Revista Ibero-Americana de Humanidades, Ciências e Educação — REASE

doi.org/10.51891/rease.v12i1.23467

SOFTWARE DE RELATÓRIOS PARA ORGANIZAÇÕES DE TERCEIRO SETOR

DEVELOPMENT OF A REPORTING SOFTWARE FOR THIRD SECTOR ORGANIZATIONS

Gabriel Martins Nepomuceno[1]
Edilson Carlos Silva Lima[2]
Jonathan Araújo Queiroz[3]
Patrícia Regia Nicácio Freire[4]

RESUMO: Organizações do terceiro setor frequentemente dependem de sistemas legados ineficientes. Para solucionar gargalos operacionais em uma ONG, foi desenvolvida a plataforma NarratePro, substituindo um sistema legado em PHP por uma nova arquitetura em Java com Spring e Padrões de Projeto (Facade, Adapter, Strategy). Este estudo avalia a eficácia da nova plataforma. A metodologia empregou uma abordagem quali-quantitativa, com testes de usabilidade remotos e questionários pós-teste aplicados a 9 voluntários da organização (n=9). Os resultados indicam alta aceitação (88,9%) da usabilidade geral e das funcionalidades simplificadas pelo padrão **Facade** (criação de agendamentos). Contudo, os testes revelaram uma falha crítica na funcionalidade de importação de dados (ligada aos padrões **Adapter** e **Strategy**), que obteve 66,7% de reprovação devido a um bug de implementação na consulta ao banco de dados. Conclui-se que, embora o padrão Facade tenha melhorado a experiência do usuário, a validação empírica foi crucial para identificar falhas de implementação nos padrões de integração de dados, demonstrando que um bom design arquitetural não garante o sucesso sem uma implementação validada.

Palavras-chave: NarratePro. Padrões de Projeto. Java Spring. Teste de Usabilidade.

ABSTRACT: Third-sector organizations often rely on inefficient legacy systems. To solve operational bottlenecks in an NGO, the NarratePro platform was developed, replacing a legacy PHP system with a new architecture in Java with Spring and Design Patterns (Facade, Adapter, Strategy). This study evaluates the effectiveness of the new platform. The methodology employed a quali-quantitative approach, with remote usability testing and post-test questionnaires administered to 9 volunteers (n=9). The results indicate high acceptance (88.9%) of the general usability and the functionalities simplified by the **Facade** pattern (scheduling creation). However, testing revealed a critical failure in the data import functionality (linked to the **Adapter** and **Strategy** patterns), which had a 66.7% failure rate due to an implementation bug in the database query. It is concluded that while the Facade pattern improved the user experience, empirical validation was crucial to identify implementation failures in the data integration patterns, demonstrating that good architectural design does not guarantee success without a validated implementation.

Keywords. NarratePro. Design Patterns. Java Spring. Usability Testing.

[1] Discente da Universidade CEUMA.
[2] Orientador, Docente da Universidade CEUMA.
[3] Docente da Universidade CEUMA.
[4] Docente da Universidade CEUMA.

1 INTRODUÇÃO

Organizações do terceiro setor desempenham um papel crucial na sociedade, mas possuem muitos desafios na sua eficiência operacional. Essas organizações enfrentam um cenário complexo, distinto das empresas comerciais, devido aos seus "objetivos, suas fontes limitadas de financiamento e a base voluntária da força de trabalho". Esses desafios incluem restrições financeiras, problemas culturais e "restrições de expertise técnica" (Nikita et al., 2024). A literatura também aponta que o setor sem fins lucrativos, em comparação com o setor privado, historicamente tem ficado para trás na adoção de tecnologias digitais. Frequentemente, restrições financeiras severas resultam "no uso estagnado de sistemas... ou uma superdependência de software... que atende apenas parcialmente aos requisitos da organização" (Nikita et al., 2024). Essas ferramentas obsoletas criam gargalos na gestão de dados, dificultam a integração de informações e comprometem a geração de relatórios de impacto, minando a análise estratégica e a transparência. Esta é uma lacuna crítica, visto que a literatura aponta a digitalização como um "motor-chave para o crescimento e sucesso" de organizações sociais, precisamente por permitir "tomar decisões informadas, aumentar a transparência, reduzir custos e impulsionar a inovação" (Damavandi & Berardi, 2025).

