A Necessidade da Reformulação dos Websites de Administração Pública

Enzo Humberto Rocha e Silva[1], Fabiana Camargo Bedreski[1]

[1]Centro Universitário Campo Real

Rua Comendador Norberto, 1299 - Santa Cruz - Guarapuava - PR - Brasil

{engs-enzosilva@camporeal.edu.br, prof_fabianabedreski@camporeal.edu.br}

Resumo

O acesso das informações dispostas em websites e sistemas, deve ter parâmetros de acessibilidade de maneira universal, em que todos os usuários, independentes dos dispositivos que usam, consigam usufruir plenamente dos dados aos quais buscam, também, estes mesmos dados, devem estar dispostos de maneira orgânica e organizada, para obtenção de melhor usabilidade dos websites. Tendo isso como propósito, este artigo visa estabelecer uma proposta de reformulação de um conjunto de aplicações, pertencentes a uma instituição configurada, como por exemplo, Administração Pública Indireta, com o objetivo de modernizar e otimizar sua estrutura e mecanismo de comunicação tecnológicos, aprimorando a usabilidade do usuário, otimização de processos e parâmetros de segurança, usando ferramentas e tecnologias modernas para melhor desempenho, juntamente com a utilização de metodologias de boas práticas no âmbito do desenvolvimento. Essa reformulação proposta, busca uma análise detalhada das aplicações existentes, observando as limitações estruturais e funcionais que impactam de forma direta na experiência do usuário, e também, na eficiência operacional da instituição. Busca-se, dessa forma, a redefinição de fluxos, componentes e interfaces de modo a torná-los mais intuitivos, responsivos e acessíveis.

Palavras-chave: Reformulação. Administração Pública. Websites.

Abstract

The access to information available on websites and systems must follow universal accessibility parameters, ensuring that all users, regardless of the devices they use, can fully benefit from the data they seek. Furthermore, this information should be presented in an organic and organized manner to enhance the overall usability of websites. With this purpose, this article aims to establish a proposal for the reformulation of a set of applications belonging to an institution configured as an Indirect Public Administration entity, with the goal of modernizing and optimizing its technological structure and communication mechanisms. The proposal focuses on improving user usability, process optimization, and security parameters through the use of modern tools and technologies for better performance, along with the adoption of best-practice methodologies in the development process.

This proposed reformulation seeks a detailed analysis of the existing applications, identifying structural and functional limitations that directly affect both user experience and the institution’s operational efficiency. Therefore, it aims to redefine workflows, components, and interfaces to make them more intuitive, responsive, and accessible.

Keywords: Reformulation. Public Administration. Websites.

1. Introdução

A presença digital da Administração Pública se estabeleceu como uma das principais formas de comunicação entre órgãos administrativos e a sociedade. Websites governamentais são hoje a porta de entrada para o acesso a serviços, informações e políticas públicas, desempenhando um papel central na aproximação entre os cidadãos e instituições. Entretanto, diversos destes portais ainda apresentam problemas estruturais, visuais e funcionais, dificultando a navegação, a acessibilidade e a transparência nos recursos oferecidos.

Diante deste cenário, a reformulação destes serviços ofertados torna-se uma necessidade estratégica para a modernização do Estado para estabelecer um vínculo com o cidadão. A reformulação de um sistema de administração pública impacta diretamente na qualidade da democracia, ao facilitar o acesso às informações, agregar na participação cidadã nas políticas públicas e fortalecer a confiança nas instituições, além de promover a inclusão digital de todos os indivíduos da população.

O objetivo geral deste trabalho, é propor um modelo para o desenvolvimento de um website moderno e acessível para um serviço municipal de Administração Pública, exibindo na prática os benefícios de uma reformulação. Baseada em aplicações modernas e contemporâneas.

Para atingir este propósito final, se estabelecem os seguintes pontos: analisar as deficiências mais comuns em portais governamentais existentes; implementar princípios de *design patterns* centrados no usuário e nos conceitos de acessibilidade *web*; demonstrar como a nova estrutura pode melhorar a experiência do cidadão e a eficiência administrativa, construindo assim, um novo modelo replicável para a modernização da presença digital do setor público.

Este artigo encontra-se dividido em seções. Após a seção de introdução, apresenta-se a Seção 2 de Referencial Teórico, responsável por expor as bases teóricas dos termos e ferramentas utilizados neste artigo. Prosseguindo na Seção 3, contempla-se o Estado da Arte, cujo seu papel é demonstrar as pesquisas e projetos já existentes, que possuem correlação com o tema abordado neste artigo, possibilitando a identificação de lacunas, as quais possam ser exploradas e solucionadas. A Seção 4, Metodologia, descreve as métricas seguidas para o desenvolvimento do protótipo e do artigo. A Seção de Resultados, Seção de número 5, tem como o seu objetivo demonstrar os resultados obtidos durante o processo de criação e de implementação do projeto proposto. Na Seção 6, Considerações Finais, em que se entende e interpreta os resultados obtidos na Seção anterior, correlacionando com os objetivos estabelecidos na Metodologia, também, destacando pontos de melhoria e objetivos posteriores à conclusão do artigo. Por fim, as referências e apêndices anexados no final do documento, para observação da origem das informações retiradas para embasamentos de citações e suportes argumentativos presentes.

2. Referencial Teórico

Nesta seção do documento, serão apresentados os embasamentos teóricos e referenciais necessários para a compreensão do tema e de suas ferramentas utilizadas para a realização do trabalho de conclusão de curso, contemplando suas definições e características necessárias para o entendimento de sua aplicação proposta para este artigo.

2.1. Administração Pública

A Administração Pública consiste em um conjunto de órgãos, entidades ou instituições, com o objetivo de ser responsável pela gestão e administração de políticas públicas que servem à sociedade, por meio de processos e atividades desenvolvidas pelo Estado (ALMG, s.d.). Podemos compreender a administração pública como um grupo de agentes e organizações responsáveis por executar ações que visam o interesse público, com o objetivo de garantir a integridade do bem-estar social, garantir direitos e atender às demandas coletivas, sendo dividida em duas categorias: administração pública direta e administração pública indireta.

2.1.1. Administração Pública Direta

Entende-se por administração pública direta, os órgãos que respondem diretamente ao Poder Executivo, em nível federal, estadual e municipal, não possuindo uma entidade jurídica separada, funcionando apenas seguindo a hierarquia governamental.

2.1.2. Administração Pública Indireta

Em contraparte à administração pública direta, a administração pública indireta é formada por instituições que possuem personalidade jurídica própria, detentoras de um CNPJ (Cadastro Nacional da Pessoa Jurídica), funcionando de forma autônoma e descentralizada do ente federativo, entretanto, ainda sujeitas à fiscalização e controle do Estado.

2.2. Websites

Um *website*, sendo uma plataforma digital acessada a partir de navegadores na internet sendo providos por um domínio, se define por um conjunto de páginas *web* agrupadas e interligadas, fornecendo conteúdos tais como: textos, imagens, vídeos, formulários, conexões, etc.

> Um *website* é uma plataforma online que permite a divulgação de informações, produtos ou serviços de uma empresa, organização ou indivíduo. Ele é composto por diversas páginas que podem conter textos, imagens, vídeos, formulários de contato, entre outros elementos interativos. Cada página de um *website* é acessível através de um navegador de internet, como o Google Chrome, Mozilla Firefox ou Safari. Para acessar uma página específica, basta digitar o endereço único do site, conhecido como URL, na barra de endereços do navegador (CAMARA, 2024).

Podem servir diversas funcionalidades para meios públicos, empresariais ou individuais, desde fornecer informações, relatar noticiários, prover entretenimento, estabelecer meios de comunicação, até mesmo facilitar o comércio eletrônico, tornando, hoje em dia, ferramentas essenciais para o funcionamento de uma sociedade digital.

2.3. Reformulação

Segundo o Dicionário Priberam da Língua Portuguesa (2024), “reformulação” é o “ato ou efeito de reformular; nova formulação.”, ou seja, para trabalhar com a reformulação de *software*, necessitamos ter uma base que vai ser reestruturada para obter um resultado de

melhor desempenho. Isso pode ser realizado de diversas maneiras, tais como, refatoração de um código para padronizar e tornar mais legível e funcional, atualização de interfaces gráficas e visuais, prototipação de testes manuais e automatizados, etc.

2.4. Banco de Dados

Define-se um banco de dados, como sendo uma coleção sistemática e organizada de informações e dados, estruturados e armazenados de maneira eletrônica em um servidor (ORACLE, 2020). Um banco de dados, pode armazenar todo tipo de informação em diferentes tipagens, tais como em formato de texto, formato numérico e formato booleano, cada um com suas respectivas normas e regras de funcionamento para a execução de inserção de dados. Para seu manuseio e controle, geralmente é utilizado um SGBD (Sistema Gerenciador de Banco de Dados).

2.4.1. Sistema Gerenciador de Banco de Dados

Um SGDB (Sistema Gerenciador de Banco de Dados, do inglês *Database System Management*) é um mecanismo para gerenciar, processar, editar e remover dados e tabelas inseridos em um banco de dados, sendo um sistema essencial para desenvolver softwares. Sua estruturação típica, tem como base um modelo de tabelas para representar um objeto, colunas para representar as características desse objeto e os dados em cada coluna são as informações desse objeto. As tabelas, são conectadas umas às outras, por meio de relacionamentos entre si, passando características de uma tabela para outra.

2.4.2. ORM (Object-Relational Mapping)

Object-Relational Mapping (ORM), ou em português, mapeamento objeto-relacional, é uma técnica para aproximar o paradigma de desenvolvimento de aplicações orientadas a objetos ao paradigma do banco de dados relacional (FONSECA, Elton. 2024). Um ORM, obtém a proposta de facilitar a comunicação de um banco de dados e o código da aplicação. O ORM, se propõe a permitir que o desenvolvedor trabalhe com consultas de objetos de forma mapeada e direta no código em desenvolvimento, realizando *scripts* de consultas e inserções de forma mais flexível.

2.5. Javascript

Javascript é uma linguagem de programação moderna, executada e suportada pela grande maioria dos navegadores atuais para seu funcionamento, tais como o Google[1] e Firefox[2], etc. Seus códigos, funcionam diretamente nos navegadores em que é suportada, o que facilita a interação com o usuário. “Javascript nos permite fazer scripts do conteúdo HTML (*HyperText Markup Language*) e da apresentação CSS (*Cascading Style Sheets*) de documentos em navegadores Web, mas também nos permite definir o comportamento desses documentos com rotinas de tratamento de evento.” (FLANAGAN, 1996, p. 10). O seu principal objetivo, é de melhorar a experiência de usuário ao navegar em uma página *Web*,

[1] https://www.google.com/
[2] https://www.firefox.com/

trazendo funcionalidades que fazem com que um *layout* de *website* com elementos estáticos e sem alteração, se tornem um *website* dinâmico e interativo com o usuário.

2.5.1. Bibliotecas Javascript

Desenvolvidas como complementação no ecossistema do *Javascript*, são uma forma de aumentar a gama de possibilidades de utilização em sistemas, desenvolvedores fabricam *scripts* pré-produzidos relacionados à uma funcionalidade e disponibilizam em forma de componentes exportáveis, aumentando o potencial de desenvolvimento e diminuindo a necessidade de escrever *scripts* elaborados para criação de componentes.

2.5.2. Typescript

Desenvolvido pela empresa Microsoft com o propósito de ser uma extensão ou melhoria do *Javascript*, o *Typescript*[3] surge com funcionalidades adicionais que não estão presentes nativamente no *Javascript* ou que possuem métodos de utilização muito complexos, tais como tipagem e suporte a programação orientada a objetos.

> O TypeScript tem ganhado popularidade como uma escolha sólida para projetos que demandam código confiável, manutenível e escalável. Esses benefícios são alcançados graças às características da linguagem, como a tipagem estática e os recursos de orientação a objetos, que permitem aos desenvolvedores detectar erros de forma antecipada, melhorar a clareza e legibilidade do código e facilitar sua manutenção ao longo do tempo. Além disso, o TypeScript oferece suporte a ferramentas de desenvolvimento poderosas, como autocompletar e refatoração, que ajudam a acelerar o processo de desenvolvimento e garantir uma base de código mais robusta (MORORÓ, 2024, p.32)

Possui imensa utilidade em projetos de larga escala, pois possibilita o descobrimento de eventuais falhas de resposta do código de forma muito mais estável e prática.

2.6. ReactJS

Criado e mantido pelo Facebook/Meta em 2011 e disponibilizado para o público geral em 2013, o *ReactJS*[4] É amplamente conhecido e utilizado na área de desenvolvimento de *software*, pois se trata de uma biblioteca de *Javascript* com o intuito de facilitar o desenvolvimento de interfaces *web*, com as suas marcas registradas de funcionalidades: a componentização, a reatividade e o arquivo JSX (*JavaScript XML*). “O React não requer o uso do JSX. Porém, a maioria das pessoas acha prático como uma ajuda visual quando se está trabalhando com uma UI (Interface de usuário, do inglês *User Interface*) dentro do código em JavaScript. Ele permite ao React mostrar mensagens mais úteis de erro e aviso.” (REACTJS.ORG, 2014).

