UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LEONARDO DAVID TORRES BEZERRA

DESENVOLVIMENTO DE UMA INTERFACE WEB PARA ALOCAÇÃO DE HORÁRIOS

RUSSAS

2025

LEONARDO DAVID TORRES BEZERRA

DESENVOLVIMENTO DE UMA INTERFACE WEB PARA ALOCAÇÃO DE HORÁRIOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Alexandre Matos Arruda

RUSSAS

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B469d Bezerra, Leonardo David Torres.
Desenvolvimento de uma interface web para alocação de horários / Leonardo David Torres Bezerra. – 2023.
32 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2023.
Orientação: Prof. Dr. Alexandre Matos Arruda.

1. Alocação de horários. 2. Usabilidade. 3. Reactjs. 4. Sistema Web. I. Título.

CDD 005

LEONARDO DAVID TORRES BEZERRA

DESENVOLVIMENTO DE UMA INTERFACE WEB PARA ALOCAÇÃO DE HORÁRIOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: 21/08/2025.

BANCA EXAMINADORA

Prof. Dr. Alexandre Matos Arruda (Orientador)
Universidade Federal do Ceará - UFC

Prof. Ms. Pitagoras Graça Martins
Universidade Federal do Ceará - UFC

Wesley Seidel Carvalho
INTELIDADOS SOLUÇÕES TECNOLÓGICAS

RESUMO

Este trabalho apresenta o desenvolvimento de uma interface web voltada à alocação de horários acadêmicos, com o objetivo de tornar esse processo mais acessível, visual e eficiente para coordenadores e docentes. A proposta surgiu da necessidade de eliminar barreiras técnicas impostas por ferramentas que dependem de linha de comando ou interfaces complexas. A interface foi construída com foco em usabilidade, oferecendo recursos como arrastar e soltar (*drag-and-drop*), bloqueio de horários, exportação e importação de dados no formato JSON, e visualização clara de conflitos.

Desenvolvida com React.js, TypeScript e Tailwind CSS, a aplicação adota uma abordagem minimalista e de fácil manutenção. Ao contrário de soluções robustas porém tecnicamente exigentes, esta ferramenta busca equilibrar simplicidade de uso e flexibilidade de edição manual, priorizando a experiência do usuário em detrimento da automação total. A metodologia aplicada foi iterativa e centrada no usuário, com validação contínua a partir de reuniões semanais.

Palavras-chave: alocação de horários; usabilidade; reactjs; sistema web.

ABSTRACT

This work presents the development of a web interface for academic schedule allocation, aiming to make this process more accessible, visual, and efficient for coordinators and professors. The proposal emerged from the need to eliminate technical barriers imposed by tools that rely on command-line usage or complex interfaces. The interface was built with a focus on usability, offering features such as drag-and-drop, time-slot blocking, data import and export in JSON format, and clear visualization of scheduling conflicts.

Developed with React.js, TypeScript, and Tailwind CSS, the application adopts a minimalist and maintainable approach. Unlike robust yet technically demanding solutions, this tool seeks to balance ease of use with the flexibility of manual editing, prioritizing user experience over full automation. The applied methodology was iterative and user-centered, with continuous validation through weekly meetings.

Keywords: schedule allocation; usability; reactjs; web system.

SUMÁRIO

1 INTRODUÇÃO

A alocação de horários acadêmicos é um desafio recorrente enfrentado por coordenadores e professores no início de cada semestre letivo. Esse processo exige a consideração de diversos fatores, como a disponibilidade dos docentes, a carga horária das disciplinas, a capacidade das salas e a inexistência de conflitos entre turmas. Embora existam ferramentas, tanto pagas quanto gratuitas, que automatizam esse processo por meio de algoritmos especializados, muitas delas apresentam interfaces complexas e pouco intuitivas, o que dificulta sua adoção por usuários que não possuem familiaridade com tecnologias mais avançadas.

Com o objetivo de automatizar esse processo, no campus da Universidade Federal do Ceará em Russas, foi desenvolvido previamente um algoritmo de alocação de horários acadêmicos, em um trabalho independente, que permite gerar alocações viáveis com base em restrições e parâmetros definidos. Esse algoritmo, eficiente do ponto de vista técnico, opera por meio de linha de comando e exige conhecimento específico sobre estruturas de entrada e sintaxe para sua correta utilização.

Neste contexto, o presente trabalho propõe o desenvolvimento de uma **interface web (frontend)** que se conecta ao algoritmo já existente, com o intuito de oferecer uma interface gráfica acessível e intuitiva, eliminando a necessidade de interação técnica com a linha de comando.

O foco central do projeto está na usabilidade e na experiência do usuário, permitindo que coordenadores e professores possam realizar a alocação de horários de maneira visual, utilizando recursos como arrastar e soltar, bloqueio de horários e exportação em formato JSON. A aplicação foi desenvolvida com tecnologias modernas como React.js, TypeScript e Tailwind CSS, adotando uma abordagem minimalista para garantir clareza e simplicidade na interface.

Diferentemente de outras soluções que costumam ser complexas e exigir múltiplas configurações, esta plataforma prioriza a facilidade de uso e a redução da carga cognitiva, ou seja, busca-se minimizar o esforço mental do usuário por meio de interfaces intuitivas, etapas reduzidas e apresentação clara de informações, permitindo que o usuário execute sua tarefa com rapidez e sem necessidade de treinamentos extensos.

Este trabalho contribui não apenas para a melhoria do processo de alocação de instituições, mas também para a democratização do acesso à soluções automatizadas, ao eliminar barreiras técnicas e tornar o sistema mais acessível a profissionais da área de gestão acadêmica.

2 OBJETIVOS

Embora existam ferramentas robustas para a alocação de horários acadêmicos, a maioria apresenta interfaces complexas, exige conhecimentos técnicos avançados ou funciona apenas por meio de linha de comando, como é o caso do método utilizado no Campus da UFC em Russas.

2.1 Objetivo Geral

Desenvolver uma plataforma web para alocação e gerenciamento de horários acadêmicos para o ensino superior na UFC - Campus Russas, que ofereça uma interface gráfica intuitiva e de fácil utilização, baseada em tecnologias modernas, permitindo integração com algoritmos de geração automática de horários e atendendo às demandas de coordenadores, docentes e demais gestores acadêmicos.

2.2 Objetivos Específicos

- **Projetar e implementar uma interface gráfica intuitiva** com recursos de arrastar e soltar para manipulação da grade de horários, visando reduzir a curva de aprendizado e eliminar a necessidade de conhecimentos técnicos avançados.
- **Desenvolver funcionalidades de interoperabilidade**, possibilitando importação e exportação de dados no formato JSON, facilitando a integração com algoritmos de *backend* e permitindo a reutilização de configurações já existentes.
- **Adotar uma arquitetura modular e escalável**, utilizando componentes reutilizáveis e padrões de projeto que favoreçam manutenção, evolução do sistema e integração futura com outros serviços.
- **Permitir configuração manual flexível** da grade, de forma visual e direta, minimizando parametrizações complexas e tornando o sistema acessível a usuários com diferentes níveis de experiência tecnológica.
- **Garantir acessibilidade e disponibilidade**, com hospedagem em nuvem e suporte a múltiplos dispositivos, assegurando acesso remoto, atualizações contínuas e alta disponibilidade.
- **Implementar funcionalidades de restrições**, como bloqueio/desbloqueio de horários, a fim de otimizar o processo de alocação.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Alocação de Horários

A alocação de horários é um desafio clássico da Ciência da Computação, amplamente modelado como um problema de *otimização combinatória*. Esse tipo de problema envolve encontrar, em um espaço finito mas extremamente vasto de combinações, uma solução que atenda a um conjunto de restrições, como evitar conflitos de horário, balancear a carga horária entre docentes e respeitar limites de disponibilidade. No ambiente acadêmico, trata-se de alocar disciplinas, professores, turmas e horários de forma eficiente e sem sobreposição.

