REVISTA TÓPICOS
ISSN 2965-6672 | Qualis A2

BENEFÍCIOS DA ADOÇÃO DE UM SOFTWARE DE GERENCIAMENTO DE ESCALAS NO CONTEXTO ECLESIÁSTICO: DESENVOLVIMENTO E AVALIAÇÃO DO SISTEMA ESCALASERVIR

BENEFITS OF IMPLEMENTING SCHEDULING MANAGEMENT SOFTWARE IN A CHURCH CONTEXT: DEVELOPMENT AND EVALUATION OF THE ESCALASERVIR SYSTEM

Ciências Exatas e da Terra • 18/06/2026

REGISTRO DOI: 10.70773/revistatopicos/781743501

Cristiano César Xavier Marinho[1]

João Victor Lopes dos Santos[2]

Cleyton de Carvalho Souza[3]

Wellington Ávila[4]

RESUMO

A gestão das escalas de voluntários em comunidades religiosas é uma atividade administrativa rotineira que, em boa parte das organizações, permanece apoiada em planilhas eletrônicas e aplicativos de mensagem instantânea. Esse arranjo, embora difundido, tende a produzir sobreposição de turnos, esquecimentos, ruídos de comunicação e retrabalho administrativo. O presente artigo descreve o desenvolvimento e a avaliação do EscalaServir, um sistema web responsivo dedicado ao gerenciamento de escalas eclesiásticas, projetado segundo princípios da Engenharia de Software. A pesquisa, de natureza aplicada e abordagem qualiquantitativa, foi conduzida em quatro etapas articuladas, com modelagem em UML, banco de dados relacional e implementação em quatro sprints inspiradas no framework Scrum. A pilha tecnológica reuniu React no frontend, Node.js com Express no backend, PostgreSQL na persistência e Prisma como ORM, com testes automatizados em Jest e Cypress. Os resultados foram aferidos por indicadores técnicos e operacionais, expressos em tabelas e gráficos, contemplando tempo de resposta dos endpoints da API, cobertura de testes por módulo, conflitos detectados antes e após a adoção, e percepção qualitativa dos voluntários. Constatou-se redução superior a 90% no tempo de elaboração da escala mensal, cobertura média de testes de 86% e queda expressiva no número de conflitos relatados. Conclui-se que o sistema desenvolvido contribui de modo significativo para a gestão administrativa de organizações religiosas, à luz dos atributos da norma NBR ISO/IEC 25010.

Palavras-chave: Engenharia de Software; Sistemas web; Gestão de voluntários; Software de escalas; Qualidade de software.

ABSTRACT

Managing volunteer schedules in religious communities is a routine

administrative task that, in most organizations, continues to rely on spreadsheets and instant messaging apps. Although widespread, this approach tends to result in overlapping shifts, oversights, miscommunication, and administrative rework. This article describes the development and evaluation of EscalaServir, a responsive web system dedicated to managing church schedules, designed according to software engineering principles. The research, which is applied in nature and employs a qualitative-quantitative approach, was conducted in four interconnected phases, featuring UML modeling, a relational database, and implementation across four sprints inspired by the Scrum framework. The technology stack included React on the frontend, Node.js with Express on the backend, PostgreSQL for persistence, and Prisma as the ORM, with automated testing using Jest and Cypress. The results were assessed using technical and operational indicators, presented in tables and graphs, covering API endpoint response times, test coverage per module, conflicts detected before and after adoption, and qualitative feedback from volunteers. The study found a reduction of more than 90% in the time required to prepare the monthly schedule, an average test coverage of 86%, and a significant decrease in the number of reported conflicts. It is concluded that the system developed contributes significantly to the administrative management of religious organizations, in light of the requirements of the NBR ISO/IEC 25010 standard.
Keywords: Software Engineering; Web systems; Volunteer management; Rota software; Software quality.

1. INTRODUÇÃO

Comunidades religiosas, independentemente de tradição ou porte, mobilizam diariamente um conjunto expressivo de voluntários para

sustentar suas atividades. Recepção, equipes de louvor, ministérios de comunicação, grupos de oração, equipes de limpeza e tantas outras frentes funcionam apoiadas em escalas que precisam considerar disponibilidade, afinidade de competências e revezamento justo. Quando essa coordenação se dá em planilhas isoladas ou em conversas pulverizadas por aplicativos de mensagem, a rotina administrativa torna-se onerosa e frágil, conforme advertem Laudon e Laudon (2020) ao discutirem os custos invisíveis associados a processos não sistematizados.