Para solucionar este desafio em uma ONG específica, foi desenvolvida a plataforma **NarratePro**. O projeto consistiu na remodelagem de um sistema de gestão de atividades, migrando de uma aplicação legada em PHP para uma arquitetura moderna em **Java**, utilizando o framework **Spring** e aplicando **Padrões de Projeto (Design Patterns)**. O objetivo era criar uma solução robusta, escalável e de fácil manutenção, capaz de otimizar o registro de atividades e a importação de dados de múltiplos formatos. Esta iniciativa responde diretamente a uma demanda da literatura pela "profissionalização da gestão no terceiro setor" , vista como essencial para que estas organizações "se tornem autónomos financeiramente do estado apostando na sua própria sustentabilidade" (Fonseca et al., 2024).

Contudo, a literatura de engenharia de software aponta para uma "lacuna entre design e implementação" (design-implementation gap), onde uma arquitetura teoricamente correta pode falhar na prática. Este artigo foca na validação empírica do NarratePro. O problema de pesquisa é: como a nova arquitetura, baseada em Padrões de Projeto, é percebida pelos usuários finais e quais falhas críticas de implementação um teste de usabilidade pode revelar?

Para responder a esta questão, o artigo apresenta os resultados de testes de usabilidade aplicados a 9 voluntários da organização. Após esta introdução, o capítulo 2 apresenta trabalhos

relacionados. O capítulo 3 detalha os métodos, incluindo a arquitetura do software (estudo de caso) e a pesquisa qualitativa com usuários. O capítulo 4 apresenta os resultados obtidos, e o capítulo 5 discute as descobertas. Por fim, o capítulo 6 apresenta a conclusão.

2 TRABALHOS RELACIONADOS

Este estudo se situa na interseção entre a validação de software para o terceiro setor e a aplicação prática de padrões de arquitetura de software, dialogando com trabalhos anteriores que abordaram o desenvolvimento de sistemas de gestão para ONGs.

2.1 SisOng: Sistema Gerenciador de ONGs

O trabalho de Machado (2014) apresenta o desenvolvimento do "SisOng", um sistema web (em Java, JSF e PostgreSQL) para o gerenciamento de uma ONG específica, a ONG ORAR.

Vantagens e Contribuições: A principal contribuição do SisOng é a criação de uma solução centralizada para sanar a carência de software da organização, que não era atendida por sistemas genéricos . O sistema visa organizar cadastros de clientes (beneficiários), doadores e o gerenciamento de doações , sendo uma solução web acessível e específica para os processos da entidade.

Desvantagens e Limitações: A validação do sistema é uma limitação notável. A seção de "Testes de Usabilidade" (4.9.3) não se baseia em validação empírica com usuários reais executando tarefas . Em vez disso, consiste em uma lista de verificação (checklist) baseada em critérios ergonômicos (ErgoList), como a presença de rótulos nos campos . Não há, portanto, uma avaliação prática que revele falhas de implementação ou a percepção do usuário sobre a arquitetura.

2.2 Sistema de Gerenciamento de Informações para a ONG Patas Dadas

O trabalho de Fischborn (2019) descreve um sistema de gerenciamento para a ONG Patas Dadas, focado em substituir o uso de planilhas Excel descentralizadas . A solução (em Node.js e MySQL) é uma aplicação web para voluntários administrarem tabelas de controle de materiais, fluxo de caixa, eventos e parceiros .

Vantagens e Contribuições: A aplicação resolve um problema crítico de gestão de dados, centralizando informações e, crucialmente, criando um "registro de movimentação dos dados" (logs de alteração) para garantir a confiabilidade que o Excel não oferecia . A validação é mais

robusta que a do Artigo 1, empregando Avaliação Heurística (Nielsen) e uma pesquisa qualitativa (entrevista) com uma voluntária .

Desvantagens e Limitações: O foco do trabalho e de sua validação é a *utilidade* (resolver o problema do Excel) e a *usabilidade* geral da interface (CRUD) . A discussão arquitetural limita-se ao padrão MVC . O estudo não utiliza o teste de usabilidade como ferramenta para validar a *implementação* de padrões de arquitetura de back-end (como os padrões GoF), focando apenas na experiência de front-end.