Do ponto de vista teórico, esse problema é classificado como *NP-difícil*, ou seja, não há algoritmo conhecido capaz de resolvê-lo em tempo polinomial para todas as instâncias (GAREY; JOHNSON, 1979). A complexidade cresce exponencialmente à medida que se adicionam professores, disciplinas e restrições, tornando inviável a solução por métodos exaustivos, especialmente em contextos institucionais com múltiplos cursos e departamentos.

Para problemas NP-difíceis como este, a literatura oferece diversas técnicas e estratégias computacionais que, embora não garantam a solução ótima em tempo polinomial, são capazes de encontrar soluções de boa qualidade em tempo computacional aceitável. Essas abordagens incluem algoritmos exatos para instâncias menores, métodos heurísticos e metaheurísticos, além de técnicas híbridas que combinam diferentes estratégias. Diversas estratégias computacionais vêm sendo empregadas na tentativa de resolver o problema de forma eficiente:

- **Backtracking (Algoritmo de Retroceder):** método de busca que explora recursivamente as possibilidades, recuando sempre que encontra uma violação de restrições. Embora forneça soluções exatas, seu custo computacional é elevado (CORMEN *et al.*, 2022).
- **Algoritmos de fluxo:** aplicáveis quando o problema pode ser modelado em grafos, com restrições representadas como limites de capacidade. Oferecem soluções eficientes em problemas estruturados (AHUJA *et al.*, 1993).
- **Simulated Annealing (Recozimento simulado):** técnica inspirada no resfriamento de metais, utiliza perturbações aleatórias e aceita soluções subótimas temporárias para escapar de mínimos locais (KIRKPATRICK *et al.*, 1983).
- **Algoritmos Genéticos:** baseados na seleção natural, cruzamento e mutação de soluções, apresentando bons resultados em problemas com múltiplos critérios e grandes espaços de busca (HOLLAND, 1975).

- **Programação por Restrições (Constraint Programming):** permite a modelagem declarativa do problema, especificando as restrições que devem ser satisfeitas, sendo especialmente eficaz em problemas com grande número de regras específicas (RUSSELL; NORVIG, 2020).

Na literatura especializada, Carter e Laporte (CARTER; LAPORTE, 1996) realizaram uma das revisões mais influentes sobre problemas de alocação de horários em instituições de ensino superior, categorizando os métodos existentes e analisando sua eficácia prática. Lewis (LEWIS, 2008) propôs melhorias por meio de técnicas híbridas baseadas em busca local e heurísticas metaheurísticas.

No Brasil, trabalhos como os de Paiva (PAIVA, 2021) e Shiba (SHIBA, 2023) propõem ferramentas web com foco na usabilidade, permitindo que usuários interajam com algoritmos complexos de maneira mais acessível. Essas soluções destacam-se não apenas pela abordagem algorítmica, mas pela preocupação com a interface e a experiência do usuário.

Assim, este trabalho se alinha às abordagens mais recentes ao buscar uma solução que priorize a simplicidade de uso sem desconsiderar a complexidade algorítmica subjacente ao problema de alocação de horários.

3.1.1 Desafios e Complexidade do Problema

O problema de alocação de horários acadêmicos é reconhecidamente complexo, tanto em termos computacionais quanto administrativos. Do ponto de vista computacional, trata-se de um problema de otimização combinatória altamente restritivo, com múltiplas variáveis e regras que devem ser simultaneamente satisfeitas. Do ponto de vista institucional, envolve diversos atores com diferentes preferências e limitações, como professores, turmas, salas e turnos.

A dificuldade do problema cresce exponencialmente à medida que o número de elementos envolvidos aumenta. Para ilustrar, considere uma situação típica de uma coordenação de curso com os seguintes dados (valores aproximados):

- 25 professores alocados;
- 40 disciplinas distintas;
- 10 salas disponíveis;
- 5 dias da semana com 4 períodos disponíveis por dia (totalizando 20 células horárias);
- Restrições de disponibilidade individual por professor;
- Restrições quanto ao uso exclusivo de salas;

- Preferências de agrupamento ou separação de aulas;
- Restrições de sobreposição entre disciplinas do mesmo período.

Para ilustrar essa complexidade, considere uma grade semanal com 20 células horárias disponíveis (5 dias úteis $\times$ 4 blocos de horário por dia) e 40 disciplinas a serem alocadas. Se cada disciplina pudesse ser alocada em qualquer célula sem restrição, o número bruto de possibilidades seria da ordem de:

$$20^{40} = 1{,}09 \times 10^{52}$$

Esses números demonstram o quão rapidamente o problema se torna intratável por métodos exaustivos, justificando o uso de algoritmos heurísticos ou interfaces interativas que facilitem o processo de tomada de decisão pelos coordenadores.

Outro aspecto que dificulta o processo é a necessidade de ajustes manuais durante o semestre, o que exige uma solução que não seja apenas automatizada, mas também flexível e de fácil modificação. Por isso, ferramentas que integrem visualização intuitiva com suporte a algoritmos de apoio à decisão tornam-se extremamente valiosas.

A proposta deste trabalho surge justamente como resposta a essas dificuldades, oferecendo uma ferramenta que, mesmo sem resolver todas as camadas algorítmicas de forma automática, reduz consideravelmente o esforço humano por meio de uma interface interativa e organizada, compatível com o fluxo real de trabalho dos coordenadores.

3.2 Desenvolvimento Web

O desenvolvimento web evoluiu significativamente nas últimas décadas, passando de apenas páginas estáticas para aplicações complexas e interativas. Atualmente, o desenvolvimento de aplicações web modernas baseia-se em arquiteturas que separam claramente as responsabilidades entre frontend e backend, permitindo maior flexibilidade, escalabilidade e manutenibilidade.

As tecnologias de frontend modernas, como React.js, Angular e Vue.js, introduziram o conceito de *Single Page Applications* (SPAs), onde a interface do usuário é renderizada dinamicamente no navegador, proporcionando experiências mais fluidas. Essas tecnologias utilizam o conceito de componentes reutilizáveis, facilitando a organização do código e permitindo o desenvolvimento modular de interfaces complexas.

O ecossistema JavaScript surgiu como uma escolha dominante para o desenvolvimento web, tanto no lado cliente quanto no servidor. Esta unificação tecnológica oferece várias vantagens: redução da curva de aprendizado para desenvolvedores, reutilização de código entre frontend e backend, e um vasto ecossistema de bibliotecas e ferramentas disponíveis.

A escolha tecnológica em projetos de desenvolvimento web sempre envolve ponderações entre diferentes fatores: performance, produtividade do desenvolvedor, curva de aprendizado, suporte da comunidade, e adequação ao escopo do projeto. Neste trabalho, optou-se por tecnologias baseadas em JavaScript devido à sua maturidade no desenvolvimento de interfaces interativas, ao vasto ecossistema de bibliotecas especializadas em drag-and-drop (arrastar e soltar) e manipulação de elementos visuais, e à facilidade de integração com APIs para futura conectividade com algoritmos de backend.

3.3 Interação Humano-Computador

A Interação Humano-Computador (IHC) é uma disciplina que estuda como as pessoas interagem com computadores e como projetar interfaces que sejam eficazes, eficientes e satisfatórias para o usuário. No contexto de sistemas acadêmicos, os princípios de IHC tornam-se fundamentais para o sucesso da adoção da ferramenta.

Os princípios de usabilidade de Nielsen (NIELSEN, 1994) estabelecem diretrizes essenciais para o design de interfaces: visibilidade do status do sistema, correspondência entre o sistema e o mundo real, controle e liberdade do usuário, consistência e padrões, prevenção de erros, reconhecimento em vez de memorização, flexibilidade e eficiência de uso, design estético e minimalista, e ajuda aos usuários no reconhecimento, diagnóstico e recuperação de erros.

Para o sistemas de alocação de horários, pode ser destacado alguns aspectos específicos de IHC que são particularmente relevantes:

- **Feedback visual:** fornecimento de respostas imediatas às ações do usuário, como destacar conflitos de horário ou confirmar a alocação de disciplinas.
- **Affordances (Recursos):** elementos da interface que sugerem naturalmente sua função, como botões que parecem clicáveis e áreas que indicam onde elementos podem ser posicionados.
- **Modelo mental:** alinhamento entre a representação mental que o usuário tem do processo de alocação e a forma como este é apresentado na interface.