2.6.1. Componentização

Método de separação de pedaços de código em blocos separados, tornando o código mais legível e flexível para cada componente com suas funcionalidades e *design* de interface

[3] https://www.typescriptlang.org/
[4] https://react.dev/

devidamente separadas de um arquivo principal. Cada componente é independente, podendo conter estilizações próprias, arquivos de testes e utilização única. Essa metodologia respeita padrões de desenvolvimento de sistemas, estruturando bem um sistema e evitando repetição por meio da reutilização de código, podendo obter, também, manutenção facilitada para encontrar possíveis erros e incongruências.

2.6.2. Reatividade

Mecanismo de atualização em tempo real dos objetos na interface do usuário, isto é, quando o objeto recebe um novo valor ou informação, atualiza automaticamente na tela, sem a necessidade de reinicialização da página para obter os dados atualizados. Um importante aspecto para se manter nos *websites* nos dias atuais, pois o *software*, tem o objetivo de ser dinâmico e interativo com o usuário para melhorar a performance.

2.7. NodeJS

Inserido na área da tecnologia em 2009, o NodeJS[5] é uma ferramenta que possibilita a execução do Javascript fora dos navegadores, permitindo a criação de aplicações com Javascript sem a necessidade de manter em um sistema de navegação *web*. "Como uma execução de JavaScript assíncrona orientada a eventos, a Node.js foi desenhada para construir aplicações de rede escaláveis." (NODEJS, s.d.). Em comparação com outras tecnologias tradicionais, o NodeJS se destaca por sua leveza, pois não exige muitos recursos computacionais para conseguir se manter estável e em ambiente de funcionamento contínuo. Uma das suas melhores vantagens em destaque, é seu repertório de complementações com o NPM (Gerenciador de Pacotes do Node, do inglês *Node Package Manager*), sendo o NPM, o maior repositório de *softwares* em escala global, possibilitando o NodeJS ser viável em qualquer situação.

3. Estado da Arte

Neste capítulo, abordaremos a pesquisa e a análise de artigos correlacionados ao tema, obtendo um panorama geral do assunto tratado. Para obter as informações utilizadas nesta seção, foi utilizada a ferramenta *Google Acadêmico*[6], por meio de palavras-chave e referências como mecanismo de busca. A seguir, serão listados os artigos relacionados encontrados para a análise.

Primeiramente, foi analisado um artigo publicado por um órgão governamental, com fonte e dados retirados diretamente da Ascom Saeb (Assessoria de Comunicação da Secretaria da Administração do Estado). Após a primeira análise, segue-se o estudo da arte se baseando em trabalhos acadêmicos e artigos publicados.

3.1. Strings de Busca

Para a obtenção de dados acadêmicos e artigos publicados correlatos ao tema abordado, foi utilizado o método de busca por "*strings*" em ferramentas de busca como o *Google Acadêmico*.

---

[5] https://nodejs.org/pt
[6] https://scholar.google.com/

O Quadro 1 demonstra as "*strings*" utilizadas para a busca dos dados encontrados:

Quadro 1 - Strings de Busca

String	Expressão	Resultados
1	("Reformulação" OR "Remodelação" OR "Reformulação")	376.000
2	("Reformulação Websites" OR "Reformulação Sistema")	105.000
3	("Acessibilidade Web" OR "Reformulação, Acessibilidade")	104.000

Fonte: O Autor, 2025.

Com base nos resultados obtidos através das "*strings*" supracitadas, percebe-se que não há uma quantia significativamente relevante de resultados, impactando na perspectiva que, este tema de reformulação ou refatoração, é amplamente abordado no âmbito do desenvolvimento *web*, tendo a consciência de sua devida importância nos dias atuais, entretanto, não se é observado essas mudanças com tanta frequência, com os mais variados motivos para tal, tais como: falta de tempo para realização de reformulação, falta de profissionais dispostos a realizar este serviço e também, dificuldade proposta para recomeçar ou refatorar um sistema.

Os resultados obtidos, foram de relevância para o estudo para a construção da percepção de que a acessibilidade *web* e a reformulação destas aplicações andam em conjunto, visto que, sempre quando há ocorrências de melhoria de acessibilidade, são decorridos de um processo de mudanças estéticas, funcionais ou de estrutura para suportar novos meios de acesso destes sistemas, como será demonstrado nas exemplificações infracitadas.

3.2. Artigo governamental do estado da Bahia

Após ser realizado uma remodelagem no *layout* e dos elementos de interface do *website* do Perfil de Administração Pública do estado da Bahia, foi relatado no *website* oficial do governo estadual, um aumento no número de acessos e de procura por este serviço e também houve aumento no número de cadastros nestas plataformas. A reformulação, teve como objetivo simplificar e tornar a aplicação menos burocrática, facilitando a navegação e tornando-a mais acessível para todos os públicos.

> Desde o dia 25 de agosto do ano passado, 7.792 cidadãos ingressaram no site do Perfil da Administração Pública em busca de informações atualizadas sobre a organização e o funcionamento dos diversos órgãos e entidades que integram a estrutura do Poder Executivo Estadual. No último mês, 469 usuários acessaram 1637 páginas do endereço eletrônico. Os números integram um balanço do impacto de um ano do lançamento de uma nova interface para o site, que está ajudando a promover transparência e ampliar o acesso da população a dados institucionais da administração pública estadual (SAEB, 2023).

3.3. Trabalho acadêmico sobre a relação "governo-cidadão" na reestruturação de websites

Ao observar o projeto acadêmico publicado e organizar os dados levantados, pode-se notar que, de acordo com os resultados originais obtidos pela pesquisa publicada pela USP (Universidade de São Paulo), cerca de 53% da população entrevistada que acessa os *websites* de administração pública, não chegam a concluir o ensino médio, enquanto 80% do público entrevistado, possui entre 26 e 40 anos, o que leva ao fato de que os *websites* precisam ser mais acessíveis, práticos e com *layouts* intuitivos para o público geral, ou seja, menos redirecionamentos de páginas, textos mostrando clareza e objetividade e maior utilização palavras simplificadas e diretas.

3.4. Trabalho de conclusão de curso com o tema da (in)acessibilidade dos sites governamentais

Analisando o trabalho de conclusão de curso mencionado, podemos notar que, segundo a W3C-WAI[7] (Iniciação da Acessibilidade Web, do inglês, *Web Accessibility Initiative*) o que define um *website* com um mal funcionamento, são duas ocasiões: a falta de estrutura das páginas de *websites*, ocasionando desorientação e confusão ao navegar pela aplicação, juntamente com o fato da utilização abusiva de elementos e informações gráficas, sendo imagens, mapas, tabelas e *scripts* desnecessários, desenvolvendo precariedade de alternativas adequadas para a navegação do usuário. O conjunto destas duas ocasiões, resulta em uma falha de planejamento, proporcionando uma má experiência de usuário.

> A *web* é um recurso cada vez mais importante em muitos aspectos da vida: educação, emprego, governo, comércio, assistência médica, recreação e muito mais. É essencial que a *web* seja acessível para fornecer acesso e oportunidades iguais a pessoas com habilidades diferentes. O acesso às tecnologias de informação e comunicação, incluindo a *web*, é definido como um direito humano básico (W3C-WAI, 2024).

De acordo com a W3C-WAI (2024), as diretrizes de desenvolvimento devem possuir para melhor acessibilidade, duas características: assegurar transformações harmoniosas e tornar o conteúdo acessível e navegável. Para cumprir com essas características, deve-se manter a aplicação acessível apesar de quaisquer limitações presentes, nas quais, por exemplo, pode-se citar as deficiências sensoriais, físicas ou cognitivas de um possível usuário. Como medidas de acessibilidade, pode-se citar como exemplo: separação da estrutura da apresentação, podendo diferenciar o conteúdo abordado no software através de comunicação visual e estética, também, deve-se incluir textos ou equivalentes textuais para interpretação, que sejam interpretados por todos os dispositivos de navegação.

3.5. Trabalho de Conclusão de Curso com o título: Reformulação de um Sistema de Gerenciamento de Cartões de Crédito para atender os princípios SOLID

O trabalho de Conclusão de Curso supracitado se correlaciona com o tema deste artigo com base na proposta de Reformulação, conceito que é debatido em ambas as pesquisas, tendo como foco principal, a reestruturação de sistemas legados para aprimorar sua

[7] https://www.w3.org/WAI/

eficiência, manutenção e adoção de boas práticas de desenvolvimento de *software*. Também, percebe-se que o conjunto de ferramentas e tecnologias utilizadas na reformulação deste trabalho apresenta forte correlação com as utilizadas neste artigo e projeto, evidenciando semelhanças nas abordagens técnicas e teóricas em ambos os estudos. O trabalho citado, apresenta como princípio a ser seguido na implementação, a metodologia *SOLID*[8], a qual tem o objetivo de modularizar e estruturar de maneira mais flexível e versátil sistemas com programação orientada a objetos, trazendo clareza nas suas funcionalidades implementadas. Com essa observação, é possível afirmar que, enquanto o trabalho de Domingues (2021) concentrou-se na reformulação de um sistema específico, voltado ao gerenciamento de cartões de crédito, este projeto adota a mesma linha de raciocínio, mas direciona a reformulação para outros padrões de desenvolvimento *web*, para uma área que é acessada por uma gama maior de usuários e que necessita de uma reformulação não apenas técnica, mas também, visual e aplicando práticas de *design patterns* para obtenção de bons resultados.

3.6. Considerações Finais do Estado da Arte

Juntando as observações acima nesta Seção, concluímos que o debate sobre acessibilidade e reformulação são ambos correlatos no âmbito do desenvolvimento *web*, um necessitando do outro para ser realizado, desta forma, percebemos que toda reformulação, tem como foco melhorar algum produto, ou seja, tornar mais acessível para determinado uso ou função. Em decorrência da perspectiva dos usuários, uma melhoria nos sistemas e *websites* que já existem, acarretam em mais usuários se interessando em utilizar os mesmos, expandindo a sua eficiência e credibilidade, pontos importantes e valiosos para o setor público.

4. Metodologia

Para o desenvolvimento da aplicação, foram utilizadas as tecnologias concebidas com os termos de *back-end* e *front-end*, utilizando as ferramentas de *back-end* NodeJS com utilização da biblioteca TypeORM, juntamente com ReactJS para formulação do *front-end*. A utilização destas ferramentas em conjunto com o *TypeScript* possibilita a criação de projetos e sistemas altamente performáticos e harmoniosos. Segundo estudos da Stripe (2018), cerca de 42% do tempo gasto com o desenvolvimento de *software* é destinado apenas para manutenção de sistemas, cerca de 3.8 horas são gastos apenas ajustando e corrigindo um “código ruim”, resultando em uma perda de $85 bilhões de dólares anualmente de oportunidade de renda. Com base no texto apresentado, percebe-se a urgência de reformulação pertinente aos *websites* considerados defasados, pois os mesmos, geram enormes custos e tempo dos desenvolvedores para ajustá-los, o que seria drasticamente reduzido se fossem reformulados ou refatorados.

4.1. Tecnologias Utilizadas

Para atender à necessidade de modernização e otimização do processo de reformulação de *websites*, diversas tecnologias de desenvolvimento *web* foram utilizadas, sendo o Node.js, o TypeScript e o React. Estas ferramentas em conjunto se destacam por sua

[8] O acrônimo SOLID representa cinco princípios que facilitam o processo de desenvolvimento

performance e capacidade de criação de interfaces dinâmicas, facilitando o desenvolvimento de sistemas. Com esse conjunto de tecnologias e ferramentas, foi possível desenvolver uma aplicação fluida, performática e de alta capacidade de manutenção, juntamente com o fato de proporcionar não apenas ganhos em desempenho e segurança digital, mas como também uma base sólida para futuras adições de funcionalidades e integrações.

4.1.1. Utilização do TypeScript

A linguagem de programação TypeScript foi utilizada em função da necessidade de um código mais robusto e previsível durante a refatoração dos componentes da aplicação. O TypeScript é uma linguagem de programação criada com base no JavaScript, sendo uma extensão da mesma, tendo tipagem opcional, orientação a objetos e recursos avançados de desenvolvimento seguro. A inserção inicial do TypeScript em projetos novos é uma excelente opção para um bom desenvolvimento escalável, quando adotamos o uso do TypeScript desde o início do projeto, estamos criando um software que nos acompanha em cada linha escrita, nos auxiliando a cada passo no desenvolvimento. Os erros que antes só apareciam em produção, durante testes manuais ou na pior das hipóteses, nas mãos dos usuários, agora são capturados diretamente no seu editor de texto, em tempo real, facilitando a obtenção de *bugs* durante o processo de desenvolvimento de um sistema, resultando em um produto final mais polido e otimizado.