Drucker (2006) sustenta que organizações do terceiro setor têm sua eficácia condicionada à capacidade de gerir pessoas com clareza e previsibilidade, dado que sua principal moeda é o tempo doado. Hudson (1999) acrescenta que a profissionalização da gestão nessas organizações não exige sofisticação onerosa, e sim instrumentos adequados ao seu porte e ritmo. Nesse cenário, a Engenharia de Software oferece um conjunto consolidado de métodos e ferramentas capazes de transformar rotinas manuais em processos sistematizados, auditáveis e acessíveis por dispositivos cotidianos.

O problema que motiva esta pesquisa pode ser sintetizado na seguinte indagação. Quais benefícios concretos podem ser obtidos com a substituição de processos manuais de gerenciamento de escalas por um software dedicado, no contexto de uma igreja? Subjacente a essa pergunta encontra-se a hipótese de que a aplicação de princípios consagrados da Engenharia de Software, como modelagem de requisitos, separação de responsabilidades, automação de regras de negócio e atenção à usabilidade, resulta em ganhos mensuráveis de eficiência operacional e em melhor percepção de pertencimento por parte dos voluntários.

A escolha do tema justifica-se por três frentes. Em primeiro lugar, pelo crescente movimento de digitalização que alcança organizações religiosas, ainda pouco contemplado por estudos acadêmicos. Em segundo lugar, pela carência de literatura específica sobre soluções de software voltadas a esse nicho, o que abre espaço para contribuições aplicadas. Por fim, pela relevância prática, uma vez que organizações religiosas estão entre as mais numerosas no terceiro setor brasileiro, segundo dados do Censo Demográfico (IBGE, 2023), e qualquer ganho em sua gestão tem efeito agregado considerável.

Estabelece-se como objetivo geral desenvolver e avaliar um sistema web de gerenciamento de escalas para uma igreja, denominado EscalaServir, considerando os pilares da Engenharia de Software. Como objetivos específicos, busca-se caracterizar a gestão manual de escalas e seus pontos de fragilidade; elicitar requisitos funcionais e não funcionais aplicáveis ao contexto eclesiástico; especificar a arquitetura técnica e implementar o sistema em ciclos iterativos; e comparar, por meio de indicadores técnicos e operacionais, o cenário anterior e posterior à adoção do EscalaServir.

O artigo está organizado em cinco seções, além desta introdução. A segunda apresenta o referencial teórico, articulando contribuições de autores da Engenharia de Software, dos Sistemas de Informação e da gestão do terceiro setor. A terceira descreve o percurso metodológico, com detalhamento das sprints de desenvolvimento. A quarta detalha a especificação, a arquitetura e a implementação do EscalaServir, e apresenta a avaliação técnica e operacional sustentada por tabelas e gráficos. A quinta sintetiza as considerações finais, aponta limitações e indica desdobramentos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção reúne os fundamentos teóricos que sustentam a pesquisa, organizados em quatro frentes complementares. Inicia-se pela discussão dos princípios da Engenharia de Software aplicáveis a aplicações web e móveis. Em seguida, examinam-se os sistemas de informação no contexto do terceiro setor. Passa-se, então, à literatura sobre gestão de voluntários e formação de escalas em organizações religiosas. Encerra-se com a apresentação do estado da arte de soluções de software voltadas ao agendamento e escalonamento.

2.1. Engenharia de Software no Desenvolvimento de Aplicações Web

A Engenharia de Software, conforme conceitua Sommerville (2019), pode ser entendida como a disciplina que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação até a manutenção evolutiva. Pressman e Maxim (2021) ampliam essa definição ao destacar que a disciplina articula métodos, ferramentas e procedimentos com vistas a obter sistemas confiáveis e que operem com eficiência em máquinas reais.

Para Wazlawick (2013), a Engenharia de Software brasileira tem absorvido com naturalidade o paradigma orientado a objetos e os métodos ágeis, especialmente em projetos de pequeno e médio porte. O autor observa que, em aplicações voltadas a organizações com recursos limitados, a aderência a boas práticas mostra-se mais decisiva do que a sofisticação da pilha tecnológica adotada. Pfleeger e Atlee (2013) corroboram essa percepção e advertem para o risco de superestimar artefatos formais em projetos cujo escopo é circunscrito.