2.3 Diferencial do NarratePro

Diferente dos trabalhos supracitados, que focam na *criação* de sistemas de gestão para ONGs, este artigo foca na *validação empírica da arquitetura de software* em um sistema já existente (NarratePro). Enquanto Machado (2014) e Fischborn (2019) validam a utilidade e a usabilidade geral de suas interfaces, o diferencial deste estudo é a análise da "lacuna entre design e implementação".

O diferencial deste trabalho é utilizar um teste de usabilidade ágil para avaliar empiricamente a implementação desses padrões de arquitetura complexos. Expomos como a experiência do usuário final é diretamente impactada por decisões de design de back-end e como essa validação pode revelar falhas críticas de implementação (bugs) que a análise técnica teórica (como a do Relatório de Software inicial) não previu.

3 MÉTODOS

A pesquisa foi conduzida de modo a ser aplicada e exploratória, visando não apenas o desenvolvimento de um artefato tecnológico, mas a compreensão dos fenômenos de interação humana com arquiteturas de software complexas. O desenho metodológico foi estruturado em duas etapas complementares: o estudo de caso, focado na construção da solução, e a pesquisa qualitativa, voltada para a validação empírica com os usuários finais.

3.1 O Estudo de Caso: Definição e Construção

O objeto de análise deste estudo é a plataforma NarratePro, uma aplicação de gestão desenvolvida para atender às demandas específicas de uma Organização Não-Governamental (ONG) atuante no terceiro setor. A escolha deste caso justifica-se pela sua representatividade em relação ao problema da obsolescência tecnológica: a organização dependia de um sistema legado em PHP que apresentava falhas críticas de segurança e usabilidade, um cenário comum

que dificulta a profissionalização de entidades sem fins lucrativos. O processo de coleta de dados para a definição dos requisitos do novo sistema ocorreu por meio de duas técnicas principais: a observação direta dos fluxos de trabalho dos voluntários e a análise documental do código-fonte e banco de dados do sistema legado. Essas fontes primárias permitiram mapear as regras de negócio essenciais que precisavam ser preservadas e os gargalos que demandavam refatoração arquitetural. Com base nos dados coletados, a infraestrutura tecnológica foi selecionada visando robustez e manutenibilidade. Para o Back-end, adotou-se a linguagem Java com o Spring Framework (Spring Boot, Spring Web MVC e Spring Data JPA). No Front-end, utilizou-se o Thymeleaf para renderização no servidor e o framework Bootstrap para a interface gráfica. O armazenamento de dados foi confiado ao SGBD MySQL, garantindo a integridade das informações migradas.

3.2 Metodologia da Pesquisa Qualitativa

A etapa de validação adotou uma abordagem qualitativa, cujo foco principal é a interpretação e compreensão dos fenômenos observados, indo além da simples métrica de sucesso ou fracasso da tarefa. O objetivo foi capturar a percepção subjetiva dos usuários sobre a complexidade oculta pelos padrões de projeto. Como técnica de coleta, utilizou-se um protocolo de teste de usabilidade remoto assíncrono, seguido pela aplicação de questionários que combinaram escalas de Likert com perguntas abertas, funcionando como entrevistas semiestruturadas. Os instrumentos foram desenhados para estimular os participantes a verbalizarem não apenas os erros técnicos (bugs), mas também suas sensações de conforto, clareza e eficiência ao utilizar a nova ferramenta. Os dados obtidos foram organizados e submetidos à Análise de Conteúdo. As respostas discursivas foram categorizadas em três eixos temáticos: "Barreiras de Implementação", "Percepção de Melhoria" e "Sugestões Evolutivas". Essa análise permitiu triangular os dados quantitativos de desempenho com os relatos qualitativos, identificando, por exemplo, que a falha na importação de dados não era apenas um erro técnico, mas um ponto de ruptura na confiança do usuário com o sistema.

4 RESULTADOS

Os resultados da pesquisa mostram a realização das duas etapas descritas na metodologia: o desenvolvimento tecnológico e a validação empírica. A seguir, apresenta-se o estudo de caso com os detalhes da implementação do NarratePro, seguido pela análise quali-quantitativa dos

dados coletados junto aos voluntários, demonstrando como a arquitetura proposta impactou a operação real da organização.