A aplicação destes princípios no desenvolvimento da plataforma de alocação de

horários justifica escolhas de design como o uso de drag-and-drop (que aproveita metáforas do mundo físico), a representação visual da grade horária (que corresponde ao formato tradicional usado em papel), e o feedback imediato para ações do usuário.

3.4 Desenvolvimento Incremental de Software

O desenvolvimento incremental de software é uma abordagem metodológica que propõe a construção de sistemas através de ciclos sucessivos de desenvolvimento, onde cada incremento adiciona funcionalidades ao produto (SOMMERVILLE, 2015). Esta abordagem contrasta com o modelo em cascata tradicional, oferecendo maior flexibilidade para adaptação a mudanças de requisitos e permitindo validação contínua com usuários finais.

Os principais benefícios do desenvolvimento incremental incluem (BECK, 2000):

- **Entrega de valor antecipada:** funcionalidades básicas podem ser disponibilizadas rapidamente, permitindo que usuários comecem a obter benefícios do sistema mesmo durante o desenvolvimento.
- **Redução de riscos:** problemas de design ou requisitos inadequados são identificados precocemente, reduzindo o custo de correção.
- **Feedback contínuo:** usuários podem avaliar incrementos funcionais e fornecer feedback para orientar o desenvolvimento subsequente.
- **Adaptabilidade:** mudanças nos requisitos podem ser incorporadas em iterações futuras sem comprometer todo o projeto.

No contexto deste trabalho, a abordagem incremental foi fundamental para o desenvolvimento da plataforma de alocação de horários. O desenvolvimento iniciou com funcionalidades básicas de visualização da grade horária, progrediu para a implementação de drag-and-drop, e posteriormente incorporou recursos de importação/exportação e bloqueio de horários. Esta progressão permitiu validação contínua da usabilidade e adequação da interface às necessidades dos usuários.

A escolha por tecnologias que suportam desenvolvimento ágil, como React.js com seu desenvolvimento baseado em componentes, facilitou a implementação desta abordagem incremental, permitindo ciclos rápidos de desenvolvimento, teste e refinamento da interface do usuário.

4 TRABALHOS RELACIONADOS

Esta seção apresenta uma análise dos principais trabalhos relacionados a este projeto, destacando suas contribuições, abordagens metodológicas e principais diferenças. A revisão da literatura foi estruturada considerando os seguintes aspectos: tecnologias empregadas, metodologias de desenvolvimento e soluções para problemas similares.

Alguns projetos na literatura utilizam tecnologias semelhantes à adotada neste trabalho, apresentando resultados relevantes para nossa análise comparativa.

4.1 Plataforma de gerenciamento de horários, Shiba

Shiba (SHIBA, 2023), desenvolveu uma plataforma de gerenciamento de horários utilizando React.JS. Foi aplicado um questionário de pesquisa de forma a validar o uso da ferramenta pelos usuários e concluído que há bem poucos trabalhos que abordem este problema com uma interface gráfica.

De funcionalidades, como mostrado na figura 1, Shiba (SHIBA, 2023) implementou login social com conta Google, tela dedicada para um tutorial de como utilizar a interface gráfica, tela de geração de grade com passos a serem seguidos informando alguns dados como nome da instancia (nome da grade), unidades curriculares (professor, disciplina, aulas semanais, min. de dias e alunos inscritos), salas (nome e capacidade), termos e restrições. Outra tela implementada foi a de Minhas grades que mostra uma listagem de todas as grades curriculares criadas, tendo opção de visualizar e deletar.

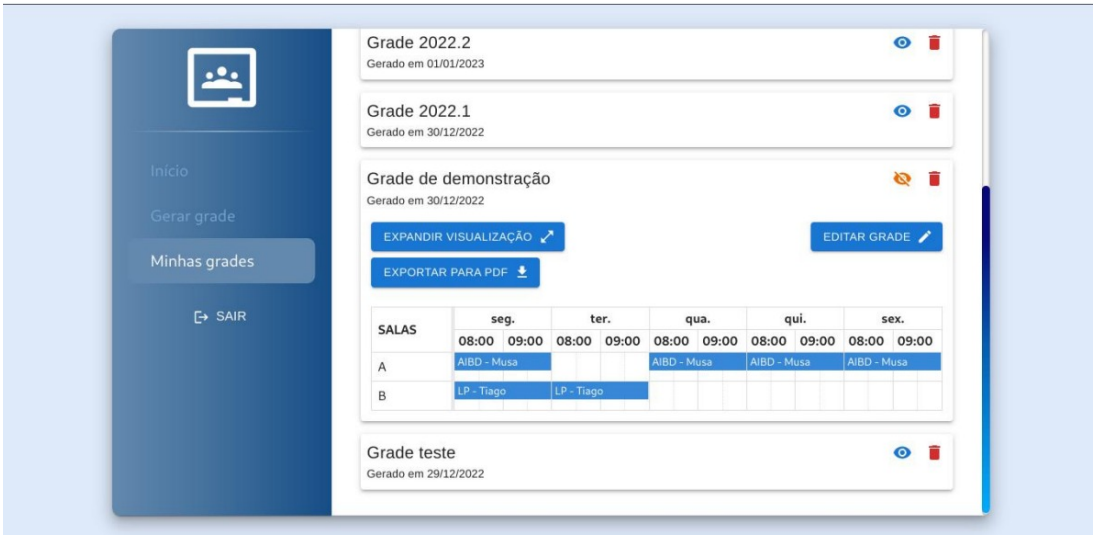

Figura 1 – Tela de visualização de grade curricular

4.2 Cronos

Existe uma solução paga no mercado chamada Cronos (Cronostime Table, 2025) (figura 2) que oferece diversas vantagens como garantia de horários sem choques, utilizando Inteligencia Artifical e modelos matemáticos para garantir isso. Parâmetros personalizáveis como eliminação de aulas triplas, aulas geminadas e isoladas, deslocamento entre sedes, acesso restrito ao sistema web, compartilhamento de horários, geração de relatórios, dentre outras.