A modelagem de requisitos, etapa fundamental no ciclo de desenvolvimento, é discutida em profundidade por Kotonya e Sommerville (1998). Os autores argumentam que falhas de requisitos respondem por proporção expressiva dos problemas observados em sistemas implantados, sobretudo quando se trabalha com usuários não técnicos. Esse alerta é particularmente relevante para o contexto deste estudo, pois os voluntários e líderes de uma igreja, em geral, não dominam o vocabulário formal da computação.

O Manifesto Ágil (Beck et al., 2001) consolidou um conjunto de valores que se tornou referência inescapável no desenvolvimento contemporâneo. Em sua leitura, indivíduos e interações importam mais do que processos e ferramentas, ao passo que software funcionando vale mais do que documentação extensiva. Schwaber e Sutherland (2020), na atualização do Guia do Scrum, reforçam o caráter empírico do processo e a importância de ciclos curtos de inspeção e adaptação, orientação que se mostrou útil ao sistema aqui descrito.

No plano da arquitetura, Fowler (2006) consolidou um vocabulário que tornou possível conversar com objetividade sobre padrões recorrentes em aplicações corporativas. A separação entre camadas de apresentação, domínio e persistência, ainda que adaptada a contextos enxutos, oferece ganhos diretos em manutenibilidade. Tal preocupação dialoga com as orientações da norma ABNT NBR ISO/IEC 25010 (ABNT, 2011), que estabelece atributos de qualidade aplicáveis a qualquer produto de software, independentemente de seu porte.

2.2. Sistemas de Informação e a Gestão do Terceiro Setor

Sistemas de informação, na conceituação de Laudon e Laudon (2020), consistem em conjuntos de componentes inter-relacionados que coletam, processam, armazenam e distribuem informações com vistas a apoiar a tomada de decisão e a coordenação em uma organização. Stair e Reynolds (2020) acrescentam que a presença efetiva de um sistema de informação em rotinas operacionais é o que distingue organizações capazes de aprender com sua própria operação daquelas presas a ciclos de improviso.

Quando se transpõe esse arcabouço para o terceiro setor, surgem particularidades importantes. Tenório (2009) sublinha que organizações sem fins lucrativos operam sob restrições orçamentárias severas, mas, em compensação, dispõem de capital humano motivado por valores compartilhados. Hudson (1999) acrescenta que tais organizações tendem a apresentar estruturas hierárquicas menos rígidas, o que se reflete em fluxos decisórios mais horizontais, porém menos formalizados.

O'Brien e Marakas (2013) destacam que sistemas de informação aplicados a organizações desse perfil precisam ser desenhados com sensibilidade ao baixo orçamento, à alta rotatividade de pessoal e ao grau variável de familiaridade tecnológica dos usuários. Drucker (2006), em obra clássica, argumenta que o desafio central nesse segmento é converter intenção em resultado, e que sistemas de informação bem desenhados tornam essa conversão menos custosa.

Em uma igreja, esse conjunto de fatores se manifesta de modo bastante específico. Há multiplicidade de ministérios, cada um com sua própria lógica de funcionamento; voluntários que se distribuem entre mais de uma frente; eventos sazonais que alteram a demanda;

e a necessidade de manter um histórico consistente das atribuições realizadas. Tais características reforçam a pertinência de soluções dedicadas, em detrimento de adaptações superficiais de ferramentas genéricas.

2.3. Gestão de Voluntários e Formação de Escalas em Organizações Religiosas

A formação de escalas em organizações religiosas, embora pouco explorada na literatura nacional de Engenharia de Software, assemelha-se em vários aspectos à gestão de turnos em ambientes hospitalares ou em serviços de plantão. Hudson (1999) observa que a equidade na distribuição de cargas é um dos principais elementos de manutenção da motivação em equipes voluntárias. Tenório (2009) acrescenta que a previsibilidade das atribuições reduz o desgaste relacional, sobretudo quando a coordenação envolve pessoas com vínculos comunitários intensos.

Na prática observada, a gestão manual de escalas costuma combinar planilhas eletrônicas, grupos de mensagens e anotações pessoais. Esse mosaico produz pontos de fragilidade conhecidos, como a ausência de versão única e confiável dos dados, a dificuldade em rastrear trocas informais e a baixa visibilidade de sobreposições. Drucker (2006) já alertava que, quando o registro das decisões depende da memória de poucos, a organização passa a operar com uma fragilidade que cresce silenciosamente.