4.1 O Produto: Plataforma NarratePro

Como resultado do desenvolvimento, a plataforma foi entregue com fluxos otimizados para a gestão da ONG. A arquitetura segue o padrão MVC, onde os controladores interagem com uma camada de Facade (a fachada), que por sua vez orquestra os serviços de negócio (como AgendamentoService e AgendamentoImportService). As principais telas e funcionalidades submetidas à validação foram:

Criação de Agendamento (Padrão Facade)

Para garantir que alguma complexidade futura da criação de registros não impactasse a interface do usuário, a lógica foi centralizada no back-end. A classe AgendamentoFacade (Figura 1) atua como um orquestrador, encapsulando as chamadas de serviço e transações, simplificando a comunicação com o controlador e evitando que regras de negócio vazem para a camada de apresentação.

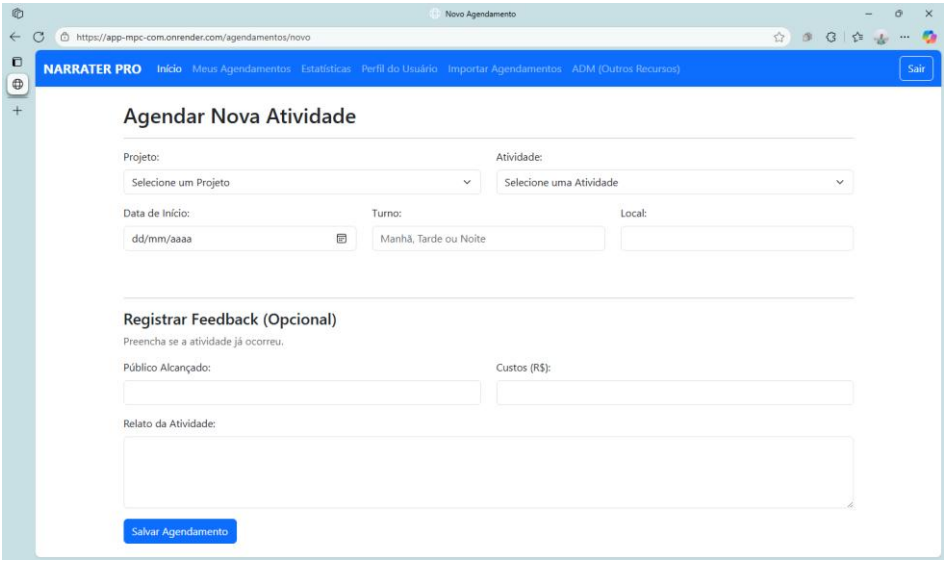
Figura 1 – Implementação do Padrão Facade para centralização de regras de negócio.


```
@Service
public class AgendamentoFacade {

    private final AgendamentoService agendamentoService;
    private final AgendamentoImportService agendamentoImportService;


    @Autowired
    public AgendamentoFacade(AgendamentoService agendamentoService,
AgendamentoImportService agendamentoImportService) {
        this.agendamentoService = agendamentoService;
        this.agendamentoImportService = agendamentoImportService;
    }

    @Transactional
    public Agendamento criarAgendamentoIndividual(Agendamento agendamento) {
        System.out.println("Facade: Iniciando processo de agendamento
individual ... ");

        Agendamento agendamentoSalvo = agendamentoService.salvar(agendamento);

        realizarAcoesPosCriacao(agendamentoSalvo);

        System.out.println("Facade: Processo de agendamento individual
concluído.");
        return agendamentoSalvo; }
```

Fonte: NEPOMUCENO, et al., 2025.

Uma interface simplificada (Figura 1) para registrar novas atividades. O usuário interage com um formulário único, e o AgendamentoFacade orquestra no back-end a validação, persistência e notificação, simplificando a experiência do usuário.

Figura 2 – Interface simplificada para registro de novas atividades.



Fonte: NEPOMUCENO, et al., 2025.

Importação de Dados (Padrões Adapter/Strategy)

A robustez na importação de dados foi alcançada através do padrão Strategy. A classe CsvAgendamentoImporter (Figura 3) exemplifica como o sistema seleciona dinamicamente o algoritmo de tratamento baseado no tipo de arquivo, desacoplando a lógica de seleção da lógica de persistência e permitindo a extensão para novos formatos sem alterar o código base.