4.1.2. Formulação do back-end

Para o desenvolvimento do conceito do *Back-end*, foi utilizado o NodeJS, uma plataforma baseada em JavaScript que permite a execução de código do lado do servidor. A escolha do Node.js se deu pela sua capacidade de lidar com múltiplas requisições simultâneas de forma eficiente, graças ao seu modelo assíncrono e orientado a eventos, possibilitando a performatividade de um código que respeita os parâmetros do *clean-code*. Trabalhado em conjunto ao TypeORM, uma biblioteca de gerenciamento ORM (*Object-Relational Mapping*), foi possível criar uma camada de versatilidade e segurança entre a aplicação e o banco de dados, facilitando operações de leitura, gravação e atualização de dados de maneira organizada e segura.

4.1.3. Formulação do front-end

No *front-end*, a escolha se deu sobre o React, uma extensão do JavaScript voltado para a criação de interfaces de usuário reativas nas ações do usuário, juntamente com a utilização de componentes reutilizáveis no projeto. O React foi utilizado em combinação com o Vite, uma ferramenta de *build* moderna e sofisticada para otimização do processo de carregamento e do desenvolvimento do projeto. Como escolha de pacote de estilização de interfaces para incrementar o desempenho do projeto, foi incrementado o TailwindCSS[9], permitindo a criação de interfaces modernas, responsivas, otimizadas e consistentes, sem a necessidade de escrita de grandes blocos de CSS de forma manual. Essa abordagem, além de proporcionar maior produtividade e versatilidade no desenvolvimento, garante um visual padronizado e modulado em toda a aplicação. A utilização do TailwindCSS, também

[9] https://tailwindcss.com/

incrementa nos parâmetros de boas práticas de estruturação de pastas, uma vez que, permite o usuário não ter a necessidade de criar um arquivo de extensão ".css" para cada componente utilizado no projeto.

4.1.4. Integração do back-end com o front-end

A integração entre os projetos *front-end* e *back-end* foi realizada por meio de requisições HTTP (*Hypertext Transfer Protocol*) utilizando o Axios, garantindo uma comunicação segura e estável da aplicação. Foram implementadas boas práticas de autenticação e controle de sessão, com o uso de JWT (*JSON Web Token*), mencionado mais detalhadamente na Seção 5.3.6, utilizado para controle de ações dentro do sistema quando necessária autenticação por parte do usuário, assegurando que apenas usuários devidamente autenticados possam acessar recursos protegidos pelo sistema.

4.1.5. Criação do Banco de Dados da aplicação

Para escolha de versionamento, gerenciamento e armazenamento de dados do projeto, foi escolhido o PostgreSQL como sistema de gerenciamento de banco de dados por ser uma tecnologia confiável, segura e amplamente utilizada no universo do desenvolvimento *web*. Por se tratar de uma ferramenta *open source*, oferece estabilidade e bom desempenho no armazenamento e manipulação de dados, além de uma comunidade ativa para formulação de melhorias e criação de conteúdo em *blogs* e *websites*. Também, em conjunto com os demais fatores, possui fácil integração com outras tecnologias utilizadas no projeto, tal como o TypeORM, facilitando a criação e manutenção da estrutura do banco de dados de forma eficiente.

4.1.6. Aplicação em Produção

Como escolha para efetuar a implementação do sistema e *website*, foi escolhido os ambientes de implantação de softwares *Railway*[10] e *Vercel*[11], respectivamente, para o *back-end* e o *front-end*. A utilização da plataforma *Railway* para o *back-end* possibilitou a implantação contínua da API (*Application Programming Interface*), com integração direta ao repositório do projeto no *GitHub*[12], permitindo que novas versões fossem automaticamente disponibilizadas após cada atualização no código-fonte, igualmente na *Vercel*, ajustando e atualizando apenas as variáveis de ambiente conforme o necessário para complementação do *front-end*, atribuindo o repositório e obtendo atualizações em tempo real assim quando o código-fonte é atualizado.

4.2. Arquitetura do Sistema

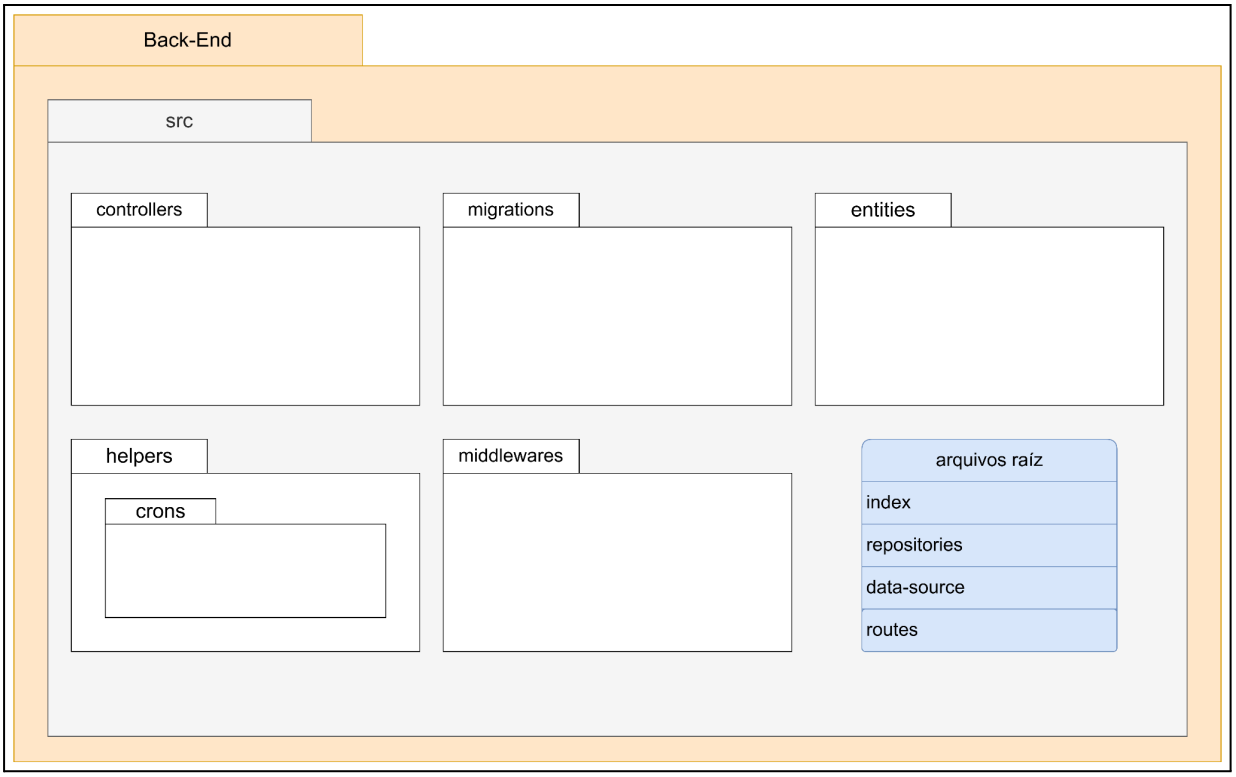
A metodologia aplicada para o desenvolvimento do sistema, seguiu parâmetros de arquitetura "*clean-code*", método de desenvolvimento de sistemas amplamente utilizado que busca garantir legibilidade, manutenibilidade e eficiência do refinamento de um código. Tal metodologia de desenvolvimento, tem o intuito de modularizar as funções presentes no código, trazendo funcionalidades bem definidas e estáveis para estruturação e identificação

[10] https://railway.com/
[11] https://vercel.com/
[12] https://github.com/

do sistema, carregando consigo, melhorias para visibilidade de falhas, implementação de boas práticas e novas funcionalidades de maneira mais ágil. Essa arquitetura também contempla o favorecimento da escalabilidade da aplicação, permitindo que o sistema possa evoluir e incluir novas funcionalidades de maneira sistemática e precisa.

Além disso, foram adotadas boas práticas de organização de pastas e padronização de nomenclaturas, visando um fluxo de desenvolvimento mais claro e limpo no ambiente de desenvolvimento. O fluxo de pastas foi estruturado de forma hierárquica, agrupando arquivos da aplicação conforme suas funções e utilidades dentro do escopo do projeto proposto, assim, definindo uma melhor separação de responsabilidades e atribuições, facilitando a manutenção do código.

Figura 1 - Diagrama de Pastas Back-End



Fonte: O Autor, 2025.

Conforme a imagem apresentada na Figura 1, temos em vista a estruturação do projeto de forma dinâmica em sua utilização de pastas, contendo todos os conteúdos pertinentes de fácil acesso, possibilitando uma clara compreensão do escopo do sistema, sendo dividido cada pasta para cada funcionalidade presente na formulação da API da aplicação projetada para este artigo.

4.3. Análise do Objeto de Estudo

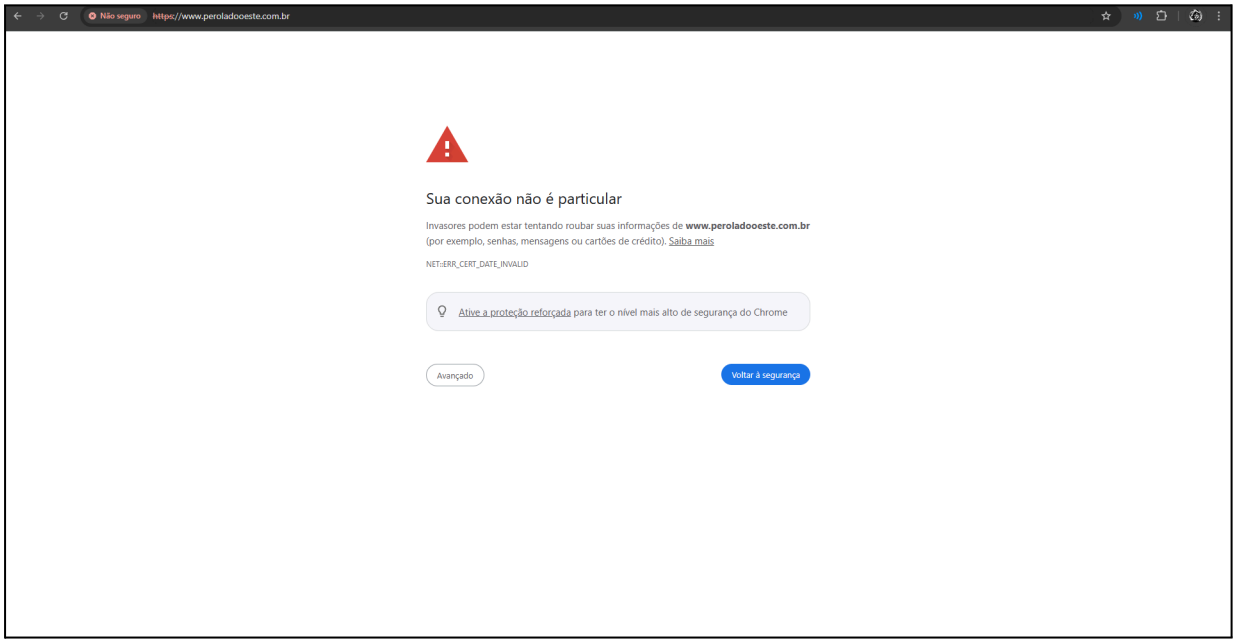
O objeto de estudo estabelecido para uma reformulação, foi definido como o conjunto de *website* da Pérola do Oeste[13], juntamente com o sistema do Portal do Aluno, cujo os

[13] https://www.peroladooeste.com.br/

mesmos, estão sob o domínio da empresa Transportes Coletivos Pérola do Oeste[14], definida como uma instituição de Administração Pública Indireta, na localidade da cidade de Guarapuava, estado do Paraná. A análise do *website* institucional disponível no atual domínio www.peroladooeste.com.br revelou diversos problemas de estruturação, dificuldades técnicas e obstáculos estéticos que comprometem a sua utilização.

Ao acessar o endereço do domínio supracitado, notamos na Figura 2, uma exibição de alerta, cuja mensagem diz “Sua conexão não é particular” no título, indicando falhas no certificado de segurança SSL (Camada de Soquetes Seguros, do inglês, *Secure Sockets Layer*), um protocolo de segurança que permite transação de dados via HTTPS (Protocolo de Transferência de Hipertexto Seguro, do inglês, *Hypertext Transfer Protocol Secure*) entre um *website* e um usuário de maneira criptografada, o certificado, estando expirado, impede o acesso seguro via o protocolo HTTPS, representando um risco à privacidade dos usuários.

