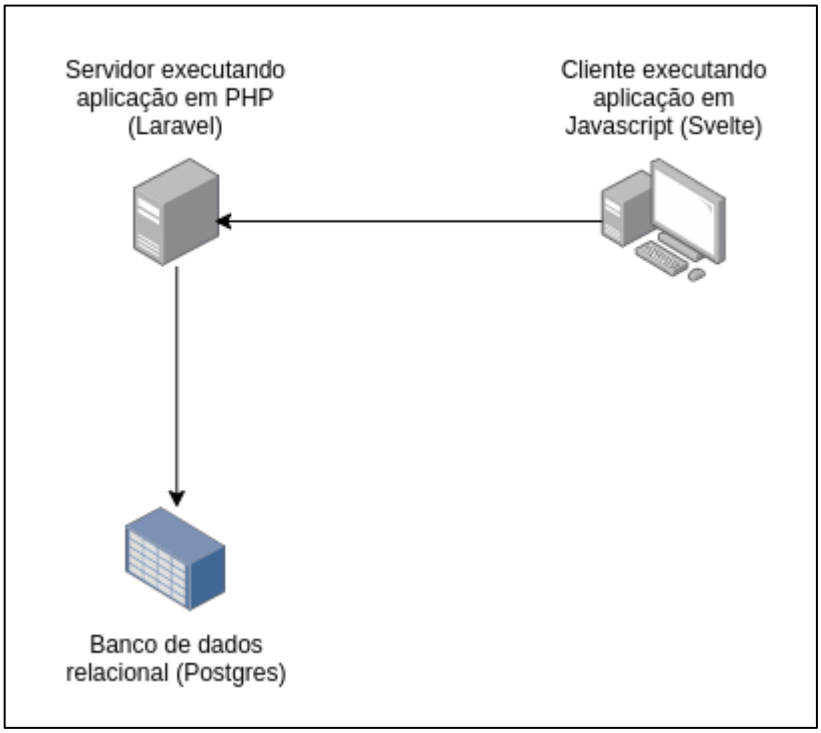
```
public static function trocar(
    int $ondeA,
    string $quandoA,
    int $ondeB,
    string $quandoB
)
{
    $a = Agenda::obter($ondeA, $quandoA);
    $b = Agenda::obter($ondeB, $quandoB);

    Agenda::definir($ondeA, $quandoA, $b->quem, $b->qual);
    Agenda::definir($ondeB, $quandoB, $a->quem, $a->qual);
}
```

4.6 ARQUITETURA DO SOFTWARE

O software desenvolvido para este trabalho é uma aplicação WEB, e como tal é utilizada através de um navegador de internet que se comunica com um servidor, que no nosso caso tem como motor uma aplicação desenvolvida na linguagem PHP. Os componentes principais da aplicação estão ilustrados no diagrama abaixo.

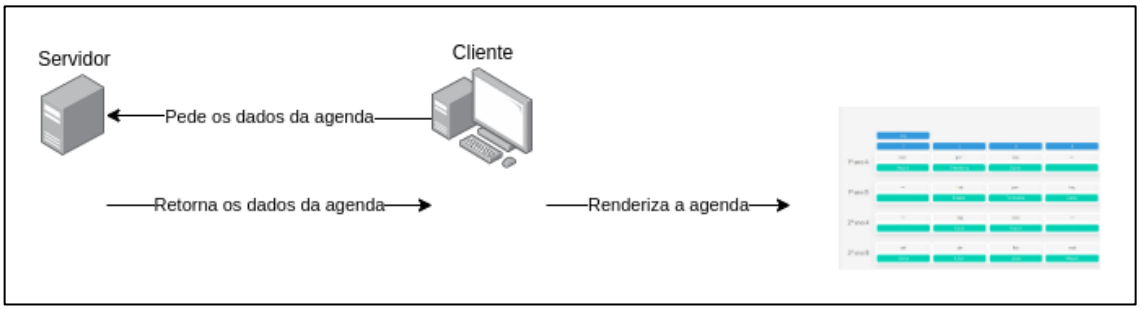
Figura 14 - Diagrama com os três componentes do software



Fonte: Os autores (2020).