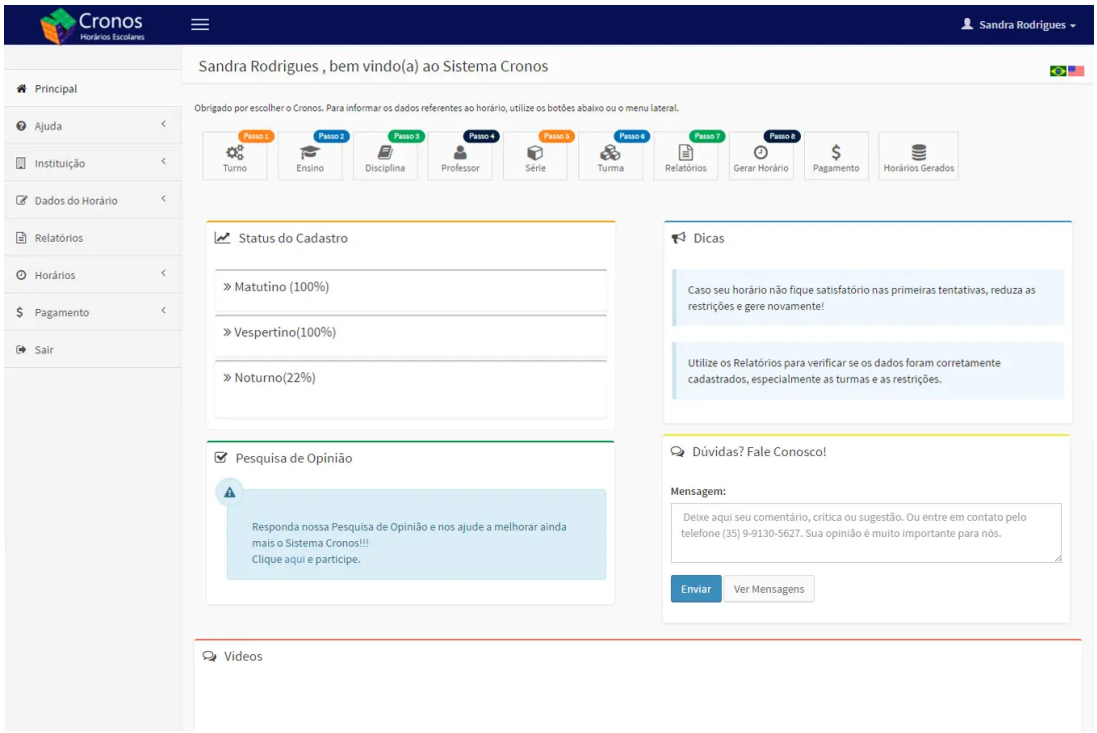

Figura 2 – Tela de visualização de grade curricular

4.3 Ambiente web para programação de horários, Paiva

Outra solução relevante identificada foi o sistema desenvolvido por Paiva (PAIVA, 2021) (figura 3), que consiste em um ambiente web para programação de horários educacionais. Embora tenha sido implementado utilizando tecnologias distintas das empregadas no presente trabalho, esta contribuição merece destaque por compartilhar princípios fundamentais semelhantes: proporcionar uma interface amigável, intuitiva e de fácil utilização para os usuários. A abordagem de Paiva reforça a importância da experiência do usuário no desenvolvimento de sistemas de gerenciamento de horários acadêmicos, alinhando-se com os objetivos principais deste projeto.

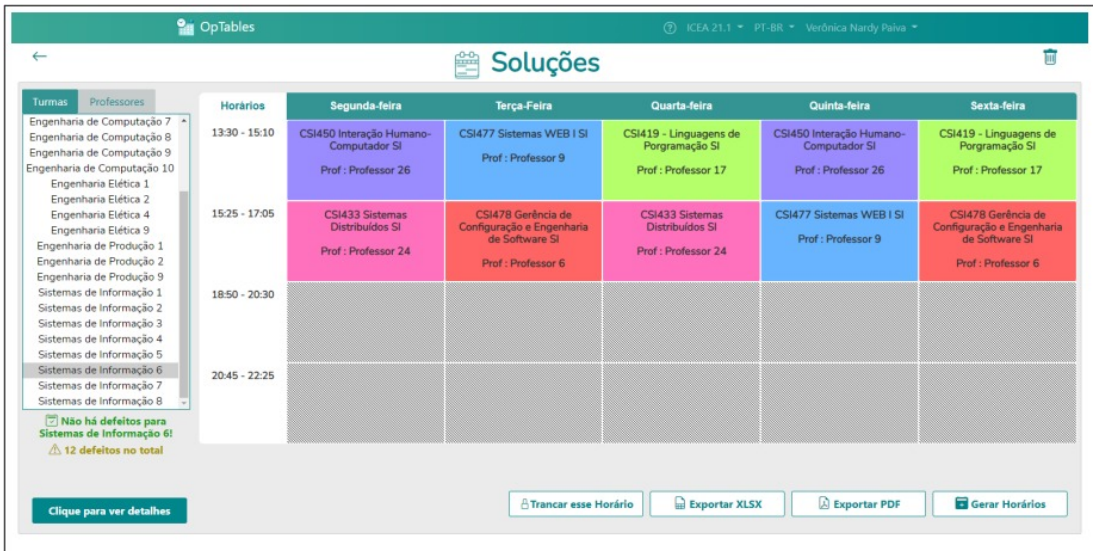

Figura 3 – Tela de visualização de grade curricular

4.4 FET - Free Timetabling Software

O FET (Free Timetabling Software) (DIRR; LALESCU, 2025), figura 4, é uma aplicação open-source para geração automática de horários educacionais. Desenvolvido em C++ com interface gráfica em Qt, o FET implementa algoritmos avançados de otimização combinatória para resolver problemas complexos de alocação de horários.

O software se destaca pela sua capacidade de lidar com numerosas restrições de horários, categorizadas como "rígidas"(que não podem ser violadas) e "flexíveis"(que podem ser parcialmente satisfeitas). Esta abordagem permite grande flexibilidade na modelagem de diversos cenários educacionais.

Entre as principais funcionalidades do FET, destacam-se:

- Suporte a múltiplos idiomas e internacionalização
- Geração de relatórios em HTML com estatísticas detalhadas
- Definição personalizada de períodos de tempo e dias da semana
- Gerenciamento de restrições temporais para professores e turmas
- Alocação de salas com base em critérios de capacidade e equipamentos
- Interface para visualização e edição manual de horários gerados

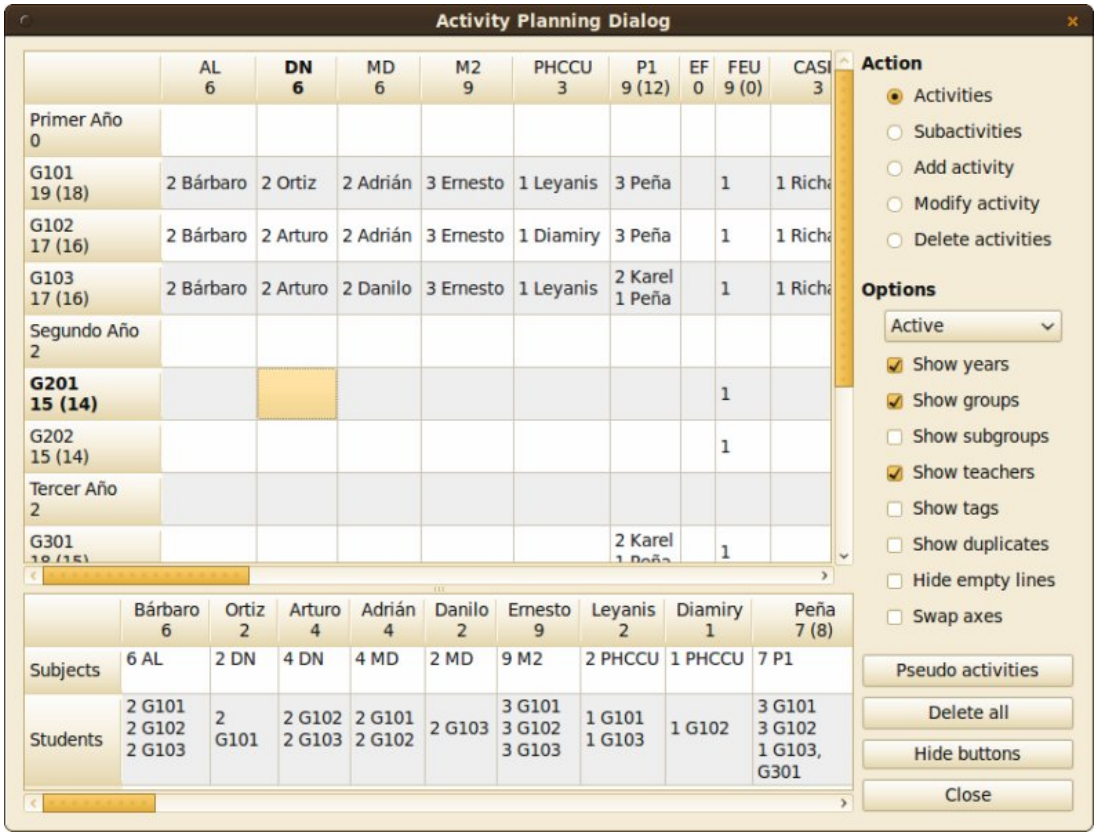

Figura 4 – Interface do FET

4.5 Análise Comparativa de Soluções

Diversas soluções têm sido desenvolvidas para resolver o problema de alocação de horários em instituições de ensino. Entre elas destacam-se: Shiba, Cronos, o sistema proposto por Paiva e o FET (Free Timetabling Software). A Tabela 1 já apresenta um resumo comparativo de suas funcionalidades. No entanto, é importante ir além das características técnicas e realizar uma análise crítica com base em critérios mais amplos.