Figura 2 - Alerta de conexão não particular do Objeto de Estudo



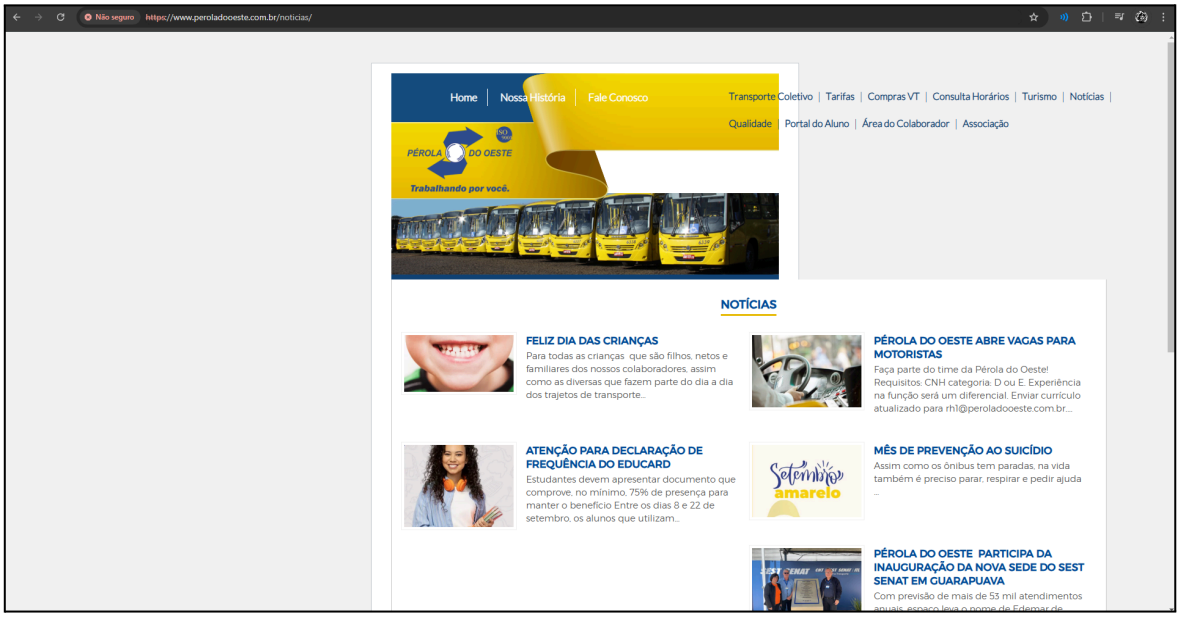
Fonte: O Autor, 2025.

Continuamente no *website*, se dá o encontro de diversos pontos que causam problemas de usabilidade para o usuário, tendo como exemplo a Figura 3 em demonstração, podemos notar que a identidade visual se mostra desatualizada e pouco atrativa. O *site* utiliza cores fortes em contraste direto e tipografia pouco harmoniosa, com grandes blocos de texto pouco convenientes para o usuário, resultando em uma hierarquia visual confusa. No exemplo da Figura 3, percebe-se que o *layout* da página está completamente desconfigurado e fora de ordenação, estando desta forma, o usuário perde a confiança nas informações a qual procura, bem como, a credibilidade pelos serviços prestados pela instituição em questão. No meio do processo, também, acaba perdendo as informações que procura, pois tende a desistir

[14] 77.147.387/0001-38

de encontrar os dados que necessita em meio ao ambiente caótico de uma página não adaptada e pouco coerente para o acesso ao público geral.

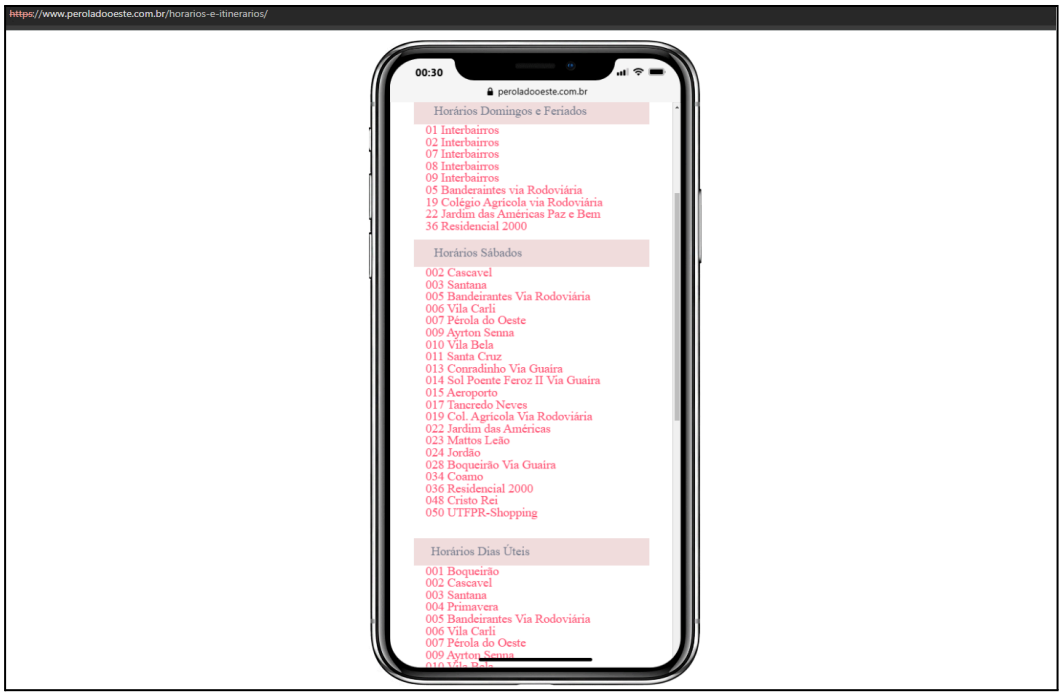
Figura 3 - Layout desformatado no Objeto de Estudo, Website da Pérola do Oeste



Fonte: O Autor, 2025.

Outro ponto relevante a ser observado, está relacionado à interface visual do sistema. O *layout* apresenta características de desenvolvimento defasado, sem aplicação consistente de *design* responsivo para atender à demanda de acessibilidade para dispositivos móveis.

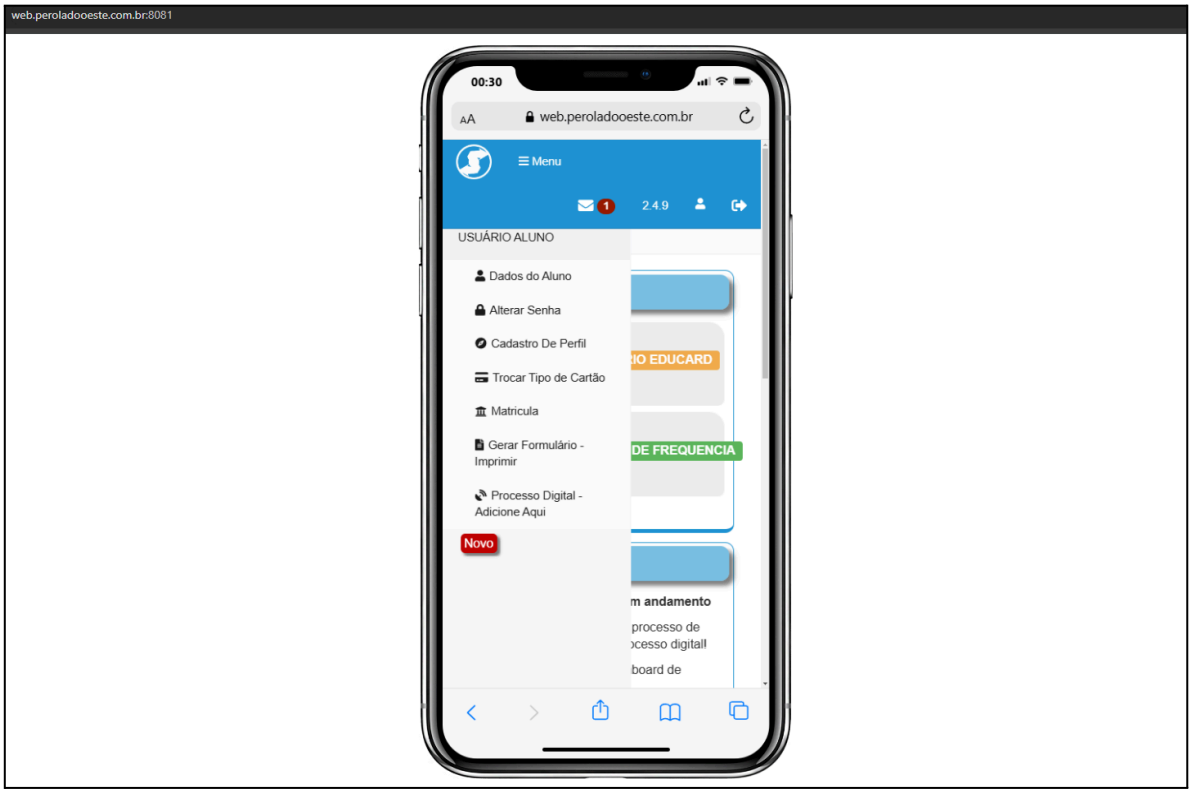
Figura 4 - Website do Objeto de Estudo em Visibilidade Mobile



Fonte: O Autor, 2025.

Em perspectiva no *website*, na visão dos dispositivos *mobile*, percebemos que há uma precariedade no desempenho de visibilidade do conteúdo para os usuários, visto que, grande parcela da população que necessita do amparo de serviços públicos irá utilizar os celulares como meio de acesso a estes *websites*. No acesso por intermédio do uso de dispositivos móveis, em certas páginas informativas, é possível notar diversos exemplos de desalinhamentos, textos cortados, *layout* inconsistente e imagens desproporcionais, evidenciando a ausência de adaptação para diferentes resoluções de tela.

Figura 5 - Layout desformatado presente no Objeto de Estudo, Informações iniciais do Sistema Portal do Aluno



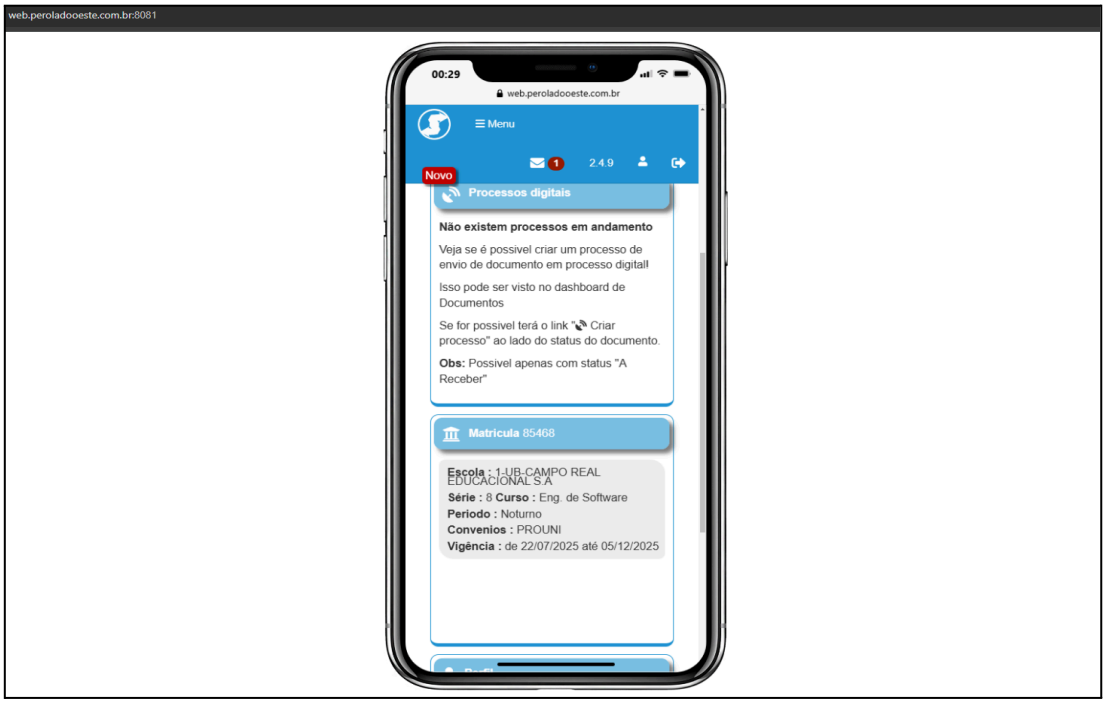
Fonte: O Autor, 2025.

Em continuidade no âmbito do sistema, seguindo no exemplo da Figura 6, pode-se observar a falta de clareza na disposição de elementos gráficos presentes para o aluno, telas com *layouts* repetidos, trazendo consigo confusão ao usuário identificar qual tela está sendo usada no momento, páginas adicionais desnecessárias nas quais poderiam ser compactadas em apenas uma tela, elementos sem objetividade, botões com funções não identificáveis, barra lateral confusa, etc. Estes problemas tornam a navegação difícil e pouco intuitiva, especialmente considerando que a maioria dos acessos à internet atualmente é realizada por meio de *smartphones*.

Além dos fatos citados, a ausência de uma hierarquia visual bem definida e estruturada, dificulta o entendimento do usuário, a falta de padronização nas cores e tipografias também contribui para uma experiência fragmentada e visualmente cansativa para o leitor da página. Outro ponto crítico é a ausência de responsividade adequada, o que

prejudica a sua usabilidade em dispositivos móveis, cerne dos dispositivos mais utilizados por usuários. Portanto, faz-se necessária uma reestruturação do *design* de interface, priorizando a clareza, a consistência e a facilidade de navegação.