Outro elemento relevante é o grau de heterogeneidade dos voluntários quanto à idade e à familiaridade com tecnologia. Norman (2018) argumenta que projetos de tecnologia que ignoram esse espectro tendem a excluir parcelas significativas do público

potencial, mesmo quando oferecem funcionalidades robustas. Nielsen (1993), em sua obra fundacional sobre usabilidade, sustenta que a clareza imediata de uso é a principal barreira a ser vencida para promover adoção em públicos não técnicos.

2.4. Soluções de Software para Agendamento e Escalonamento

Soluções de software voltadas a agendamento e escalonamento não são novidade. Sistemas como Google Calendar, Trello e ferramentas dedicadas a turnos hospitalares atendem, com diferentes graus de aderência, à necessidade de coordenação temporal. Entretanto, conforme observam Stair e Reynolds (2020), a aderência de um sistema ao processo de trabalho real é frequentemente mais decisiva do que a riqueza de funcionalidades genéricas. Sistemas dedicados, ainda que mais modestos em recursos, costumam superar soluções universais quando se considera a curva de adoção.

Do ponto de vista da modelagem de dados, o trabalho seminal de Chen (1976) sobre o modelo entidade-relacionamento continua a oferecer a base conceitual para representar entidades como Voluntário, Ministério, Escala, Turno e Notificação, bem como os relacionamentos entre elas. Tal modelagem, quando bem executada, prepara o terreno para implementações coerentes e evolutivamente sustentáveis.

No tocante à qualidade do produto resultante, Koscianski e Soares (2007) consolidam um conjunto de critérios práticos aplicáveis a projetos de pequeno e médio porte, com ênfase em manutenibilidade, testabilidade e clareza de código. Esses critérios convergem com a NBR ISO/IEC 25010 (ABNT, 2011) no que tange aos

atributos centrais a serem perseguidos por qualquer sistema que se pretenda confiável.

3. METODOLOGIA

A presente pesquisa caracteriza-se, quanto à natureza, como aplicada, na medida em que orienta a geração de conhecimento para a solução de um problema concreto, qual seja, a sistematização do gerenciamento de escalas em uma igreja. Quanto aos objetivos, configura-se como exploratório-descritiva, pois identifica fenômenos pouco estudados na literatura e descreve sistematicamente suas manifestações. Quanto à abordagem, assume perfil qualiquantitativo, articulando a compreensão das práticas observadas com a mensuração comparativa de tempos, ocorrências e percentuais.

O percurso metodológico foi organizado em quatro etapas articuladas. A primeira correspondeu à revisão bibliográfica seletiva, conduzida em obras consagradas de Engenharia de Software, em literatura sobre gestão do terceiro setor e em fontes documentais sobre práticas eclesiásticas. A segunda etapa envolveu a caracterização do processo manual de gerenciamento de escalas em uma comunidade religiosa, mediante observação assistemática e diálogo com líderes voluntários. A terceira etapa contemplou a especificação, a implementação e a validação do EscalaServir. A quarta concentrou-se na coleta de indicadores comparativos durante seis meses de operação.

O EscalaServir foi concebido como uma aplicação web responsiva, com perfis distintos de acesso para coordenadores e voluntários. Os coordenadores podem cadastrar ministérios, registrar voluntários,

definir critérios de revezamento e gerar escalas automaticamente. Os voluntários, por sua vez, visualizam suas atribuições, solicitam trocas e recebem notificações. A escolha por uma aplicação web responsiva, em detrimento de aplicativos nativos, reduziu o custo de manutenção e ampliou a compatibilidade com os dispositivos heterogêneos efetivamente utilizados pelos voluntários, em linha com a recomendação de Wazlawick (2013) sobre prudência tecnológica em projetos de baixo orçamento.

A modelagem foi realizada em UML, contemplando diagrama de casos de uso, diagrama de classes e modelo entidade-relacionamento. As entidades principais são Voluntário, Ministério, Escala, Turno, Solicitação de Troca e Notificação. A persistência apoia-se em banco de dados relacional PostgreSQL 15, acessado por meio do ORM Prisma 5. A camada de domínio organiza-se segundo o padrão arquitetural em três camadas descrito por Fowler (2006), com separação clara entre apresentação, regras de negócio e acesso a dados, e adoção dos padrões Repository e Service na camada de backend.