Tabela 1 – Comparativo entre sistemas de geração de horários

Critério	Shiba	Cronos	FET	OpTables - Paiva	Este Trabalho
Facilidade de uso	Alta	Baixa	Baixa	Alta	Alta
Personalização	Média	Alta	Alta	Alta	Média
Interface gráfica	Intuitiva	Complexa	Complexa	Intuitiva	Intuitiva
Suporte a múltiplos usuários	Sim	Sim	Sim	Sim	Não
Visualização de conflitos	Sim	Sim	Sim	Sim	Sim
Exportação de dados	Sim (PDF)	Sim	Sim	Sim (PDF/Planilha)	Sim (JSON)
Importação de dados	Sim	Sim	Sim	Sim (Planilha)	Sim (JSON)
Instalação necessária	Não	Não	Sim	Não	Não

Fonte: Elaborada pelo autor.

4.5.1 Facilidade de Uso

A usabilidade é um fator determinante na adoção de sistemas. Plataformas como FET e Cronos, embora poderosas, possuem interfaces densas e repletas de parâmetros, o que pode dificultar seu uso por usuários não técnicos. O sistema desenvolvido por Paiva avança nesse sentido ao oferecer uma interface amigável, mas ainda depende de conhecimento técnico para parametrização. A plataforma proposta neste trabalho, por sua vez, adota uma abordagem minimalista, baseada em interação por arrastar e soltar, tornando o processo mais acessível e intuitivo.
Como métrica de usabilidade, foi considerado o **tempo gasto para realizar uma alocação**. Atualmente, esse processo pode variar bastante, mas em geral demanda um tempo elevado, pois a solução existente opera por linha de comando e exige a consideração manual de restrições. A proposta aqui apresentada busca justamente reduzir esse tempo por meio de uma interface visual e interativa.

4.5.2 Licença e Custo

O FET é um software *open-source* sob licença GPL, o que o torna atrativo para instituições com baixo orçamento. Por outro lado, o Cronos é uma plataforma comercial, com custos associados e limitações de personalização. As soluções acadêmicas (Shiba e Paiva) têm distribuição limitada ou não especificada. A plataforma desenvolvida neste trabalho é gratuita, com código aberto e arquitetura modular, o que facilita sua adoção, adaptação e manutenção.

4.5.3 Funcionalidades e Personalização

Em termos de funcionalidades, Cronos se destaca por utilizar inteligência artificial e evitar automaticamente conflitos de horário. O FET também é altamente configurável, permitindo a definição de restrições rígidas e flexíveis. No entanto, ambas exigem curva de aprendizado alta. A solução aqui proposta é mais limitada em termos de automação, mas oferece flexibilidade por meio da manipulação direta da grade, o que, para muitos usuários, pode ser mais eficiente do que navegar por menus complexos.

4.5.4 Escalabilidade e Integração

Soluções como FET e Cronos demonstram maior maturidade em cenários de grande escala, como universidades com múltiplos campi. A proposta deste trabalho ainda não foi testada em

cenários amplos, mas sua arquitetura modular e hospedagem em nuvem permitem expansão e integração com sistemas institucionais, caso necessário.

4.5.5 Considerações sobre os trabalhos relacionados

A análise revela que não existe uma solução superior: cada sistema se posiciona de forma distinta no equilíbrio entre complexidade e usabilidade. O FET e o Cronos oferecem alta automação e múltiplas configurações, mas com interfaces menos acessíveis a usuários não-técnicos. As soluções Shiba e Paiva melhoram a experiência de uso, porém com menor alcance em termos de funcionalidades avançadas. A plataforma proposta neste trabalho encontra seu diferencial na simplicidade operacional e no alinhamento com o fluxo real de trabalho de coordenadores acadêmicos. Ao priorizar a interação direta, torna-se especialmente adequada para a instituição que busca uma solução prática, mesmo que com menor grau de automação inicial. A Tabela 2 apresenta um resumo dessa comparação.

Princípios Aplicados

- **Organização Visual**: separação clara entre área de trabalho e controles
- **Feedback Imediato**: respostas visuais em tempo real
- **Códigos Visuais**: cores e ícones padronizados
- **Simplicidade**: redução da curva de aprendizado

Benefícios para o Usuário

- Interface intuitiva para usuários não técnicos
- Redução de erros através de validação visual
- Processo de alocação mais eficiente
- Flexibilidade para ajustes manuais

Tabela 2 – Análise consolidada dos critérios dos sistemas avaliados

Sistema	**Usabilidade**	**Facilidade de uso**	**Interface**
Shiba	Alta	Alta	Intuitiva
Cronos	Baixa	Média	Complexa
FET	Baixa	Baixa	Complexa
OpTables - Paiva	Alta	Alta	Intuitiva
Este Trabalho	Alta	Alta	Intuitiva

Fonte: Elaborada pelo autor.

5 METODOLOGIA

Para o desenvolvimento deste projeto, foram utilizadas tecnologias visando facilitar o desenvolvimento e valorizar experiências prévias do desenvolvedor. A arquitetura implementada foi projetada para ser robusta, atendendo às necessidades atuais e futuras da aplicação. O conjunto de tecnologias escolhidas prioriza desempenho, segurança e escalabilidade.

5.1 Conjunto de tecnologias para desenvolvimento Frontend

Este trabalho adotou um conjunto específico de tecnologias para o desenvolvimento da interface do usuário, priorizando ferramentas modernas e amplamente aceitas pela comunidade de desenvolvimento web. A seleção baseou-se em critérios como facilidade de manutenção, performance, curva de aprendizado e disponibilidade de recursos da comunidade. As tecnologias selecionadas formam um ecossistema integrado que viabiliza desenvolvimento ágil e código sustentável, fundamentais para um projeto passível de expansão futura.

As três principais ferramentas adotadas foram React.js para a construção da interface, TypeScript para adicionar tipagem estática ao JavaScript, e Tailwind CSS para estilização eficiente. Esta combinação representa um conjunto moderno e consolidado no mercado de desenvolvimento web, proporcionando produtividade no desenvolvimento e facilidade de manutenção do código.

5.1.1 React.js

O React.js surgiu em 2013 como uma biblioteca JavaScript desenvolvida pela equipe do Facebook (atual Meta) para resolver problemas específicos relacionados à construção de interfaces de usuário complexas e dinâmicas. Criado inicialmente por Jordan Walke, o React revolucionou o desenvolvimento web ao introduzir conceitos como componentes reutilizáveis e o Virtual DOM, uma representação em memória da estrutura de elementos HTML que permite atualizações eficientes da interface.

Antes do React, desenvolvedores frequentemente enfrentavam dificuldades para gerenciar o estado de aplicações web complexas, especialmente quando múltiplos elementos da página precisavam ser atualizados simultaneamente. O React resolveu esse problema propondo uma abordagem declarativa, onde o desenvolvedor descreve como a interface deve parecer em cada estado, e a biblioteca se encarrega de fazer as mudanças necessárias de forma eficiente.

Atualmente, o React é uma das bibliotecas JavaScript mais populares do mundo, sendo utili-

zada por grandes empresas como Netflix, Airbnb, Instagram, WhatsApp e milhares de outras organizações. Segundo pesquisas anuais como o Stack Overflow Developer Survey, o React consistentemente aparece entre as tecnologias mais amadas e desejadas pelos desenvolvedores, demonstrando sua relevância no mercado atual.

O React.js (Meta Platforms, Inc., 2025) foi adotado como *framework* principal para o desenvolvimento *frontend*, devido à sua arquitetura baseada em componentes, oferecendo:

- **Renderização eficiente através do Virtual DOM:** permite atualizações rápidas da interface sem recarregar a página inteira;
- **Fluxo de dados unidirecional:** facilita a depuração e torna o comportamento da aplicação mais previsível;
- **Reutilização de componentes:** reduz redundâncias no código e acelera o desenvolvimento;
- **Ecossistema maduro:** conta com inúmeras bibliotecas complementares, incluindo soluções específicas para *drag-and-drop* necessárias neste projeto.

5.1.2 TypeScript

O TypeScript foi criado pela Microsoft em 2012 como uma extensão do JavaScript que adiciona tipagem estática opcional à linguagem. Desenvolvido sob a liderança de Anders Hejlsberg (também criador do C e Delphi), o TypeScript surgiu para resolver limitações do JavaScript em projetos de grande escala, onde a ausência de tipos dificulta a manutenção e pode levar a erros em tempo de execução.