Figura 6 - Tela de Dashboard inicial do Objeto de Estudo, Sistema do Portal do Aluno



Fonte: O Autor, 2025.

4.4. Obtenção de Feedbacks

Como forma de obtenção crucial de resultados referentes ao protótipo desenvolvido, adotou-se uma metodologia direcionada ao aperfeiçoamento do produto e à análise de *feedbacks* com a utilização dos Formulários Google[15], ferramenta capaz de juntar dados e com os mesmos, obtenção de resultados em forma de disposição de gráficos e porcentagem, sendo de fundamental importância para a fundamentação dos resultados, abordados posteriormente na Seção 5.4.

5. Resultados e Discussões

Este tópico tem como objetivo apresentar e dissertar os resultados obtidos durante o desenvolvimento e implementação da aplicação.

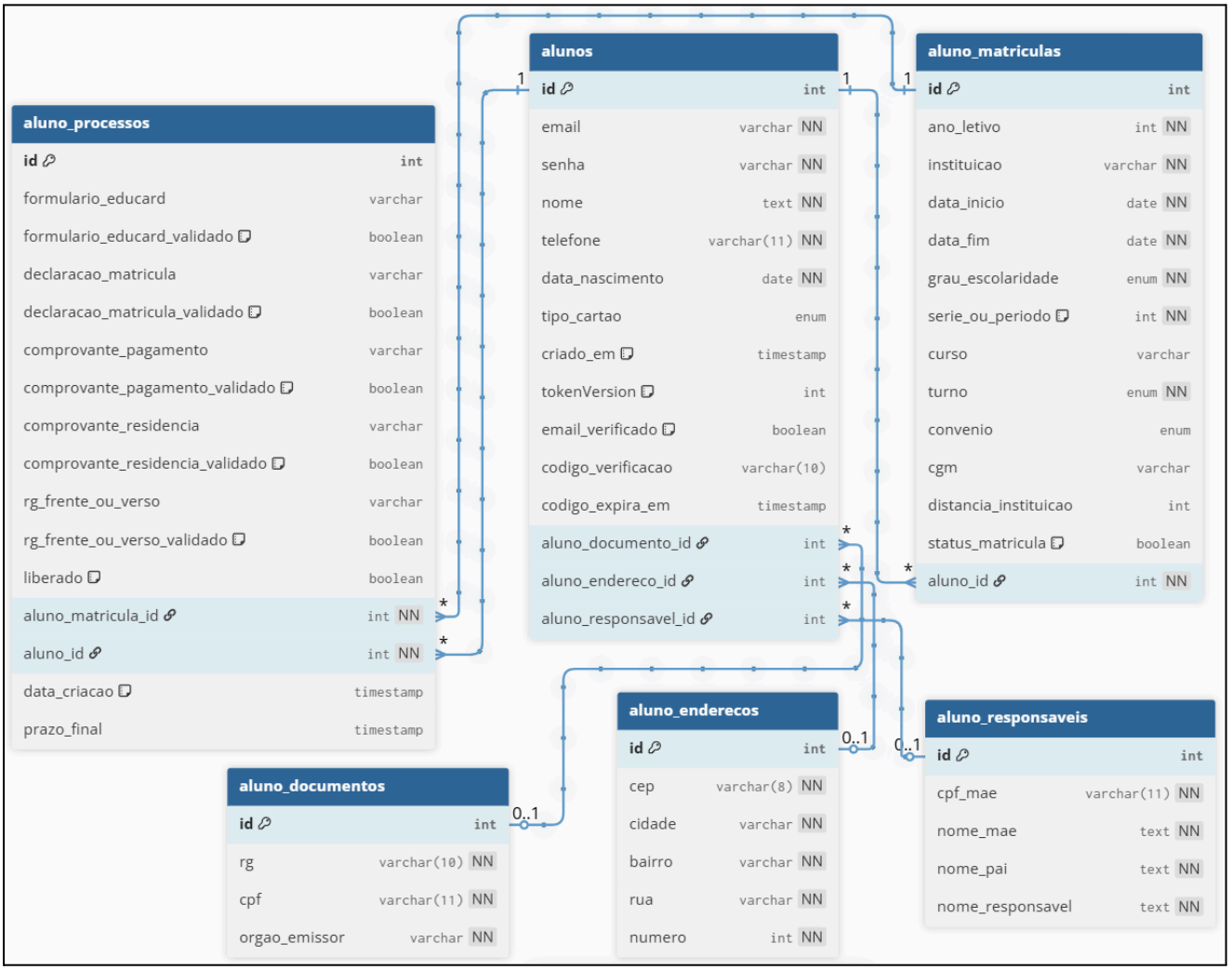
5.1. Banco de Dados

Um dos recursos do *back-end* mantido tanto no código legado quanto no código reformulado foi a utilização do banco de dados. Conforme mencionado na Seção 4.1, o PostgreSQL foi escolhido como SGBD do projeto, utilizado em conjunto ao typeORM para implementação de criação de entidades, versionamentos, controladores e validações de dados. Essa integração permite que as regras e restrições de dados sejam descritas diretamente nas

[15] https://docs.google.com/

funcionalidades controladoras do *back-end*, proporcionando maior visibilidade, padronização e facilidade de manutenção do sistema, permitindo também, melhor controle de alterações e gerenciamento de futuras mudanças.

Figura 7 - Diagrama Relacional de Banco de Dados



Fonte: O Autor, 2025.

O banco de dados foi modelado de forma a suportar os dados pertinentes ao sistema do Portal do Aluno, como demonstrado na Figura 7, o modelo Relacional representado, apresenta uma visão mais precisa do que foi implementado no sistema. Essa visão de diagrama contempla tabelas, chaves, tipos de dados, atributos e relacionamentos que serão utilizados no Banco de Dados do projeto.

5.2. Produção da Interface Visual

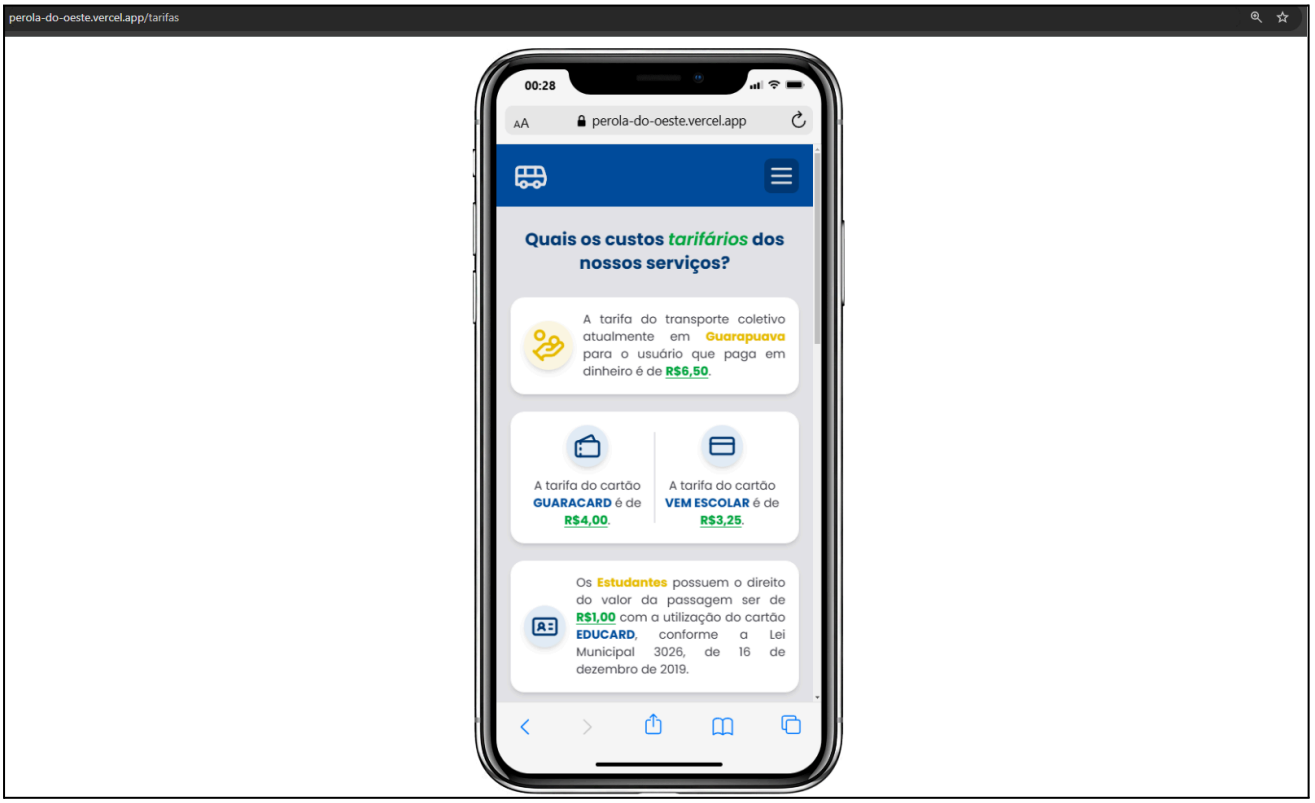
Com o objetivo de desenvolver uma interface mais coerente com o perfil do público majoritário, adotou-se o conceito da metodologia *mobile-first*. Essa abordagem prioriza a construção da interface visual voltado primeiramente para os dispositivos móveis,

favorecendo a criação de aplicações mais adequadas para o público-alvo da maioria dos sistemas.

> A pesquisa mostrou que 142 milhões fizeram uso diário ou quase diário da rede no país. A maior parte dos usuários de Internet brasileiros (62%) acessa a rede exclusivamente pelo celular, realidade de mais de 92 milhões de indivíduos. A conclusão é da TIC Domicílios 2022 (Comitê Gestor da Internet no Brasil, 2023).

Além dos fatos citados, o método *mobile-first* facilita o desenvolvimento responsivo, pois, ao inicializar o projeto para telas menores, a adaptação para dispositivos de telas com maiores dimensões torna-se mais consistente e eficiente para realizar ajustes quando necessário.

Figura 8 - Reformulação da Página de Tarifas do Website



Fonte: O Autor, 2025.

Seguindo exemplo demonstrado na Figura 8, percebe-se o resultado de uma Reformulação direcionada para usuários utilizando um celular, contendo os blocos de informações pertinentes contemplados na formatação inicial da tela, sem necessidade do usuário utilizar o *scroll* para exibir as demais informações importantes presentes na página. Também, pode-se notar a presença da utilização de variação de tons de cores em palavras-chave no texto, sendo utilizados como indicativos para os usuários poderem achar os dados pertinentes a qual procuram de maneira mais fluída e rápida.

Os textos utilizados permaneceram os mesmos, sem atualização de dados ou criação de informações fictícias. Entretanto, a disposição dos elementos foi reajustada para

proporcionar melhor visibilidade na interface, como é possível observar nas Figuras 9 e 10. Houve uma mudança significativa apenas na disposição das informações, a nova interface mescla diversas informações que, antes, estavam dispersas em várias páginas do *site* atual, agora centralizadas em uma única página.

Figura 9 - Interface Reformulada, página “Sobre Nós” do Website

Fonte: O Autor, 2025.

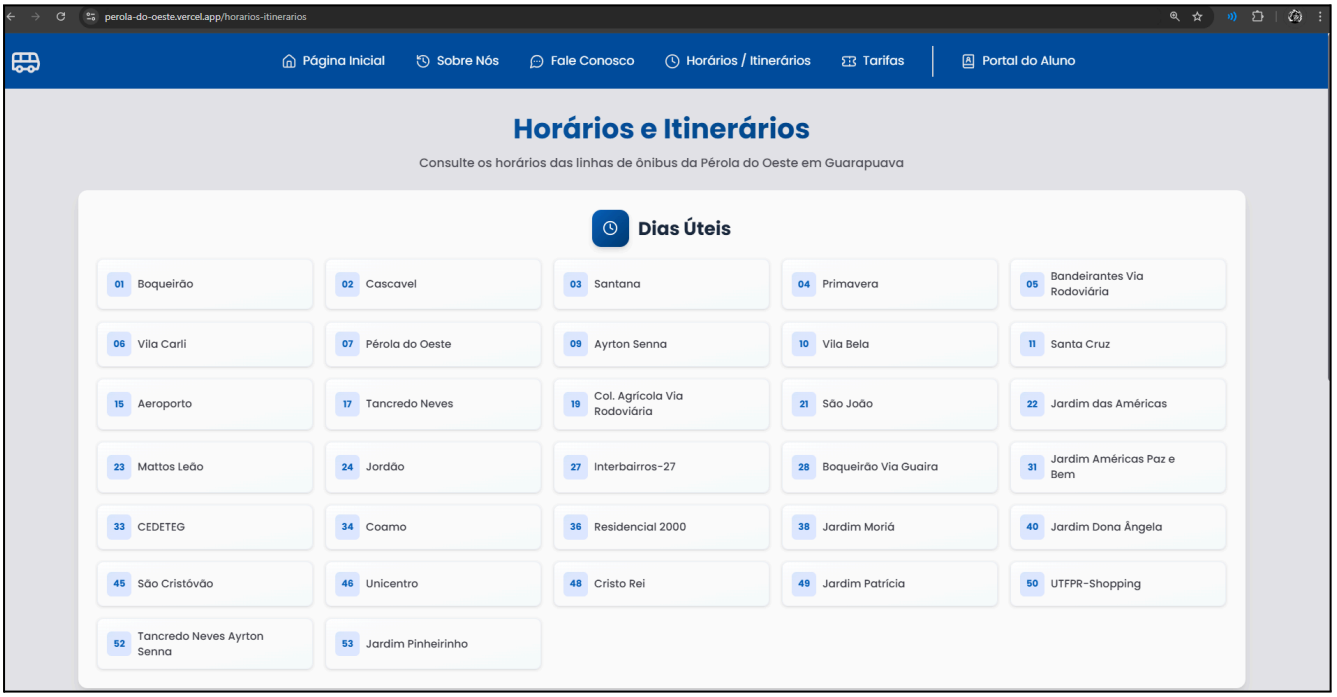
Nas Figuras 9 e 10 em demonstração, podemos perceber as informações que antes, diversificadas em páginas distintas, agora unidas em apenas uma seção de página, fornecendo clareza ao leitor e centralização de informações, reduzindo o número de páginas totais.