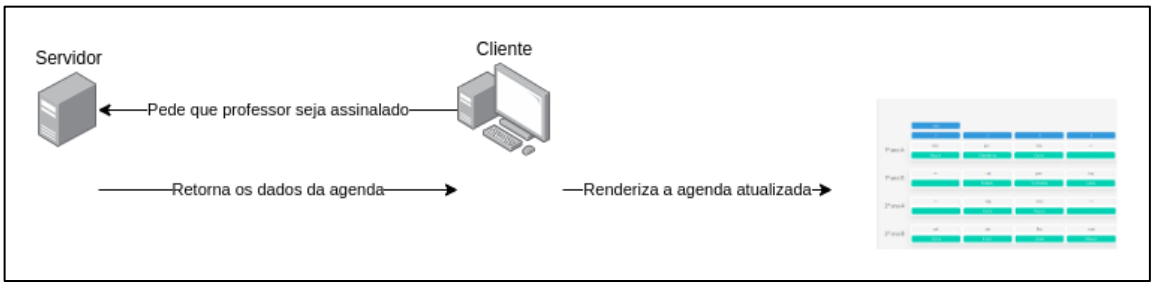
A fim de mantermos todo o "cérebro" do software nos diversos contextos da aplicação em PHP do servidor, utilizando diretamente o modelo relacional para identificar conflitos, como demonstrado na seção anterior, a aplicação em Javascript é o que chamamos de *thin client*, isto é, um cliente que realiza o mínimo de processamento possível, repassando a maior parte do trabalho para o servidor. Consequentemente, a utilidade do software depende diretamente da disponibilidade de uma conexão contínua entre o cliente e o servidor. Os dois diagramas a seguir mostram um esquema simplificado de como funciona a comunicação entre o cliente e o servidor.

Figura 15 - Diagrama A, demonstrando a requisição de inicialização do sistema



Fonte: Os autores (2020)

Figura 16 - Diagrama B, demonstrando a atualização da grade de horários



Fonte: Os autores (2020).

No diagrama A, o cliente, na inicialização do sistema, faz uma requisição dos dados da agenda ao servidor. Assim que os recebe, utiliza-os para renderizar a agenda no navegador. No diagrama B, o cliente pede que determinado professor seja assinalado a determinado bloco de horário. O servidor realiza a ação e retorna novamente os dados da agenda, que o cliente utiliza para atualizar a renderização, a fim de refletir visualmente as alterações realizadas. O ponto importante a se notar é que o cliente não faz nenhuma manipulação direta dos dados, e se contenta em meramente exibi-los da forma como foram retornados pelo servidor.

4.7 RENDERIZAÇÃO DA GRADE DE HORÁRIOS

Como a grade de horários é retornada inteira a cada requisição, e pode precisar ser atualizada em diversos locais devido a possíveis conflitos, é importante que a

operação de renderização seja rápida e eficiente. Para tal, utilizamos a *framework* de componentes Svelte, que, através de uma etapa de compilação, transforma o código declarativo num código imperativo em Javascript altamente eficiente – as atualizações da grade de horários são realizadas quase que instantaneamente, sem nenhuma lentidão para o usuário.

Para diminuir o tamanho do retorno do servidor, a fim de deixar a comunicação mais rápida, na inicialização do cliente é realizada uma etapa extra, que consiste em baixar um dicionário com todos os nomes das salas de aula, professores e disciplinas. Assim, estes não precisam ser enviados nas requisições posteriores, que podem conter somente os identificadores numéricos como referência: a atualização de uma agenda com 80 blocos de horário gera uma resposta de apenas 5kb.

5 CONCLUSÃO

A abordagem tradicional ao problema da programação de horários consiste em descrever um algoritmo ou modelo que permita a "resolução" do problema, no sentido de que o resultado da aplicação da técnica é uma agenda de horários sem conflitos, e por vezes até obedecendo a determinados parâmetros qualitativos, ou então um aviso de que a agenda é impossível de ser montada. Sistemas assim implementados possuem, entretanto, muitos estorvos para as pequenas escolas brasileiras – primeiro porque todos os parâmetros precisam ser configurados previamente para que a agenda possa ser montada, o que se dará automaticamente, mas que de qualquer modo custa muito trabalho fastidioso, e segundo porque a agenda não vai levar em consideração preferências por vezes muito pessoais, intrínsecas a ambientes de trabalho menores, e restrições muito específicas, que podem variar de escola para escola. Por fim, tais sistemas normalmente não são pensados para realizar pequenas alterações pontuais posteriores à criação da agenda, que acabam tendo que ser feitas manualmente.

Nesta monografia, seguimos a linha de indagação de Mathaisel e Comm (1991), e desenvolvemos um sistema que, em vez de substituir o tipo de trabalho realizado pelo usuário – o agendamento pela configuração de parâmetros –, age como uma ferramenta para auxiliar o trabalho original. Tal ferramenta é necessariamente interativa e funciona em tempo real, dando ampla liberdade ao usuário. Sendo interativa desde o princípio, não existe nenhuma dificuldade em realizar pequenas alterações pontuais na agenda, pois o processo se mantém o mesmo.

Utilizando apenas cinco ações, todas consistentes umas com as outras, criamos um *software WEB* através do qual é possível realizar o processo de programação da grade de horários interativamente, com *feedback* em tempo real. O modelo relacional mostrou-se muito adequado para o problema em questão, e a maior parte do processamento do software pôde ser repassado para o banco de dados.