A linguagem JavaScript, embora extremamente flexível e popular, não possui verificação de tipos nativa, o que significa que erros relacionados a tipos de dados só são descobertos quando o código é executado. Em projetos pequenos isso não representa um problema significativo, mas em aplicações maiores pode levar a bugs difíceis de detectar e corrigir.

O TypeScript compila para JavaScript puro, mantendo compatibilidade com qualquer ambiente que execute JavaScript, incluindo navegadores web e servidores Node.js. Grandes empresas como Microsoft, Google e Slack adotaram TypeScript em seus projetos, demonstrando sua eficácia em ambientes de produção. O próprio Angular, framework do Google, foi reescrito em TypeScript, evidenciando a confiança da comunidade na tecnologia.

O TypeScript (Microsoft Corporation, 2025) foi utilizado como linguagem de programação, proporcionando:

- **Detecção precoce de erros:** identifica problemas durante o desenvolvimento, antes da execução do código;
- **Autocompletar e sugestões inteligentes:** melhora significativamente a experiência de desenvolvimento;
- **Documentação implícita através da tipagem:** os tipos servem como documentação viva do código;
- **Refatoração segura:** permite modificar o código com confiança, essencial para a evolução do projeto.

5.1.3 Tailwind CSS

O Tailwind CSS foi criado em 2017 por Adam Wathan, Jonathan Reinink, David Hemphill e Steve Schoger como uma alternativa aos frameworks CSS tradicionais como Bootstrap. Surgiu da frustração dos criadores com a rigidez dos frameworks existentes, que frequentemente impunham estilos pré-definidos difíceis de customizar sem sobrescrever extensivamente o CSS.
Diferentemente dos frameworks CSS tradicionais que fornecem componentes prontos (como botões e modais estilizados), o Tailwind adota uma abordagem "utility-first", fornecendo classes pequenas e específicas que fazem apenas uma coisa. Por exemplo, em vez de uma classe "btn-primary"que define múltiplos estilos, o Tailwind oferece classes como "bg-blue-500"(fundo azul), "text-white"(texto branco) e "px-4"(padding horizontal), permitindo maior flexibilidade na composição visual.
Esta abordagem inicialmente pode aparentar maior complexidade, mas proporciona vantagens significativas: designs únicos e personalizados, menor necessidade de escrever CSS customizado, e melhor performance final devido à eliminação automática de estilos não utilizados. O Tailwind rapidamente ganhou adoção em empresas como GitHub, Netflix, Shopify e muitas startups, tornando-se uma das ferramentas de estilização mais populares no desenvolvimento web moderno.
O Tailwind CSS (Tailwind Labs Inc., 2025) foi implementado para estilização, resultando em:

- **Aceleração significativa no desenvolvimento visual:** classes utilitárias permitem estilização rápida sem sair do HTML;
- **Consistência no design:** sistema de design padronizado garante uniformidade visual;
- **Otimização automática para produção:** remove automaticamente CSS não utilizado, reduzindo o tamanho final;

- **Personalização adaptada ao projeto:** facilita a criação de uma identidade visual específica para a aplicação.

5.2 Controle de Versão

O controle de versão é uma prática no desenvolvimento de software que permite acompanhar mudanças no código ao longo do tempo, facilitando a colaboração entre desenvolvedores e oferecendo segurança contra perda de código. O conceito surgiu da necessidade de gerenciar diferentes versões de arquivos em projetos de software, especialmente quando múltiplos desenvolvedores trabalham no mesmo código.

O Git foi criado em 2005 por Linus Torvalds, o mesmo criador do sistema operacional Linux, como uma solução para gerenciar o desenvolvimento do kernel do Linux. Antes do Git, o projeto utilizava um sistema proprietário que se tornou inadequado para as necessidades da comunidade. Linus desenvolveu o Git em apenas algumas semanas, priorizando velocidade, integridade dos dados e suporte a fluxos de trabalho distribuídos.

Diferentemente de sistemas de controle de versão centralizados, onde existe um servidor central que armazena todo o histórico, o Git é distribuído, significando que cada desenvolvedor possui uma cópia completa da história do projeto. Isso proporciona maior segurança, pois não há um ponto único de falha, e possibilita trabalho offline.

O GitHub, lançado em 2008, revolucionou ainda mais o desenvolvimento colaborativo ao fornecer uma plataforma web para hospedar repositórios Git. Com recursos como pull requests, issues, e integração contínua, o GitHub se tornou a plataforma mais popular para desenvolvimento de software open source e corporativo. Atualmente, milhões de desenvolvedores ao redor do mundo utilizam Git e GitHub diariamente, desde projetos pessoais até grandes corporações como Google, Microsoft e Facebook.

Para o gerenciamento do código-fonte deste projeto, foram empregadas:

- **Git (Software Freedom Conservancy, 2025):** sistema distribuído que permite versionamento eficiente, controle de mudanças e colaboração segura;
- **GitHub (GitHub, Inc., 2025):** plataforma que oferece armazenamento seguro na nuvem, ferramentas de colaboração e integração com outras ferramentas de desenvolvimento.

5.3 Plataformas de Hospedagem em Nuvem

A computação em nuvem transformou radicalmente a forma como aplicações web são desenvolvidas e disponibilizadas. Tradicionalmente, hospedar uma aplicação web exigia configurar e manter servidores físicos, instalar sistemas operacionais, configurar servidores web, gerenciar bancos de dados e lidar com questões de segurança e escalabilidade. Este processo era complexo, caro e exigia conhecimento técnico especializado.

As plataformas de hospedagem em nuvem modernas surgiram para simplificar este processo, oferecendo infraestrutura gerenciada onde desenvolvedores podem focar no código da aplicação em vez de configurações de servidor. Empresas pioneiras como Amazon (AWS), Google (Google Cloud) e Microsoft (Azure) estabeleceram os fundamentos desta revolução tecnológica.

Mais recentemente, surgiram plataformas especializadas em desenvolvedores frontend, como Netlify e Vercel, que se concentram especificamente em aplicações web modernas. Estas plataformas oferecem deployment automático e integração nativa com repositórios Git, simplificando o processo de publicação de aplicações web.

A Vercel, criada em 2015, foi desenvolvida especificamente para aplicações frontend modernas, incluindo frameworks como React, Vue e Angular. A plataforma se destaca por sua facilidade de uso e foco na experiência do desenvolvedor, oferecendo deployment em segundos e preview automático para cada mudança no código.

A plataforma *Vercel* (Vercel Inc., 2025) foi adotada para hospedagem e *deployment* contínuo, garantindo:

- **Integração nativa com repositórios Git:** cada commit gera automaticamente uma versão de preview da aplicação;
- **Monitoramento em tempo real:** oferece métricas de performance e analytics de uso da aplicação;
- **Escalabilidade automática:** ajusta recursos automaticamente conforme a demanda, sem configurações complexas;
- **CDN global e HTTPS automático:** garante acesso rápido e seguro de qualquer parte do mundo.

5.4 Etapas do Planejamento e Decisões Metodológicas

O desenvolvimento da plataforma web para alocação de horários foi conduzido de forma iterativa e centrada no usuário. A metodologia adotada seguiu princípios da engenharia de software ágil, com ciclos curtos de desenvolvimento e validação contínua. A seguir, são descritas as principais etapas do planejamento, bem como as justificativas das decisões tomadas em cada uma delas.

5.4.1 Levantamento de Requisitos

A primeira fase consistiu na identificação das necessidades do público-alvo, conduzida por meio de consultas com especialista na área de alocação de horários e usuário do modelo de alocação existente. Este processo envolveu reuniões periódicas com o desenvolvedor do algoritmo de alocação via linha de comando, que forneceu informações valiosas sobre as limitações técnicas e operacionais da solução atual.