Figura 10 - Interface Reformulada, página “Sobre Nós” do Website

Fonte: O Autor, 2025.

Ao analisar as problemáticas encontradas no Objeto de Estudo, foi possível desenvolver interfaces mais dinâmicas e consistentes, ainda com influência do modo de funcionamento do produto original, mas realçando a UI, tornando polida e sofisticando a usabilidade, conforme feita a comparação da Figura 11, a página reformulada da grade de Horários e Itinerários à Figura 4, anteriormente apresentada. Configurando assim, uma organização mais consistente e melhorada, não interferindo no processo de fluxo de usabilidade do produto do Objeto de Estudo, mantendo a mesma dinâmica de acesso, apenas desenvolvendo de uma forma mais capaz de satisfazer o usuário ao acessar a aplicação.

Figura 11 - Tela Reformulada de grade de Horários e Itinerários



Fonte: O Autor, 2025

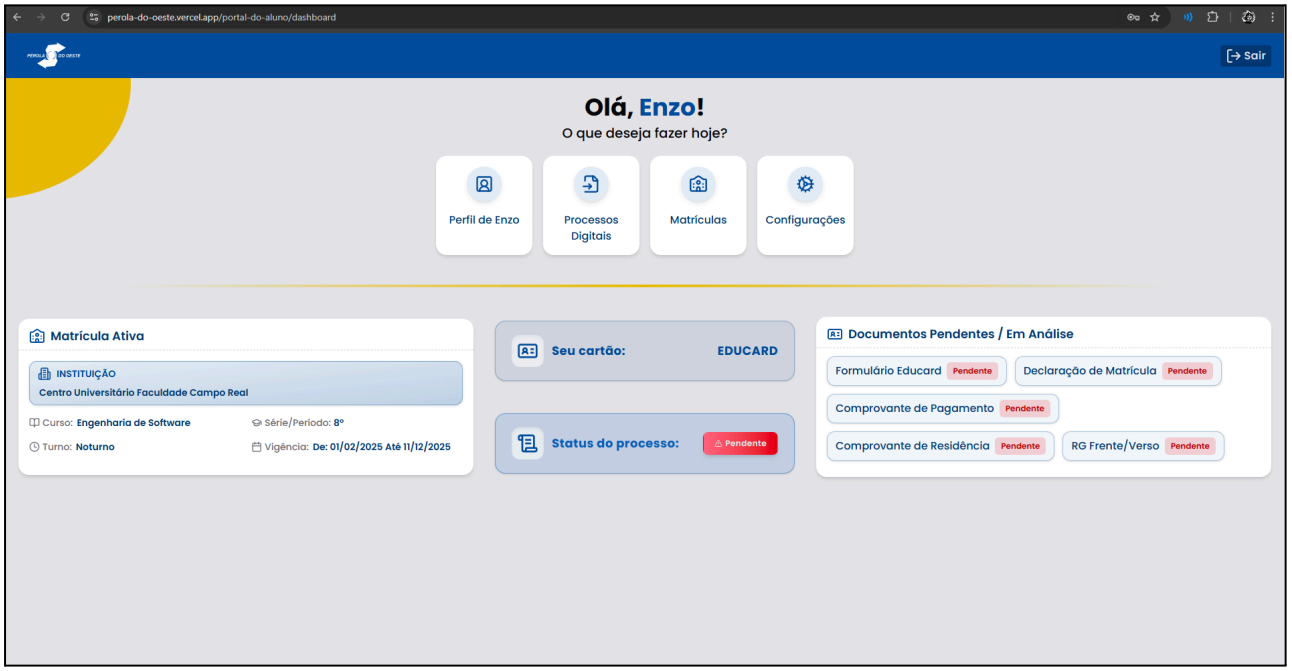
Partindo para o Sistema do Portal do Aluno reformulado, obtemos uma mudança mais drástica na interface visual, não seguindo padrões previamente estabelecidos no sistema atual, fornecendo uma reformulação mais completa. Observando o atual estado, percebe-se uma incongruência de um *layout* bem formulado e apresentável, partindo dos *feedbacks* coletados posteriormente na Seção 5.4, há uma insatisfação presente na localização do conteúdo necessário para a navegação e utilização deste serviço.

Na versão reformulada, os pontos de interesse iniciais a qual o aluno pode ter, foram distribuídas de maneira regular e componentizada em blocos de dados distintos, fazendo com que cada bloco tenha seu propósito informativo concluído, organizando uma *Dashboard* inicial mais limpa e coesa, servindo seu propósito de ser uma versão resumida das funcionalidades presentes no sistema, juntamente com pontos de navegação para as mesmas.

No atual estado do Objeto de Estudo, um fator determinante que dificulta a navegação para o usuário, é o número excessivo de páginas desnecessárias, resultando em uma

experiência fragmentada e confusa durante a utilização do sistema. Esse acúmulo de pontos de navegação faz com que o usuário precise realizar diversos cliques e transições entre telas para executar tarefas simples, comprometendo a eficiência do sistema. Para solução de tal empecilho, foi utilizada a abordagem do uso de modais, modelos de interação sobreposta à tela, sem a necessidade de troca de páginas no processo. O uso de modais, além de contribuir para a melhoria da performance do sistema, uma vez que evita o carregamento completo de páginas em transição, ele também concentra as interações em um único fluxo, fazendo com que o usuário necessite de menos cliques e interações para a utilização do sistema, respeitando uma boa usabilidade de interface de usuário.

Figura 12 - Tela de Dashboard Inicial Reformulada do Sistema do Portal do Aluno



Fonte: O Autor, 2025.

De acordo com NIELSEN (1994), “minimize a carga de memória do usuário tornando os elementos, ações e opções visíveis". O usuário não deve ter que se lembrar de informações de uma parte da interface para outra.”. Além disso, tendo uma indevida redundância de páginas pode gerar lentidão no carregamento, aumentar o tempo de resposta e dificultar a localização de informações. Juntamente com a utilização dos modais, um outro fator que contribui para servir como propósito de guiar o usuário, se deve à utilização de ícones, determinantes para o reconhecimento de cada funcionalidade e informação disposta na tela. Como padronização de biblioteca de ícones, foi utilizado o pacote de extensão do Lucide[16]. Ao analisar as páginas de maior relevância e frequência de uso dentro do sistema, identificou-se a tela de criação de processos digitais e a de criação de matrículas, pois as mesmas, são o cerne da existência da aplicação.

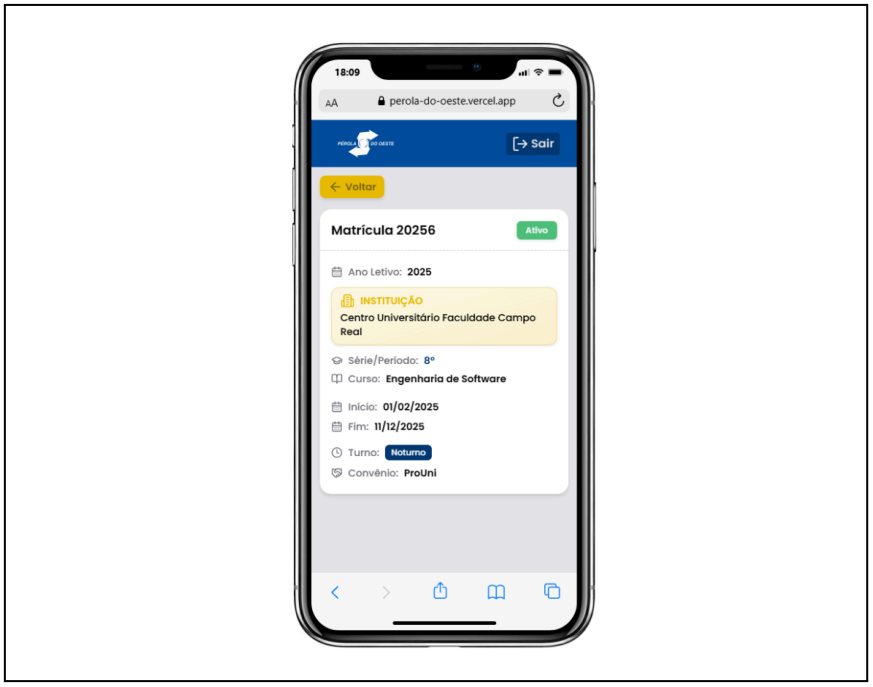
[16] https://lucide.dev/

Figura 13 - Componente de Modal, utilizado no Sistema Portal do Aluno Reformulado

Fonte: O Autor, 2025.

Uma problemática encontrada ao disponibilizar informações de processos e matrículas em formato de tabelas, são os desafios da aplicação responsiva para os mesmos, logo, foi adotada a metodologia do uso de *Cards* para exibição em dispositivos móveis, intercalando entre tamanhos de tela para exibição de *Cards* e Tabelas.

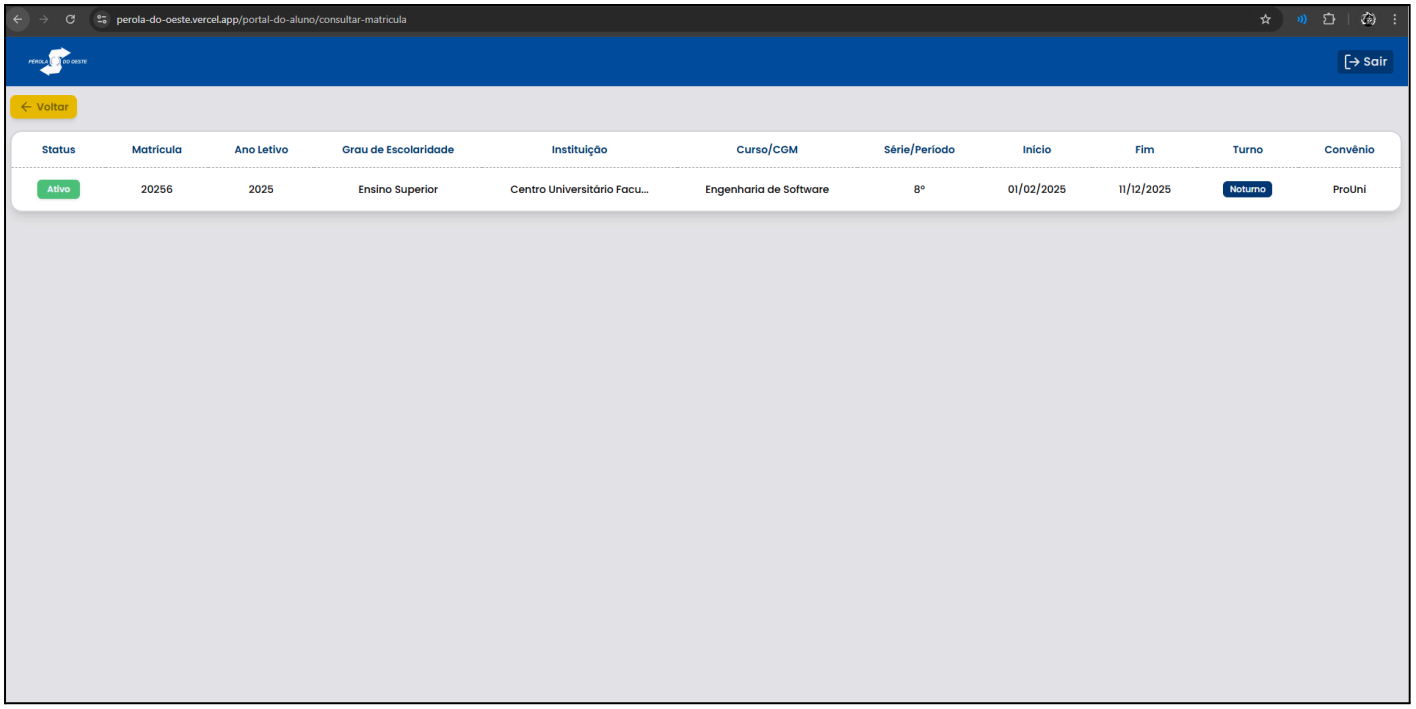
Figura 14 - Exemplo de disposição de elementos em formato de Card nas Matrículas



Fonte: O Autor, 2025.