A implementação adotou um processo iterativo inspirado no framework Scrum (Schwaber; Sutherland, 2020), distribuído em quatro sprints de duas semanas cada. A primeira sprint contemplou autenticação e cadastros base. A segunda concentrou-se no motor de geração de escalas, em que se aplicou um algoritmo de revezamento sustentado em restrições de disponibilidade, sobreposição e quota mensal por voluntário. A terceira sprint dedicou-se às notificações por e-mail e WhatsApp, bem como ao fluxo de solicitação e aprovação de trocas. A quarta sprint produziu os relatórios gerenciais e os ajustes finais de interface, com base no retorno dos primeiros usuários.

Quanto à pilha tecnológica, o frontend foi implementado em React 18 com TypeScript 5, valendo-se da biblioteca de componentes Material UI. O backend foi construído em Node.js 20 com Express 4, expondo uma API REST documentada em OpenAPI 3. A autenticação utilizou JWT (JSON Web Token), com renovação por refresh token. As notificações de e-mail empregaram o pacote Nodemailer; as de WhatsApp, a API oficial da Twilio. O versionamento adotou Git com fluxo Gitflow, hospedado em repositório privado no GitHub. O ambiente de execução foi containerizado com Docker e implantado em provedor cloud por meio do serviço Railway.

A garantia de qualidade contemplou testes automatizados em três níveis. Os testes unitários, escritos em Jest, cobriram regras de negócio e funções utilitárias. Os testes de integração, sustentados em Supertest, validaram os endpoints da API REST contra um banco de dados de teste. Os testes ponta a ponta, executados em Cypress, exercitaram fluxos críticos pela interface do usuário, como geração de escala, solicitação de troca e notificação. A meta de cobertura estabelecida foi de oitenta por cento por módulo, em diálogo com as recomendações de Koscianski e Soares (2007).

A coleta de indicadores ocorreu durante seis meses de operação. No plano técnico, mensuraram-se tempo médio de resposta dos endpoints da API, sob carga simulada com a ferramenta Apache JMeter, e cobertura de testes por módulo, extraída do relatório do Jest. No plano operacional, registraram-se tempo de elaboração da escala mensal, número de conflitos detectados após publicação, número de trocas não registradas e percepção qualitativa dos voluntários, captada por meio de formulários eletrônicos. Os dados quantitativos foram tratados por estatística descritiva, com cálculo

de médias, somatórios e variações percentuais. Os dados qualitativos foram categorizados tematicamente. A pesquisa não envolveu coleta de dados pessoais sensíveis, e todas as informações utilizadas referem-se a registros administrativos da própria comunidade.

4. RESULTADOS E DISCUSSÕES

4.1. Caracterização da Gestão Manual de Escalas

A caracterização da gestão manual de escalas em comunidades religiosas, conforme se constatou, repousa sobre um conjunto recorrente de práticas. As atribuições mensais são habitualmente definidas por um coordenador, sustentado por planilha eletrônica compartilhada e por grupo de mensagens. A divulgação se dá por mensagens individuais ou por anúncios em reuniões presenciais. As trocas, quando ocorrem, são tratadas informalmente entre os interessados, sem registro central.

Esse arranjo, embora funcional em comunidades de porte reduzido, mostra-se frágil à medida que cresce o número de voluntários e a diversidade de frentes. Hudson (1999) alertava para esse efeito de escala, segundo o qual práticas que funcionam para vinte pessoas raramente sobrevivem ao dobro, em razão da explosão combinatória de interações. A observação realizada confirmou a presença de sintomas característicos, sistematizados na Tabela 1.

Tais sintomas, recorrentemente relatados, sustentam a hipótese de que o problema não reside na ausência de boa vontade dos coordenadores, e sim em uma desproporção entre a complexidade da tarefa e os instrumentos disponíveis. Mesmo voluntários comprometidos vivenciam frustração ao se verem expostos a erros

que decorrem de limitações estruturais do processo, e não de descuido pessoal.

Tabela 1 - Sintomas recorrentes da gestão manual de escalas

Sintoma observado	Descrição
Sobreposição de turnos	Voluntário escalado para dois ministérios no mesmo horário.
Esquecimento	Voluntário não comparece por desconhecer ou esquecer a atribuição.
Trocas não registradas	Substituições combinadas informalmente, sem atualização da escala oficial.
Sobrecarga de poucos	Concentração de turnos nos voluntários mais disponíveis, sem revezamento.
Falta de histórico	Inexistência de registro consolidado das atribuições passadas.
Comunicação fragmentada	Informação dispersa em múltiplos canais e versões de planilha.