Adicionalmente, foi realizada consulta com o professor que lida regularmente com o processo de alocação de horários, permitindo identificar as principais dificuldades enfrentadas no dia a dia. Os especialistas consultados, incluindo profissionais com experiência em sistemas acadêmicos, contribuíram com feedback sobre funcionalidades essenciais e possíveis melhorias para otimizar o fluxo de trabalho.

Nesta etapa foram levantadas as principais limitações dos métodos atuais, como ferramentas com excesso de parâmetros técnicos, falta de feedback visual imediato, necessidade de conhecimento em linha de comando, e dificuldade para realizar ajustes manuais rápidos durante o semestre letivo. Os múltiplos ciclos permitiram refinar constantemente os requisitos, garantindo alinhamento entre as necessidades reais dos usuários e as funcionalidades propostas.

5.4.2 Ciclos de desenvolvimento incremental

O desenvolvimento foi conduzido em ciclos curtos, com entregas parciais e feedback contínuo. Empregou-se a stack React.js, TypeScript e Tailwind CSS, garantindo modularidade, escalabilidade e boa performance da interface.

Decisão: priorizou-se desenvolver primeiramente o frontend, simulando dados e interações, antes de integrar com o backend. Isso possibilitou refinar a interface e validar conceitos de design antes de investir tempo em integrações complexas.

6 RESULTADOS

Este capítulo apresenta os resultados obtidos com o desenvolvimento da plataforma web para alocação de horários acadêmicos. A aplicação desenvolvida demonstra a viabilidade de criar uma interface intuitiva e eficiente para o gerenciamento de grades horárias, integrando funcionalidades essenciais como visualização interativa, cadastro simplificado de atividades, persistência de dados e controle flexível de restrições.

A integração das tecnologias selecionadas resultou em uma aplicação que pode ser destacada por:

1. Tempo de carregamento inicial reduzido;
2. Base de código sustentável e preparada para expansões futuras;
3. Processo de desenvolvimento ágil com rápida iteração e *feedback*;
4. Interface amigável e otimizada.

Os resultados são apresentados através da demonstração das principais funcionalidades implementadas, análise da interface desenvolvida e avaliação dos aspectos de usabilidade alcançados.

6.1 Interface Principal e Visualização da Grade

A Figura 5 apresenta a tela inicial do sistema desenvolvido. À esquerda, há uma barra lateral contendo o botão *Adicionar Card* e a lista de cards disponíveis. Cada card representa uma aula, exibindo o curso, nome do professor, disciplina e quantas vezes aquela aula se repetirá na semana. Os cards podem ser arrastados e soltos sobre a tabela principal utilizando o recurso de arrastar e soltar.

A tabela de horários constitui o elemento central da interface, organizando todos os horários a serem gerenciados de forma visual e intuitiva. Esta grade representa a estrutura temporal de uma semana acadêmica, facilitando a compreensão imediata da distribuição das atividades. A organização em linhas e colunas permite que coordenadores visualizem rapidamente conflitos, lacunas na programação e a distribuição equilibrada das disciplinas ao longo da semana.

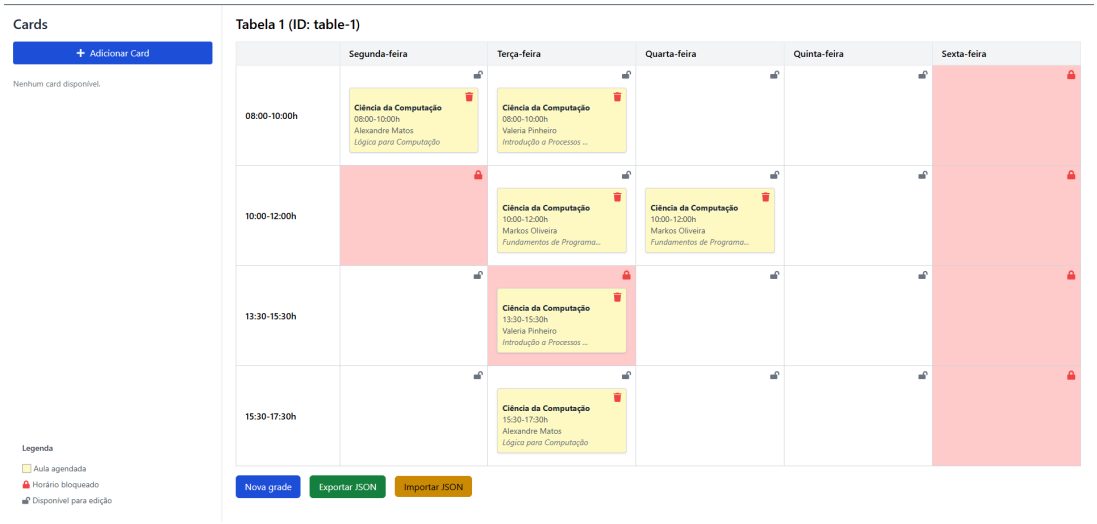

Figura 5 – Tela inicial do sistema com a grade preenchida e cards alocados

No centro da interface, encontra-se a grade curricular representada por uma tabela. As colunas indicam os dias da semana e as linhas os intervalos de horário. O sistema permite alocar cards nas células da tabela, possibilitando a construção de uma grade personalizada. Cada célula da grade pode assumir três estados visuais: **aula agendada** (card em amarelo alocado em uma célula), **horário bloqueado** (célula em vermelho com ícone de cadeado) e **disponível para edição** (célula com ícone de cadeado cinza aberto).

Cada card inserido na tabela exibe o curso, professor, disciplina, e o intervalo de horário correspondente. O usuário pode excluir um card da tabela clicando no ícone de lixeira presente no canto superior direito de cada bloco.

Na barra lateral esquerda, encontra-se a legenda que explica o significado das cores e ícones das células. Além disso, há três botões funcionais: **Nova grade**, que cria uma nova tabela abaixo da tabela atual; **Exportar JSON**, que permite salvar a configuração atual da grade em um arquivo `.json`; e **Importar JSON**, que possibilita carregar uma configuração previamente salva.

6.2 Modal de cadastro

Com o intuito de facilitar o cadastro de novas atividades acadêmicas, foi desenvolvido um modal dedicado que simplifica o processo de inserção de disciplinas na grade. Esta abordagem elimina a necessidade de navegação entre diferentes telas, mantendo o usuário no contexto da visualização da grade enquanto realiza o cadastro.

A Figura 6 exibe o modal de cadastro acionado ao clicar no botão *Adicionar Card*. Este modal contém um formulário onde o usuário pode cadastrar uma nova disciplina, preenchendo

informações como curso, professor, nome da disciplina, quantidade de dias em que será oferecida e os respectivos horários.

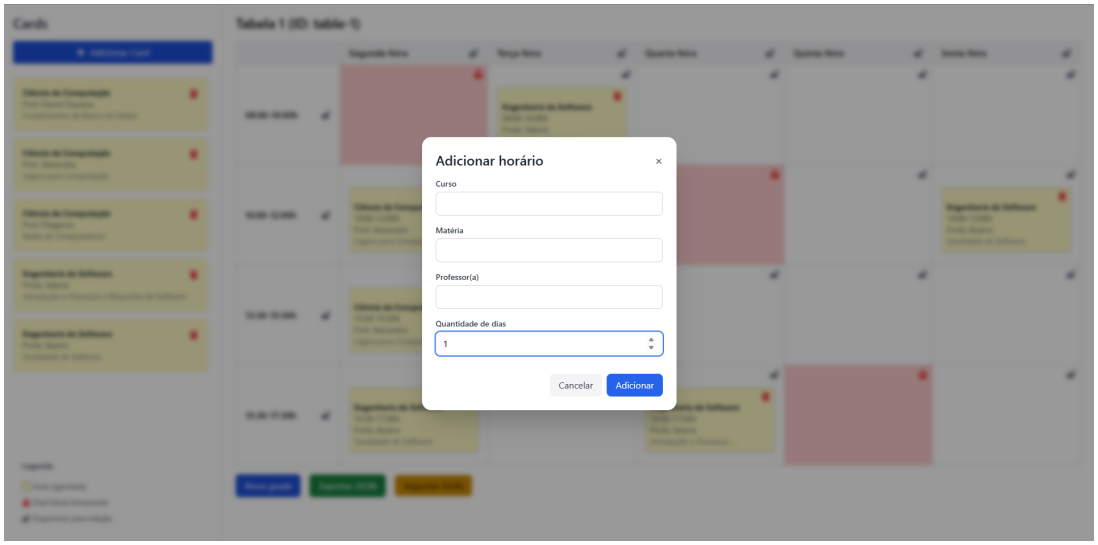

Figura 6 – Modal de cadastro de horário

Após o envio do formulário, um novo card é gerado e adicionado à barra lateral, ficando disponível para ser alocado na tabela. O modal foi projetado com uma interface clara, permitindo ao usuário fechar a janela com facilidade ou realizar a inserção de maneira direta.