Figura 3 – Implementação da Estratégia de Importação (Pattern Strategy).

```
@Component
public class CsvAgendamentoImporter implements AgendamentoImporter {

    private final AgendamentoImportAdapter adapter;
    private final AgendamentoRepository agendamentoRepository;
    private final UsuarioRepository usuarioRepository;

    public CsvAgendamentoImporter(AgendamentoImportAdapter adapter,
AgendamentoRepository agendamentoRepository, UsuarioRepository
usuarioRepository) {
        this.adapter = adapter;
        this.agendamentoRepository = agendamentoRepository;
        this.usuarioRepository = usuarioRepository;
    }

    @Override
    public boolean supports(String fileName) {
        return fileName != null && fileName.toLowerCase().endsWith(".csv");
    }

    @Override
    public ImportResult importar(MultipartFile file, Principal principal)
throws IOException {
        Usuario usuarioLogado =
usuarioRepository.findByEmail(principal.getName())
                .orElseThrow(() -> new IllegalStateException("Usuário logado
```

Fonte: NEPOMUCENO, et al., 2025.

Complementarmente, a integridade dos dados migrados do sistema antigo é assegurada pelo padrão Adapter. Conforme demonstrado na Figura 4, a classe AgendamentoImportAdapter realiza a tradução e validação dos objetos de transferência de dados (DTOs) legados para as entidades de domínio atuais, aplicando regras de negócio essenciais antes da persistência.

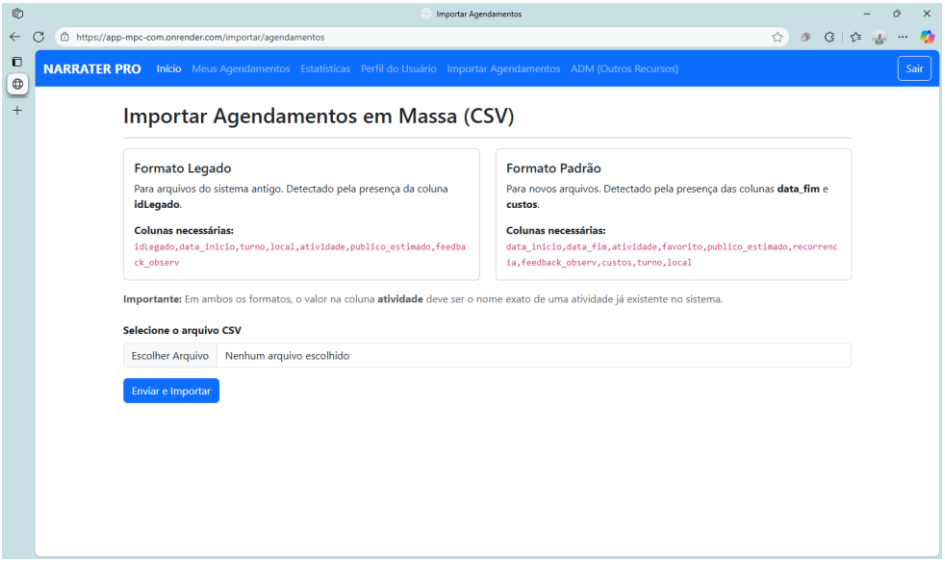
Figura 4 – Lógica de Adaptação de Dados Legados (Pattern Adapter).

```
@Component
public class AgendamentoImportAdapter {

    private final AtividadeRepository atividadeRepository;

    @Autowired
    public AgendamentoImportAdapter(AtividadeRepository atividadeRepository) {
        this.atividadeRepository = atividadeRepository;
    }

    // O método fromLegacyDto não precisa de alterações.
    public Agendamento fromLegacyDto(LegacyAgendamentoCsvDto dto, Usuario
usuarioLogado) {
        // ... (código existente)
        if (dto.getAtividade() == null || dto.getAtividade().trim().isEmpty())
{
            throw new IllegalArgumentException("A coluna 'atividade' está
vazia para o registro com idLegado: " + dto.getIdLegado());
        }
        Atividade atividade =
atividadeRepository.findByNome(dto.getAtividade().trim())
                .orElseThrow(() -> new RuntimeException("Atividade '" +
dto.getAtividade() + "' não foi encontrada no banco de dados."));
        Agendamento agendamento = new Agendamento();
        agendamento.setData_inicio(LocalDate.parse(dto.getData_inicio()));
        agendamento.setTurno(dto.getTurno());
```

Fonte: NEPOMUCENO, et al., 2025.