Nosso objetivo era criar uma ferramenta para ajudar no problema de programação de horários em escolas, mas a *framework* desenvolvida pode ser utilizada, com poucas modificações, para resolver diversos outros problemas semelhantes. Nosso orientador, Thyago Bohrer, por exemplo, sugeriu a utilização do

software em instituições de ensino superior, para auxiliar na formulação das ofertas de disciplinas.

REFERÊNCIAS

BARDADYM, Victor A.. Computer-aided school and university timetabling: the new wave. **Practice And Theory Of Automated Timetabling**, [S.L.], p. 22-45, 1996. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/3-540-61794-9_50.

BARRACLOUGH, E. D.. The application of a digital computer to the construction of timetables. **The Computer Journal**, [S.L.], v. 8, n. 2, p. 136-146, 01 fev. 1965. Oxford University Press (OUP). http://dx.doi.org/10.1093/comjnl/8.2.136.

BRASIL. Lei nº 9.394, de 20 de dezembro de 1996. Estabelece as diretrizes e bases da educação nacional. **Diário Oficial da União**, Brasília, DF, 23 dez. 1996.

BRASIL. Lei nº 13.415, de 16 de fevereiro de 2017. Institui a Política de Fomento à Implementação de Escolas de Ensino Médio em Tempo Integral. **Diário Oficial da União**, Brasília, DF, 17 de fev. 2017.

DEED. **Censo da Educação Básica**. 2019. Disponível em: http://portal.inep.gov.br/documents/186968/0/Notas+Estat%C3%ADsticas+-+Censo+da+Educa%C3%A7%C3%A3o+B%C3%A1sica+2019/43bf4c5b-b478-4c5d-ae17-7d55ced4c37d?version=1.0. Acesso em: 20 out. 2020.

FONSECA, George Henrique Godim da. **Métodos de Buca Heurística para Problemas de Programação de Horários Modelados em XHSTT**. 2013. 97 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal de Ouro Preto, Ouro Preto, 2013.

GOTLIEB, C. C. **The construction of class-teacher timetables**. Proceeding IFIP Congress 1962, Amsterdam, p. 73-77, 1963.

HARRIS, Rich. **Svelte 3**: rethinking reactivity. Rethinking reactivity. 2019. Disponível em: https://svelte.dev/blog/svelte-3-rethinking-reactivity. Acesso em: 18 out. 2019.

HILTON, A.J.W.. School Timetables. **North-Holland Mathematics Studies**, [S.L.], p. 177-188, 1981. Elsevier. http://dx.doi.org/10.1016/s0304-0208(08)73465-0.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P.. Optimization by Simulated Annealing. **Science**, [S.L.], v. 220, n. 4598, p. 671-680, 13 maio 1983. American Association for the Advancement of Science (AAAS). http://dx.doi.org/10.1126/science.220.4598.671.

MATHAISEL, Dennis F.X.; COMM, Clare L.. Course and classroom scheduling: an interactive computer graphics approach. **Journal Of Systems And Software**, [S.L.], v. 15, n. 2, p. 149-157, maio 1991. Elsevier BV. http://dx.doi.org/10.1016/0164-1212(91)90052-8.

MEC. **Base Nacional Comum Curricular**. Disponível em: http://basenacionalcomum.mec.gov.br/. Acesso em: 20 out. 2020.

POULSEN, Camilo José Bornia. **Desenvolvimento de um Modelo para o School Timetabling Problem Baseado na Meta-Heurística Simulated Annealing**. 2012. 141 f. Dissertação (Mestrado) - Curso de Administração, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2012.

SANTOS, Haroldo Gambini; SOUZA, Marcone Jamilson Freitas. Programação de Horários em Instituições Educacionais: formulações e algoritmos. **XXXIX SBPO** A Pesquisa Operacional e o Desenvolvimento Sustentável, Fortaleza, p. 2827-2884, 31 jun. 2007.

TRIPATHY, Arabinda. School Timetabling -- A Case in Large Binary Integer Linear Programming. **Management Science,** v. 30, n. 12, 1984.

VEENSTRA, Marjolein; VIS, Iris F.A.. School timetabling problem under disturbances. **Computers & Industrial Engineering**, [S.L.], v. 95, p. 175-186, maio 2016. Elsevier BV. http://dx.doi.org/10.1016/j.cie.2016.02.011.

WERRA, D. de. An introduction to timetabling. **European Journal Of Operational Research**, [S.L.], v. 19, n. 2, p. 151-162, fev. 1985. Elsevier BV. http://dx.doi.org/10.1016/0377-2217(85)90167-5.