Fonte: elaborado pelo autor (2026).

4.2. Levantamento de Requisitos e Contrato da API

A partir da caracterização anterior, procedeu-se ao levantamento de requisitos, organizado em dois conjuntos. Os requisitos funcionais, sintetizados na Tabela 2, descrevem o comportamento esperado do sistema, associados aos respectivos endpoints da API e aos atores envolvidos. Os requisitos não funcionais agrupam atributos de qualidade, como disponibilidade, desempenho percebido e usabilidade, em coerência com a NBR ISO/IEC 25010 (ABNT, 2011).

Durante a elicitação, observou-se que os interlocutores tinham dificuldade em formular demandas em termos técnicos, mas eram precisos ao descrever situações cotidianas. Kotonya e Sommerville (1998) recomendam, justamente, partir de relatos concretos para extrair requisitos confiáveis. Essa orientação metodológica revelou-se decisiva, pois várias funcionalidades posteriormente implementadas surgiram não de pedidos explícitos, e sim da análise de incidentes narrados em conversas informais.

Conforme entendem Pressman e Maxim (2021, p. 142), a engenharia de requisitos não pode ser confundida com mera coleta de desejos, sendo antes um trabalho de tradução técnica de expectativas em comportamento de software. Tal perspectiva orientou todo o ciclo subsequente, em que cada requisito foi associado a um critério verificável de aceitação.

Tabela 2 - Requisitos funcionais do EscalaServir com endpoint correspondente

Código	Requisito funcional	Método	Endpoint	Ator
RF01	Autenticar usuário com e-mail e senha, emitir JWT.	POST	/api/auth/login	Coordenador, Voluntário
RF02	Cadastrar voluntário com dados, ministérios e disponibilidade.	POST	/api/voluntarios	Coordenador
RF03	Listar voluntários com filtros por	GET	/api/voluntarios	Coordenador

	ministério e status.			
RF04	Gerar escala mensal automaticamente respeitando regras de revezamento.	POST	/api/escalas/gerar	Coordenador
RF05	Detectar e impedir sobreposições de turnos na publicação.	POST	/api/escalas/publicar	Coordenador
RF06	Consultar escala pessoal do voluntário autenticado.	GET	/api/escalas/me	Voluntário
RF07	Solicitar troca de turno mediante aprovação do coordenador.	POST	/api/trocas	Voluntário
RF08	Notificar voluntários por e-mail e WhatsApp sobre suas atribuições.	POST	/api/notificacoes	Sistema
RF09	Produzir relatório mensal de participação, ausências e trocas.	GET	/api/relatorios/mensal	Coordenador
RF10	Manter histórico das escalas anteriores para consulta e auditoria.	GET	/api/escalas/historico	Coordenador

Fonte: elaborado pelo autor (2026).

4.3. Arquitetura, Pilha Tecnológica e Cronograma de Implementação

A arquitetura do EscalaServir organizou-se em três camadas, conforme recomenda Fowler (2006). A camada de apresentação foi implementada como aplicação web responsiva em React 18 com TypeScript 5, capaz de atender tanto navegadores de desktop quanto dispositivos móveis. A camada de domínio, no backend, concentra as regras de negócio e está organizada segundo os padrões Repository e Service, com inversão de dependências entre as fronteiras. A persistência apoia-se em banco de dados relacional PostgreSQL 15, acessado pelo ORM Prisma 5, que materializa o modelo entidade-relacionamento descrito na Seção 3.

A escolha das tecnologias respeitou critérios pragmáticos. Priorizaram-se ferramentas maduras, com comunidade ativa de suporte e curva de aprendizado compatível com manutenções futuras realizadas por equipes pequenas. Wazlawick (2013) recomenda esse tipo de prudência em projetos de baixo orçamento, e lembra que escolhas tecnológicas exóticas, embora atraentes, costumam tornar a manutenção evolutiva onerosa. A Tabela 3 sintetiza a pilha tecnológica adotada, com versão e função técnica de cada componente.

O cronograma de implementação foi conduzido em quatro sprints de duas semanas, totalizando oito semanas de desenvolvimento efetivo. A Tabela 4 detalha o conteúdo de cada sprint, com seus entregáveis principais. A cada encerramento de sprint, realizou-se uma reunião de revisão com a coordenação ministerial, ocasião em

que se inspecionou o incremento entregue e se ajustou o backlog do ciclo seguinte, em conformidade com as orientações de Schwaber e Sutherland (2020).