Essa lógica suporta a funcionalidade de Importação de Dados (Figura 5), que permite aos administradores carregarem arquivos CSV. O sistema utiliza o padrão Strategy para selecionar o importador correto (ex: "CSV Legado" ou "CSV Padrão") e o padrão Adapter para traduzir os dados.

Figura 5 – Interface de importação de dados (baseada em Adapter/Strategy).



Fonte: NEPOMUCENO, et al., 2025.

Limitação Metodológica (Padrão Observer)

Uma limitação deste estudo foi a incapacidade de avaliar o padrão Observer. A funcionalidade de pontuação de usuário, que seria acionada pelo Observer, não teve sua implementação finalizada. Portanto, a validação empírica deste padrão permanece como um trabalho futuro.

4.2 Pesquisa Quali-Quantitativa

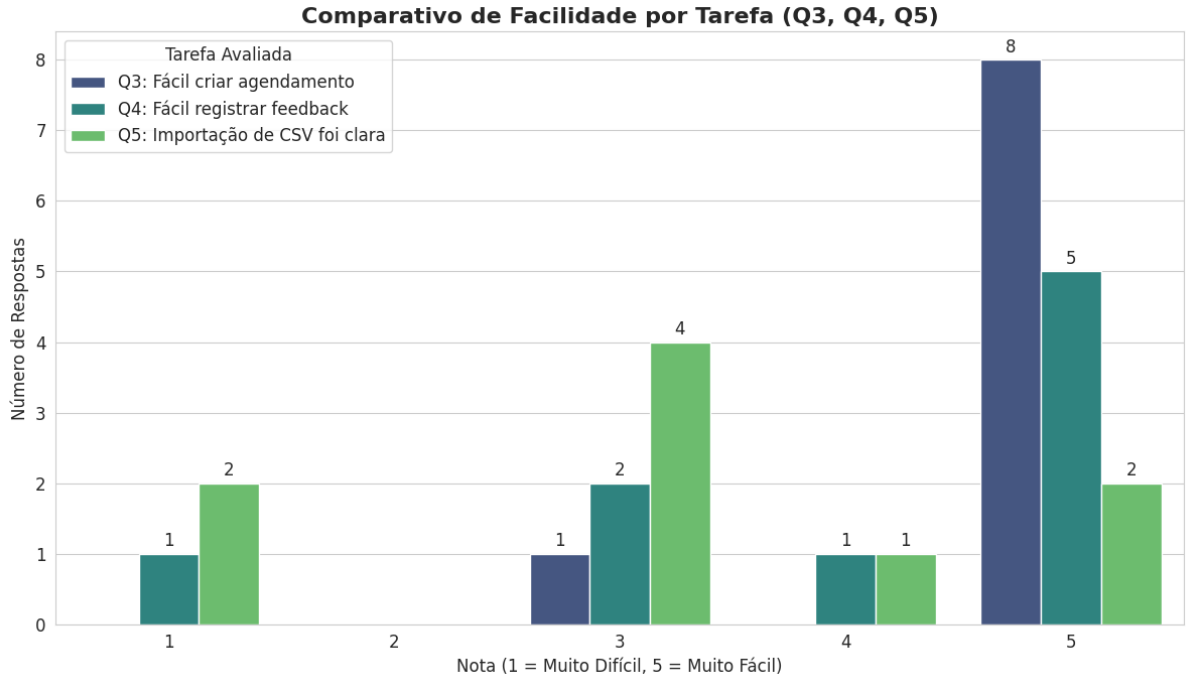
Esta seção apresenta os dados coletados dos 9 voluntários que testaram a plataforma NarratePro. Os resultados são divididos em análise quantitativa da percepção de usabilidade e análise qualitativa das falhas e sugestões.

Usabilidade e do Padrão Facade

Na primeira pergunta, "Como você avalia a facilidade geral de uso da plataforma NarratePro?", observou-se uma avaliação majoritariamente positiva. Conforme ilustrado no Gráfico 1, 88,9% (8 de 9 usuários) avaliaram a experiência como "Fácil" ou "Muito Fácil" (Notas 4 ou 5 na escala Likert).