Ao ato de transição de formato e tamanho de telas, a disposição dos elementos mudam, se adaptando para uma melhor visibilidade de informações para cada tipo de dispositivo utilizado para acessar a aplicação. Cada tipo de disposição, é um componente distinto, carregado de acordo com a resolução de tela detectada pelo navegador. Este tipo de método utilizado, facilita a manutenção de componentes, pois ao detectar cada inconformidade encontrada em diferentes telas ou disposições, o desenvolvedor consegue rapidamente identificar o problema, pois cada recurso está devidamente separado e componentizado de forma sistemática no produto desenvolvido.

Figura 15 - Exemplo de disposição de elementos em formato de Tabela nas Matrículas



Fonte: O Autor, 2025.

5.3. Formulação do back-end

Como forma de construção do *back-end*, conforme mencionado na Seção 4.1, as ferramentas utilizadas foram o NodeJS, em conjunto com o TypeORM, ferramenta de extensão para auxílio da construção do banco de dados, abordado na Seção 5.1, sendo estruturados de maneira objetiva, como demonstrado na Figura 1.

5.3.1. Estrutura de Camadas

A arquitetura do *back-end*, visivelmente é semelhante com o Modelo MVC (*Model-View-Controller*), tratando da utilização de controladores, *migrations*, entidades e rotas, compondo uma arquitetura em camadas na aplicação, padrão semelhante ao Framework PHP, Laravel[17]. O objetivo do uso de um padrão semelhante ao MVC, deve-se à organização e a uma melhor separação de atividades exercidas por cada arquivo e pasta, facilitando a legibilidade e manutenção regular do código.

[17] https://laravel.com/

5.3.2. Entities

Primeiramente, se elabora a camada de entidades (pasta *Entities*) cuja sua finalidade é representar os modelos de dados da aplicação, refletindo diretamente como as tabelas do banco de dados. Essa camada utiliza os recursos do TypeORM, permitindo o mapeamento objeto-relacional (ORM), criando as entidades como tabelas, colunas, tipos de dados, validações e relacionamentos, assim, assegurando a consistência do banco de dados por meio de uma estruturação diretamente no código. A utilização do TypeORM, define as entidades de maneira simplificada e de fácil visibilidade para alterações e identificação dos elementos presentes para cada tabela e coluna criada no banco de dados, sendo de fundamental uso durante o processo de criação da aplicação.

Figura 16 - Exemplo de criação de Entidade Aluno com uso do TypeORM

TCC - Aluno.ts

```
1  @Entity("alunos")
2  export class Aluno {
3    @PrimaryGeneratedColumn()
4    id: number;
5
6    @Column({ unique: true, nullable: false })
7    @IsEmail(
8      { require_tld: true },
9      { message: "Insira um endereço de email válido." }
10   )
11   email: string;
12
13   @Column({ nullable: false })
14   @MinLength(6, { message: "A senha deve conter pelo menos 6 caracteres." })
15   senha: string;
```

Fonte: O Autor, 2025.

5.3.3. Migrations

As migrações (pasta *Migrations*) são utilizadas para o controle das alterações realizadas no banco de dados, permitindo a criação, atualização e reversão da estrutura formada pelas entidades anteriormente citadas. Essa prática favorece a rastreabilidade das alterações realizadas no banco, criando arquivos que possuem os comandos SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*), guardados como histórico efetuado para eventuais consultas posteriores.

5.3.4. Controllers

Os controladores (pasta *Controllers*), são onde a lógica responsável pelo tratamento das requisições HTTP são concebidas na aplicação, exportando suas funcionalidades para as rotas e acionando os serviços necessários, retornando as respostas apropriadas ao cliente em formato de uma API. Essa camada de requisições de controladores, atuam no sistema como

intermédio entre a interface de comunicação (*front-end*) e as regras de negócio do funcionamento do sistema, sendo o coração do sistema, pois todo o seu funcionamento, gira em torno dos controladores e em como os mesmos se comunicam com o resto da aplicação como um todo.

Figura 17 - Exemplo de função de logout, presente em um Controller

TCC - AuthController.ts

```
async logout(req: Request, res: Response) {
  const alunoId = req.alunoLogin.id;

  const aluno = await AlunoRepository.findOneBy({ id: alunoId });
  if (!aluno) {
    throw new BadRequestError("Aluno não encontrado.");
  }

  aluno.tokenVersion += 1;
  await AlunoRepository.save(aluno);

  res.status(200).json({ message: "Logout realizado com sucesso!" });
}
```

Fonte: O Autor, 2025.

Além de receber dados, os controladores realizam as validações iniciais, verificando se os campos obrigatórios estão presentes e se os dados seguem os padrões definidos anteriormente com o uso de objetos de validação, no caso desta aplicação, validações utilizando ferramentas auxiliares do TypeORM. Após a execução dos serviços controladores, os mesmos formatam a resposta de forma adequada para o cliente, definindo mensagens personalizadas para respostas e códigos HTTP, tais como o código 404, código para definição de erro de "*Not Found*" (erro "Não Encontrado", em português).

5.3.5. Routes

As rotas do sistema (pasta *Routes*), representam a camada responsável por definir os caminhos e os métodos de acesso às funcionalidades construídas na aplicação. Cada rota definida, é constituída por uma funcionalidade aplicada por um *Controller*, direcionando as funções para o *front-end*. Por meio dessas definições, a aplicação sabe qual função deve ser chamada em resposta a cada tipo de requisição, garantindo que os dados sejam circulados corretamente entre o cliente e o servidor.

Figura 18 - Exemplo de uso de Rotas no Sistema

TCC - routes.ts

```ts
routes.put("/api/aluno/alterar-senha", new AuthController().alterarSenha);

routes.post("/api/aluno/documento", new AlunoDocumentoController().create);
routes.get("/api/aluno/documento", new AlunoDocumentoController().list);

routes.post("/api/aluno/endereco", new AlunoEnderecoController().create);
routes.get("/api/aluno/endereco", new AlunoEnderecoController().list);

routes.post("/api/aluno/responsavel", new AlunoResponsavelController().create);
routes.get("/api/aluno/responsavel", new AlunoResponsavelController().list);

routes.post("/api/aluno/matricula", new AlunoMatriculaController().create);
routes.get("/api/aluno/matricula", new AlunoMatriculaController().list);
```

Fonte: O Autor, 2025.

5.3.6. JWT

Como medida para garantir o âmbito de segurança e meios de autenticação da aplicação, foi adotado o uso do JWT (*JSON Web Token*) como mecanismo de controle de sessão. O JWT funciona como uma "chave digital", essa chave é gerada pelo servidor no momento de uma requisição de login e retornada para o cliente.

Figura 19 - Utilização de JWT em requisição de Login

TCC - AuthController.ts

```ts
const verificarSenha = await bcrypt.compare(
      authData.senha,
      alunoAuth.senha
    );

    alunoAuth.tokenVersion += 1;
    await AlunoRepository.save(alunoAuth);

    const authToken = jwt.sign(
      { id: alunoAuth.id, tokenVersion: alunoAuth.tokenVersion },
      process.env.JWT_PASS ?? "",
      {
        expiresIn: "6h",
      }
    );

    const alunoAuthSemDados = {
      email: alunoAuth.email,
    };

    res.status(201).json({
      alunoAuth: alunoAuthSemDados,
      token: authToken,
    });
```

Fonte: O Autor, 2025.

O *token* fica armazenado nas "*headers*", mais precisamente, guardados em *cookies* no lado do cliente, com o objetivo de que sempre que o usuário realizar uma nova requisição em rotas as quais necessitam, o *token* é enviado juntamente, permitindo que o servidor identifique quem está acessando, sem precisar pedir o login novamente ou realizar mais verificações desnecessárias.

5.3.7. Crons do Sistema

Crons, tem como o objetivo, a execução automática de tarefas em determinados tempos pré-estabelecidos, variando de minutos, horas, dias, até semanas ou meses, para aplicar suas funcionalidades. Sua adição em aplicações permite a automação de processos internos, sem demandar de atualizações periódicas manuais por parte do usuário ou desenvolvedor.

No objeto reformulado, foi-se utilizado crons para realizar serviços referentes ao Sistema do Portal do Aluno, sendo tais funções: desativação automática de matrícula vigente, limpeza de processo com prazo de validade expirado, remover tipo do cartão do aluno cuja matrícula não está ativa e validação de processos caso haja alguma documentação inserida.

Figura 20 - Exemplo de Cron utilizado no Sistema, Cron de verificação de Ano Letivo para atualização de Matrícula Vigente

TCC - verificarAnoLetivoCron.ts

```
export function verificarAnoLetivoCron() {
  cron.schedule("0 0 1 1 *", async () => {
    const anoAtual = new Date().getFullYear();

    await AlunoMatriculaRepository.createQueryBuilder()
      .update()
      .set({ status_matricula: false })
      .where("ano_letivo < :anoAtual", { anoAtual })
      .execute();
  });
}

```

Fonte: O Autor, 2025.

5.3.8. Middlewares

Middlewares são todo o tipo de funções definidas entre a comunicação HTTP e a resposta final que será enviada para o cliente através da requisição de uma rota, funcionando como interceptadores que atuam na camada central entre a requisição e a resposta, tratando devidamente os dados que poderão ser enviados.

A importância da utilização de *middlewares*, deve-se ao controle das requisições realizadas no *back-end*, ao invés de concentrar toda a lógica por trás de um objeto controlador, usa-se os *middlewares* para separação de etapas de segurança e validações necessárias para que a requisição seja efetuada. Observa-se na Figura 21, um exemplo de utilização de *middleware*, demonstrando uma ocasião do uso desta funcionalidade como parâmetro de segurança no projeto, limitando o número de requisições para a função de login.

Figura 21 - Exemplificação de middleware para limitador de login

TCC - loginRateLimiterMiddleware.ts

```ts
export const loginRateLimiterMiddleware = rateLimit({
  windowMs: 10 * 60 * 1000,
  max: 10,
  standardHeaders: true,
  legacyHeaders: false,
  handler: (req, res, next) => {
    next(
      new TooManyRequestsError(
        "Muitas tentativas de login. Tente novamente mais tarde."
      )
    );
  },
});
```

Fonte: O Autor, 2025.

6. Considerações Finais

Com base no desenvolvimento do projeto e das análises apresentadas, este trabalho evidenciou a importância de uma reformulação estrutural e visual em aplicações institucionais, principalmente aqueles destinados à prestação de serviço público. A análise do objeto de estudo sobre o conjunto de aplicações pertencentes à Transportes Coletivos Pérola do Oeste, configurada como uma instituição de Administração Pública Indireta, abrangendo o *Website* Institucional dos serviços prestados, juntamente com o Sistema Portal do Aluno. Resultados da análise do objeto de estudo, evidenciaram a ausência de boas práticas de *design* responsivo, acessibilidade e organização arquitetural, pontos estes que comprometem significativamente a experiência do usuário e a credibilidade da instituição perante seu público.

Através da adoção de metodologias práticas, arquitetura de sistema definida com clareza por meio de funções bem definidas, escopo e planejamento de *design* de interface, foi possível desenvolver um conjunto de aplicações centralizadas na boa usabilidade para o usuário, com ênfase na distribuição do produto desenvolvido, para o público-alvo majoritário, nos quais, obtém o acesso via dispositivos móveis.

Além disso, o projeto reforçou a relevância do uso de ferramentas modernas e de boas práticas de desenvolvimento, que contribuíram para uma estrutura otimizada, segura e de

fácil leitura e manutenção. A arquitetura desenvolvida possibilitou a criação de uma base sólida para futuras expansões, como adições de funcionalidades e melhores adaptações das aplicações, promovendo uma maior eficiência e escalabilidade no ambiente digital da instituição.

Também, se destaca importantes possibilidades para planejamentos futuros do projeto, especialmente à sua implementação prática no ambiente institucional no âmbito do serviço público. Entre as propostas, enfatiza-se o desenvolvimento de um painel administrativo cuja função será a gestão centralizada do conteúdo do Portal do Aluno. A integração dessa camada administrativa às aplicações reformuladas irá garantir uma maior autonomia operacional, ampliando a sua gestão e assegurando que o ecossistema digital da instituição se mantenha eficiente e alinhado às demandas dos usuários.