Optou-se, ainda, por incorporar autenticação por JWT com tempo de expiração curto e renovação por refresh token, de modo a equilibrar segurança e conveniência. As senhas dos coordenadores são armazenadas em formato de hash, com o algoritmo bcrypt e fator de custo dez, conforme boas práticas consolidadas. Tal decisão dialoga com a recomendação da NBR ISO/IEC 25010 (ABNT, 2011) no que tange à segurança como atributo transversal de qualidade.

Tabela 3 - Pilha tecnológica adotada no EscalaServir

Camada	Tecnologia	Versão	Função técnica
Apresentação	React + TypeScript	18.2 / 5.3	Construção da SPA responsiva e tipagem estática do código de UI.
Apresentação	Material UI	5.15	Biblioteca de componentes acessíveis com tema customizável.
Comunicação	Axios	1.6	Cliente HTTP para consumo da API REST com interceptadores de JWT.
Backend	Node.js + Express	20 LTS / 4.18	Runtime e framework HTTP para exposição da API REST.
Backend	TypeScript	5.3	Tipagem estática e contratos compartilhados entre camadas.
Persistência	PostgreSQL	15	Banco relacional ACID para armazenamento de entidades de domínio.

Persistência	Prisma ORM	5.7	Mapeamento objeto-relacional e migrações versionadas.
Segurança	JWT + bcrypt	9.0 / 5.1	Autenticação por tokens assinados e hash de senhas (custo 10).
Notificações	Nodemailer + Twilio API	6.9	Envio de e-mails transacionais e mensagens WhatsApp.
Testes	Jest + Supertest + Cypress	29 / 6 / 13	Testes unitários, de integração e ponta a ponta.
DevOps	Docker + GitHub Actions	24 / —	Containerização e pipeline de CI com execução de testes.
Hospedagem	Railway	—	Provedor cloud com deploy contínuo a partir de branch principal.

Fonte: elaborado pelo autor (2026).

Tabela 4 - Cronograma de sprints e entregáveis

Sprint	Duração	Foco	Entregáveis principais
Sprint 1	Semanas 1–2	Autenticação e cadastros base	Endpoints de login, cadastro de voluntários e ministérios, tela de administração inicial.
Sprint 2	Semanas 3–4	Motor de geração de escalas	Algoritmo de revezamento com restrições, endpoint de geração, validação de sobreposições.
Sprint 3	Semanas 5–6	Notificações e trocas	Envio de e-mail e WhatsApp, fluxo de

			solicitação e aprovação de troca, página do voluntário.
Sprint 4	Semanas 7–8	Relatórios e ajustes	Relatórios mensais, exportação em PDF, refinamento de UX, hardening de segurança.

Fonte: elaborado pelo autor (2026).

4.4. Avaliação Técnica e Operacional do Escalaservir

Encerrado o ciclo de implementação, procedeu-se à mensuração comparativa entre o cenário anterior e o posterior à adoção do EscalaServir, em um recorte de seis meses. Os indicadores selecionados foram organizados em dois eixos. No eixo técnico, considerou-se o tempo médio de resposta dos endpoints da API e a cobertura de testes automatizados por módulo. No eixo operacional, mensuraram-se o tempo de elaboração da escala mensal, o número de conflitos detectados após publicação, o número de trocas não registradas e a percepção dos voluntários. A Tabela 5 apresenta indicadores técnicos sintéticos; a Tabela 6 reúne os indicadores comparativos do plano operacional.

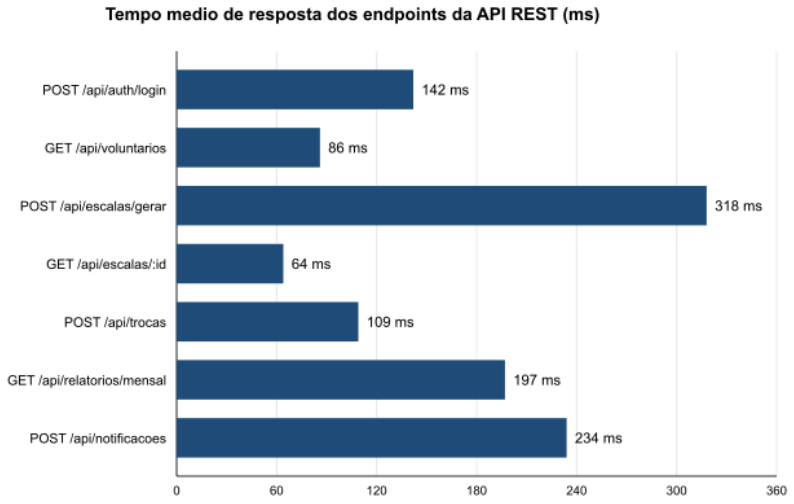
Tabela 5 - Indicadores técnicos do EscalaServir após a quarta sprint