Essa alta aprovação foi espelhada na avaliação da tarefa de "Criar um novo agendamento" (ligada ao padrão Facade), que também obteve 88,9% de aprovação.

Gráfico 1 – Avaliação da usabilidade das tarefas principais (n=9).



Fonte: NEPOMUCENO, et al., 2025.

Falha Crítica na Importação (Padrões Adapter/Strategy)

Em nítido contraste, a funcionalidade de "Importação de Arquivos", crucial para validar os padrões Adapter e Strategy, foi reprovada. Apenas **33,3% (3 de 9 usuários)** avaliaram a tarefa positivamente. Isso significa que **66,7% (6 de 9 usuários)** não conseguiram completar a tarefa ou a avaliaram negativamente.

A explicação para essa falha não estava na usabilidade (o design da interface), mas sim em um bug crítico de implementação, revelado na análise qualitativa. Um usuário reportou o erro exato:

"tive sim [dificuldade], ao importar um dos arquivos tive esse erro: Ocorreram 3 erros: Linha Padrão (Data...): **Query did not return a unique result: 2 results were returned**"

Este feedback qualitativo foi crucial, pois identificou uma falha de back-end (um erro de consulta no Spring Data JPA) que comprometia toda a funcionalidade dos padrões Adapter e Strategy, invalidando a hipótese de sua implementação bem-sucedida.

Sugestões de Melhoria Na pergunta aberta sobre melhorias, as sugestões dos usuários focaram em otimizações de usabilidade, principalmente em "Melhoria de Formulários".

(Nota: O bug crítico da importação foi classificado como falha, não como sugestão.)

5 DISCUSSÃO

A análise dos resultados revela uma dicotomia: um sucesso de usabilidade impulsionado pelo padrão **Facade** e uma falha funcional crítica na implementação dos padrões **Adapter** e **Strategy**. A hipótese de que o padrão Facade simplificaria a experiência do usuário foi plenamente validada. A aprovação de 88,9% na criação de agendamentos e na facilidade geral, corroborada pelo feedback qualitativo de que o fluxo era "bem objetivo" e "muito fácil", demonstra que a arquitetura cumpriu seu objetivo. Ao ocultar a complexidade do subsistema (múltiplos serviços, validações), o Facade entregou uma experiência de usuário limpa e eficiente, muito superior à do sistema legado.

Este resultado é particularmente significativo, pois aborda diretamente as principais dificuldades para a modernização do terceiro setor apontadas pela literatura. Um estudo de Fonseca et al. (2024) com gestores de ONGs (IPSS) identificou que as barreiras centrais para a adoção de novas ferramentas de gestão são a "dificuldade em adaptar tais ferramentas há realidade das instituições", a "resistência há mudança" e a "falta de formação e preparação dos

recursos humanos". A alta aceitação do Facade sugere que seu design simplificado foi eficaz em mitigar essa resistência e se adaptar à realidade do usuário, superando a barreira de usabilidade que frequentemente leva projetos à falha.

A hipótese de que os padrões Adapter e Strategy solucionariam a importação de dados foi invalidada no teste prático (66,7% de reprovação). Este achado é um exemplo claro da "lacuna entre design e implementação". A **arquitetura** (o design) estava correta: usar o Strategy para selecionar o algoritmo de importação e o Adapter para traduzir os dados era a solução ideal para a manutenibilidade. Este tipo de desafio na fase de implementação, mesmo com um design arquitetural correto, é um ponto de falha comum. Machado (2014), por exemplo, ao desenvolver o sistema SISONG, também relatou que as principais dificuldades do projeto não estavam no design de requisitos, mas sim na implementação técnica, apontando a "dificuldade [em] trabalhar com Java Server Faces e outras tecnologias como Prime Faces e Bootstrap em conjunto". Isso demonstra que a complexidade da implementação — seja um bug na lógica de dados (como no NarratePro) ou o atrito na integração de ferramentas (como no SISONG) — é um obstáculo crítico que a análise teórica do design não prevê.

No entanto, a **implementação** falhou. O bug reportado ("Query did not return a unique result") indica uma falha na lógica de persistência (um erro de consulta SQL/JPQL) dentro de uma das estratégias concretas. Isso demonstra que, embora o padrão de projeto tenha organizado o código corretamente, uma falha na implementação da lógica de banco de dados comprometeu toda a funcionalidade. O teste empírico com usuários foi a única forma de revelar essa falha.