6.3 Funcionalidades de exportação e importação

O sistema oferece suporte para persistência dos dados da grade curricular por meio dos botões **Exportar** e **Importar JSON**, localizados abaixo da tabela. Ao clicar em **Exportar JSON**, o usuário pode baixar um arquivo contendo a estrutura atual da grade em formato `.json`, o que permite armazenar a configuração criada.

O botão **Importar JSON** possibilita o carregamento de um arquivo previamente salvo, restaurando o estado da grade, incluindo os cards, os horários alocados e o status de cada célula. Essa funcionalidade é útil tanto para continuidade do trabalho quanto para compartilhamento de grades.

6.4 Bloqueio de horários

A realização de ajustes manuais constitui um aspecto fundamental para coordenadores solucionarem questões específicas de alocação, uma vez que sempre surgem necessidades de pequenos ajustes durante o processo de elaboração da grade horária. Situações como indisponibilidade

temporária de professores, conflitos com eventos institucionais, manutenção de equipamentos ou restrições específicas de salas demandam flexibilidade para modificações pontuais que algoritmos automatizados podem não contemplar adequadamente.

Reconhecendo esta necessidade prática, o sistema implementa uma funcionalidade robusta de bloqueio de horários. Algumas células da tabela aparecem com fundo vermelho e um ícone de cadeado, indicando que estão **bloqueadas** para edição. Estas células não permitem a alocação de cards, servindo para representar restrições reais da instituição, como horários indisponíveis por falta de sala ou conflito com outros eventos.

Além disso, há células com um ícone de cadeado em cinza, indicando que estão **disponíveis para edição**. Nestes casos, os usuários podem arrastar cards e posicioná-los livremente, respeitando apenas a lógica de não sobreposição.

Por fim, quando um card está em uma célula e essa célula é bloqueada, o card alocado nesta, por sua vez, não estará disponível para ser movido para outra célula. Ou seja, o card é fixado naquele horário até que o usuário desbloqueie aquele horário.

6.5 Análise de usabilidade

Durante o desenvolvimento do sistema, buscou-se adotar princípios de usabilidade para garantir uma experiência de usuário intuitiva, eficiente e satisfatória. A interface foi projetada com foco na clareza das ações disponíveis e na redução da curva de aprendizado, permitindo que mesmo usuários sem familiaridade com sistemas de alocação possam utilizá-lo com facilidade.

7 CONCLUSÃO E TRABALHOS FUTUROS

O desenvolvimento desta plataforma web para alocação de horários representa uma contribuição relevante para a otimização de processos administrativos acadêmicos. A solução proposta tem por objetivo transformar um procedimento atualmente complexo em uma experiência mais acessível e eficiente, mantendo o foco na simplicidade da interface e na intuitividade das interações.

A escolha de tecnologias como React.js, TypeScript e Tailwind CSS proporcionou um desenvolvimento ágil e organizado e também definiu bases sólidas para a manutenção e expansão futura da aplicação. A arquitetura implementada garante desempenho adequado e permite a modularização do sistema, favorecendo o acréscimo de novas funcionalidades à medida que as necessidades dos usuários evoluam.

Durante o desenvolvimento, a abordagem metodológica adotada foi fundamental para entregar um produto que atendesse às demandas de coordenadores de curso e demais usuários. Funcionalidades como *drag-and-drop*, bloqueio de horários e exportação/importação de dados em JSON foram implementadas com o objetivo de equilibrar flexibilidade e definir um formato para implementação da conextão com o back-end.

Como próximos passos, tem-se a **integração com o algoritmo de alocação já desenvolvido**, o qual atualmente opera via linha de comando. Essa conexão com o *backend* permitirá que a plataforma não apenas ofereça uma interface gráfica eficiente, mas também se torne capaz de realizar **sugestões automáticas de alocação com base em restrições definidas pelo usuário**, como disponibilidade de professores, horários bloqueados e capacidade das salas.

Com isso, mesmo em sua versão atual, a plataforma representa um avanço importante, e sua continuidade com foco em integração com *backend* e automação consolidará seu papel como ferramenta estratégica no contexto educacional.

REFERÊNCIAS

AHUJA, R. K.; MAGNANTI, T. L.; ORLIN, J. B. **Network Flows: Theory, Algorithms, and Applications**. [S.l.]: Prentice Hall, 1993.

BECK, K. **Extreme programming explained: embrace change**. [S.l.]: Addison-Wesley Professional, 2000.

CARTER, M. W.; LAPORTE, G. Recent developments in practical course timetabling. In: **Proceedings of the International Conference on the Practice and Theory of Automated Timetabling**. [S.l.]: Springer, 1996.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**. [S.l.]: MIT Press, 2022.

Cronostime Table. **Cronostime Table - Plataforma de Gerenciamento de Horários Escolares**. 2025. Disponível em: https://www.cronostimetable.com/. Acesso em: 09 mar. 2025.

DIRR, V.; LALESCU, L. **FET - Free Timetabling Software**. 2025. Software open-source para geração automática de horários educacionais. Disponível em: https://www.lalescu.ro/liviu/fet/. Acesso em: 10 mar. 2025.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. [S.l.]: W. H. Freeman, 1979.

GitHub, Inc. **GitHub Documentation**. 2025. Disponível em: https://docs.github.com. Acesso em: 10 mar. 2025.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. [S.l.]: University of Michigan Press, 1975.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **Science**, v. 220, n. 4598, p. 671–680, 1983.

LEWIS, R. A survey of metaheuristic-based techniques for university timetabling problems. **OR Spectrum**, v. 30, p. 167–190, 2008.

Meta Platforms, Inc. **React Documentation**. 2025. Disponível em: https://react.dev/. Acesso em: 10 mar. 2025.

Microsoft Corporation. **TypeScript Documentation**. 2025. Disponível em: https://www.typescriptlang.org/docs/. Acesso em: 10 mar. 2025.

NIELSEN, J. **Usability engineering**. [S.l.]: Morgan Kaufmann, 1994.

PAIVA, V. N. **OpTables - um sistema web para a programação de horários educacionais**. 89 p. — Universidade Federal de Ouro Preto, João Monlevade, 2021. Monografia (Graduação em Engenharia de Computação) - Instituto de Ciências Exatas e Aplicadas. Disponível em: http://www.monografias.ufop.br/handle/35400000/5214. Acesso em: 22 ago. 2025.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 4. ed. [S.l.]: Pearson, 2020.

SHIBA, Y. A. **Sistema Web Para Programação de Horários e Alocação de Salas em Universidades**. Dissertação (Mestrado) — Universidade Federal de São Paulo, São Paulo, 2023. Disponível em: https://repositorio.unifesp.br/server/api/core/bitstreams/e0fa236d-e2e9-4a08-bcb2-0cdce8d5d832/content. Acesso em: 09 mar. 2025.

Software Freedom Conservancy. **Git Documentation**. 2025. Disponível em: https://git-scm.com/doc. Acesso em: 10 mar. 2025.

SOMMERVILLE, I. **Software engineering**. 10. ed. [S.l.]: Pearson, 2015.

Tailwind Labs Inc. **Tailwind CSS Documentation**. 2025. Disponível em: https://tailwindcss.com/docs. Acesso em: 10 mar. 2025.

Vercel Inc. **Vercel Documentation**. 2025. Disponível em: https://vercel.com/docs. Acesso em: 10 mar. 2025.