Por fim, a conclusão se deve que o processo de reformulação proposto inicialmente alcançou os objetivos estabelecidos, proporcionando uma experiência de usuário mais intuitiva e acessível, estando coerente com as demandas atuais de comunicação administrativa no meio digital. Este trabalho, demonstra como a consolidação do papel da engenharia de *software*, se apresenta como ferramenta essencial na modernização de serviços públicos e na ampliação do acesso à informação.

Referências

ASSEMBLEIA LEGISLATIVA DO ESTADO DE MINAS GERAIS (ALMG). Administração pública – informações gerais. Disponível em: https://politicaspublicas.almg.gov.br/temas/administracao_publica/entenda/informacoes_gerais.html?tagNivel1=176&tagAtual=176. Acesso em: 10 jun. 2025.

BAHIA. Assessoria de Comunicação da Secretaria da Administração do Estado da Bahia. Reformulação em site amplia acesso a informações sobre estrutura do Estado. Salvador: Governo do Estado da Bahia, 25 ago. 2023. Disponível em: https://www.ba.gov.br/administracao/noticia/2024-02/7721/reformulacao-em-site-amplia-acesso-informacoes-sobre-estrutura-do-estado. Acesso em: 14 jun. 2025.

CAMARA, Lucas. O que é um Website: definição, exemplos e importância. Disponível em: https://camaraux.com.br/o-que-e-um-website-definicao-exemplos-importancia/ Acesso em: 12 jun. 2025.

CENTENARO, Jonas. DESENVOLVIMENTO DE UM SOFTWARE WEB PARA GERENCIAMENTO DE REQUISITOS DE SOFTWARE, Francisco Beltrão: Universidade Tecnológica Federal do Paraná, 2014. Disponível em: https://repositorio.utfpr.edu.br/jspui/bitstream/1/20053/3/FB_DESIDM_I_2014_09.pdf. Acesso em: 14 jun. 2025.

COMITÊ GESTOR DA INTERNET NO BRASIL. 92 milhões de brasileiros acessam a internet apenas pelo telefone celular, aponta TIC Domicílios 2022. São Paulo: CGI.br, 2022. Disponível em:

https://www.cgi.br/noticia/releases/92-milhoes-de-brasileiros-acessam-a-internet-apenas-pelo-telefone-celular-aponta-tic-domicilios-2022/. Acesso em: 17 set. 2025.

DICIONÁRIO PRIBERAM DA LÍNGUA PORTUGUESA. Reformulação. Lisboa: Priberam Informática, 2024. Disponível em: https://dicionario.priberam.org/reformulação. Acesso em: 23 out. 2025.

DOMINGUES, Thales dos Santos. Reformulação de um Sistema de Gerenciamento de Cartões de Crédito para atender os princípios SOLID. UFMT (Universidade Federal de Mato Grosso). Disponível em: https://bdm.ufmt.br/bitstream/1/2004/1/TCC_2021_Thales%20dos%20Santos%20Domingues.pdf. Acesso em: 20 set. 2025.

FLANAGAN, David. *JavaScript: o guia definitivo*. 6. ed. Porto Alegre: Bookman, 2012.

FONSECA, Elton. O que é ORM? Disponível em: https://www.oracle.com/br/database/what-is-database/. Acesso em: 06 jul. 2025.

MORORÓ, Jadson Faustino. UM ESTUDO COMPARATIVO ENTRE JAVASCRIPT E TYPESCRIPT. UFC (Universidade Federal do Ceará). Disponível em: https://repositorio.ufc.br/ri/bitstream/riufc/78817/1/2024_tcc_jfmororo.pdf. Acesso em: 24 set. 2025.

NIELSEN, Jakob. 10 Usability Heuristics for User Interface Design. [S.l.]: Nielsen Norman Group, 24 abr. 1994. Disponível em: https://www.nngroup.com/articles/ten-usability-heuristics/. Acesso em: 28 out. 2025.

ORACLE. O que é banco de dados? Disponível em: https://www.oracle.com/br/database/what-is-database/. Acesso em: 14 jun. 2025.

RODRIGUES, Cleber dos Santos. Gerenciamento das comunicações do projeto: relação governo-cidadão na reestruturação do website de uma prefeitura. USP (Universidade de São Paulo). Disponível em: https://periodicos.ufsm.br/pap/article/view/86287/64817. Acesso em: 14 jun. 2025.

SALES, Beatriz. O que é administração direta e indireta. *Jusbrasil*, 2022. Disponível em: https://www.jusbrasil.com.br/artigos/o-que-e-administracao-direta-e-indireta/1263096094. Acesso em: 12 jun. 2025.

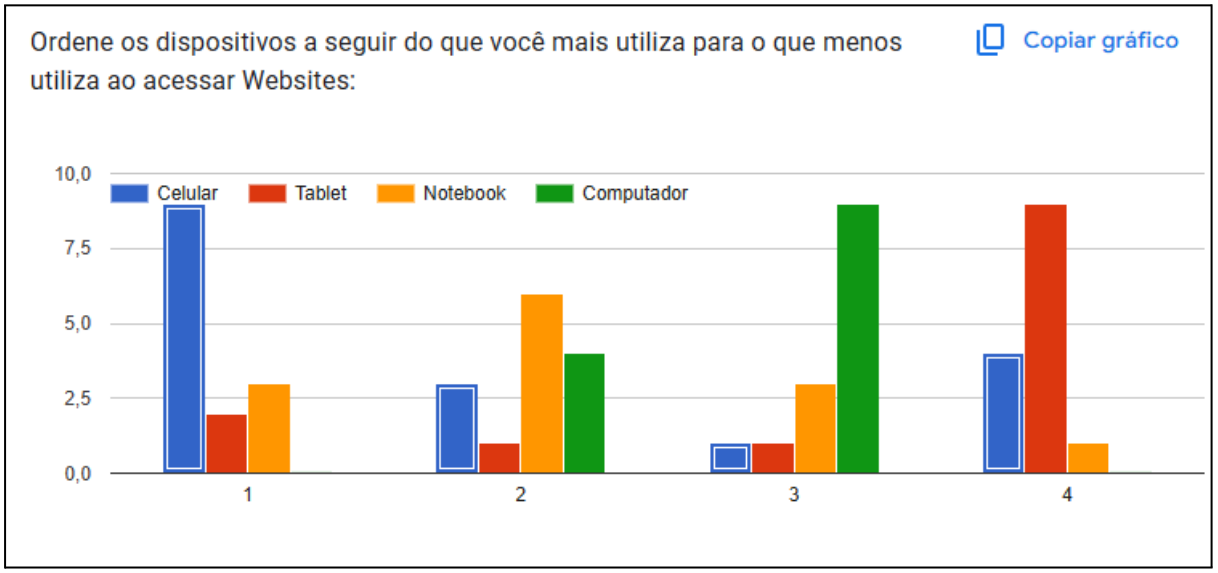
STRIPE. The Developer Coefficient: software engineering efficiency and its $3 trillion impact on global GDP. Stripe, set. 2018. Disponível em: https://stripe.com/files/reports/the-developer-coefficient.pdf. Acesso em: 20 set. 2025.

WELLER, Joe. SmartSheet: Website redesign guide. Disponível em: https://pt.smartsheet.com/content/website-redesign-guide. Acesso em: 13 jun. 2025.

APÊNDICE A - FEEDBACKS VIA FORMULÁRIOS

Conforme previamente mencionado na Seção 4.4, a utilização dos resultados obtidos por meio dos formulários do Google disponibilizados foi de fundamental importância para o desenvolvimento de melhorias no sistema, bem como para a obtenção de *feedbacks* necessários à demonstração da relevância da temática abordada neste artigo. Foram elaboradas diversas perguntas relacionadas tanto ao conjunto de aplicações atuais do Objeto de Estudo quanto às percepções e sugestões sobre a proposta de reformulação.

Figura A.1 – Resultados sobre utilização das aplicações por dispositivo usado

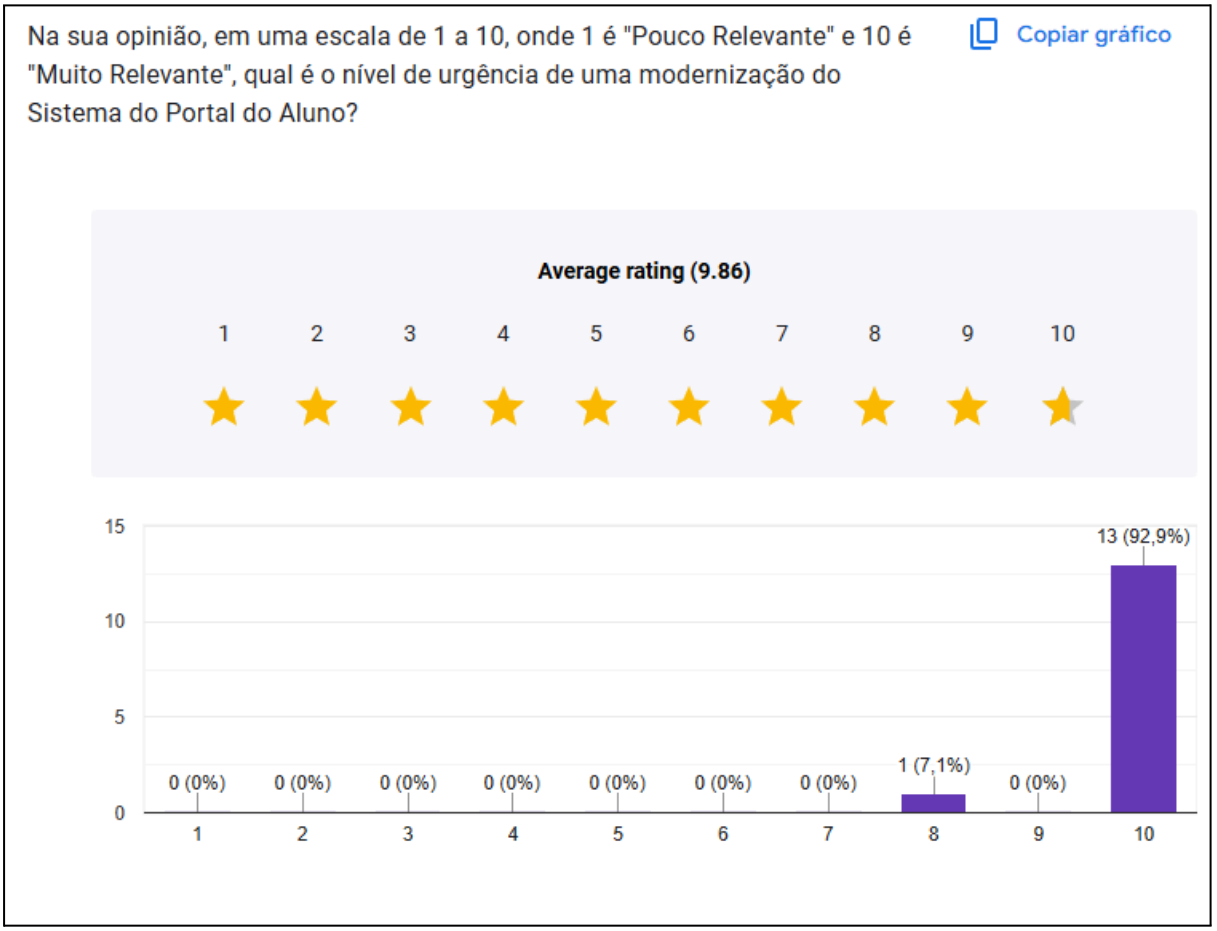


Fonte: O Autor, 2025.

Ao observar a Figura A.1, nota-se um dado que corrobora a citação do Comitê Gestor da Internet no Brasil, mencionada anteriormente na Seção 5.2. O gráfico demonstra que a maior parte dos usuários acessam as aplicações por meio de dispositivos móveis, como *smartphones*. Dessa forma, a reformulação com foco em *mobile* revela-se uma pauta pertinente, atendendo à necessidade de reestruturação na visibilidade de informações e funcionalidades voltadas a esses dispositivos. Além disso, conforme ilustrado na sequência, os resultados obtidos por meio do formulário evidenciam uma percepção significativa acerca da importância de uma modernização do Sistema do Portal do Aluno, indicando uma insatisfação considerável com o modelo atualmente disponibilizado.

No formulário, também foram coletados *feedbacks* com sugestões de melhorias e relatos sobre *bugs* identificados durante o uso da aplicação em ambiente de produção. Ressalta-se que a existência de um formulário ou outro método de coleta contínua dessas informações é de extrema importância para o funcionamento eficiente de um sistema. Os *feedbacks* dos usuários constituem uma ferramenta essencial para a identificação de inconsistências e ajustes necessários, promovendo uma melhor comunicação entre usuário e sistema, além de favorecer a transparência e fortalecer a credibilidade da instituição responsável pelo mecanismo digital.

Figura A.2 – Resultados sobre a relevância de uma modernização do Sistema



Fonte: O Autor, 2025.