A falha na implementação desta funcionalidade de integração de dados não é um caso isolado. Fischborn (2019), ao relatar o desenvolvimento de um sistema de gestão para outra ONG, aponta a integração de dados como a "principal limitação" do projeto, afirmando que "a unificação dos bancos de dados não foi possível de ser realizada". Isso reforça que, embora o design (o uso dos padrões Adapter/Strategy) fosse a solução correta para o problema da integração, a sua *implementação* prática é um ponto de falha crítico, que apenas a validação empírica [com usuários] foi capaz de expor.

Uma limitação deste estudo foi a incapacidade de avaliar o padrão Observer. A funcionalidade de pontuação de usuário, que seria acionada pelo Observer, não foi incluída na plataforma e, portanto, sua validação empírica permanece como um trabalho futuro.

6 CONCLUSÃO

O objetivo deste trabalho foi desenvolver e avaliar a plataforma NarratePro, uma solução em Java/Spring com Padrões de Projeto para substituir um sistema legado em uma ONG. O estudo buscou validar se essa nova arquitetura resultaria em uma ferramenta mais robusta e com melhor usabilidade.

A pesquisa validou parcialmente as hipóteses. Apesar do sucesso do padrão Facade na melhoria da usabilidade, evidenciado pela aprovação majoritária da funcionalidade, houve falha na implementação dos padrões Adapter/Strategy, expondo "a lacuna entre design e implementação":

Este projeto demonstra que a engenharia de software e os Padrões de Projeto têm alto potencial para otimizar a gestão no terceiro setor. Contudo, reforça que apenas a validação empírica com usuários reais é capaz de expor falhas críticas de implementação, servindo como a verdadeira métrica de sucesso para uma arquitetura de software.

Como trabalhos futuros, os dados definem um roteiro claro:

1. Correção prioritária do bug "Query did not return a unique result" no módulo de importação.
2. Implementação das melhorias de UX sugeridas (autocompletar e salvar opções em formulários).
3. Reaplicação de um ciclo de testes para validar a correção do bug e, desta vez, avaliar a eficácia do padrão Observer.
4. Estudo estruturado da solução legada atual da organização, para analisar a viabilidade e o impacto da substituição integral do sistema legado pelo sistema NarratePro
5. Implementação de novas funcionalidades, como visualização das atividades da semana/mês num formato de calendário, diferentes filtros para visualização de estatísticas, e ferramentas de IA como chatbots para facilitar ainda mais o agendamento de atividades e seus feedbacks.

REFERÊNCIAS

1. DAMAVANDI S, BERARDI L. **Exploring the impact of digitalization on the social economy: a systematic literature review**. Management Control, 2024; 3: 61-85. Disponível em: https://www.torrossa.com/it/resources/an/5914479

2. FISCHBORN K. **Sistema de Gerenciamento de Informações para a ONG Patas Dadas**. Monografia (Bacharel em Ciência da Computação) – Instituto de Informática. Universidade Federal do Rio Grande do Sul, Porto Alegre, 2019. Disponível em: https://lume.ufrgs.br/handle/10183/198552

3. FONSECA SMSR, et al. **A importância da implementação da gestão estratégia nas organizações do terceiro setor**. e³ – Revista de Economia, Empresas e Empreendedores na

CPLP, 2024; 10(2). Disponível em: https://www.revistas.ponteditora.org/index.php/e3/article/view/939

4. MACHADO EC. **SISONG: SISTEMA GERENCIADOR DE ONGs**. Trabalho de Conclusão de Curso (Bacharel em Análise e Desenvolvimento de Sistemas) – Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2014. Disponível em: https://repositorio2.utfpr.edu.br/srv-riut-s02/api/core/bitstreams/35416bf7-057b-466f-aef8-48c5f6484118/content

5. NIKITA NA, et al. **Digital transformation in non-profit organizations: strategies, challenges, and successes**. Advanced International Journal of Multidisciplinary Research, 2024; 2(5). Disponível em: https://www.researchgate.net/publication/384667845_Digital_Transformation_in_Non-Profit_Organizations_Strategies_Challenges_and_Successes