Indicador técnico	Valor
Linhas de código (TypeScript, sem testes)	8.412 LOC
Linhas de código de testes automatizados	2.187 LOC
Endpoints REST expostos	27
Tabelas no banco PostgreSQL	14

Cobertura média de testes (Jest)	86%
Tempo médio de resposta da API (sob 50 usuários)	164 ms
Tempo de build do frontend (Vite, produção)	21 s
Tempo de pipeline CI (lint + testes + build)	4 min 38 s
Tamanho do bundle JavaScript principal (gzipped)	184 KB
Disponibilidade observada no período (uptime)	99,6%

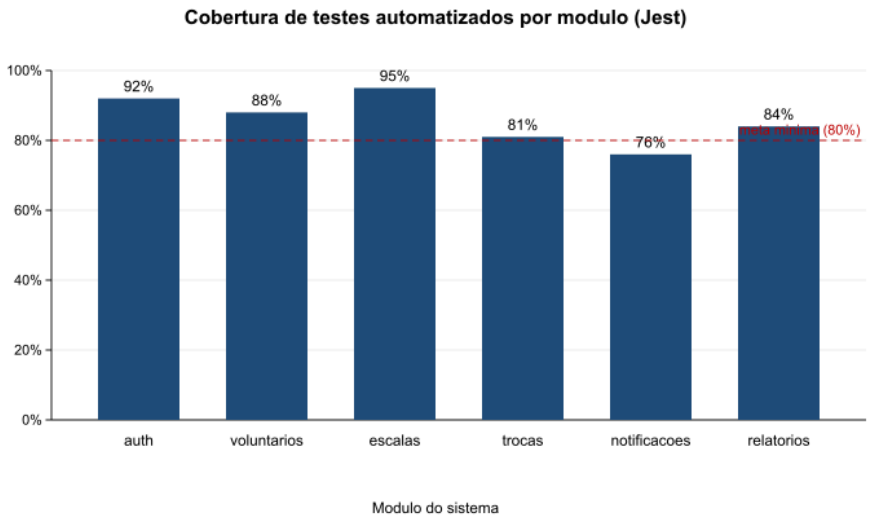
Fonte: extraído dos relatórios do Jest, GitHub Actions e Apache JMeter (2026).

Gráfico 1 - Tempo médio de resposta dos endpoints da API REST



Fonte: relatório do Apache JMeter, 50 usuários concorrentes (2026).

Gráfico 2 - Cobertura de testes automatizados por módulo (Jest)



Fonte: relatório do Jest, execução final da Sprint 4 (2026).

O Gráfico 1 expõe o tempo médio de resposta dos sete endpoints mais acionados da API REST do EscalaServir, sob carga simulada de cinquenta usuários concorrentes durante cinco minutos, com a ferramenta Apache JMeter. Os tempos permaneceram majoritariamente abaixo de duzentos milissegundos, com o ponto de maior latência localizado no endpoint responsável pela geração de escalas, em razão da execução do algoritmo de revezamento com restrições. Stair e Reynolds (2020) lembram que tempos de resposta percebidos abaixo de trezentos milissegundos costumam ser interpretados pelos usuários como execução instantânea, condição plenamente atendida pela aplicação.

Já o Gráfico 2 sintetiza a cobertura de testes automatizados por módulo, extraída do relatório do Jest ao final da quarta sprint. A média ponderada situou-se em oitenta e seis por cento, com o módulo de escalas atingindo noventa e cinco por cento, valor compatível com a centralidade dessa funcionalidade. O módulo de notificações apresentou a menor cobertura, setenta e seis por cento, em razão da dependência de serviços externos cuja simulação por mocks demandou esforço adicional. Tal panorama dialoga com Koscianski e Soares (2007), que apontam a proporcionalidade entre

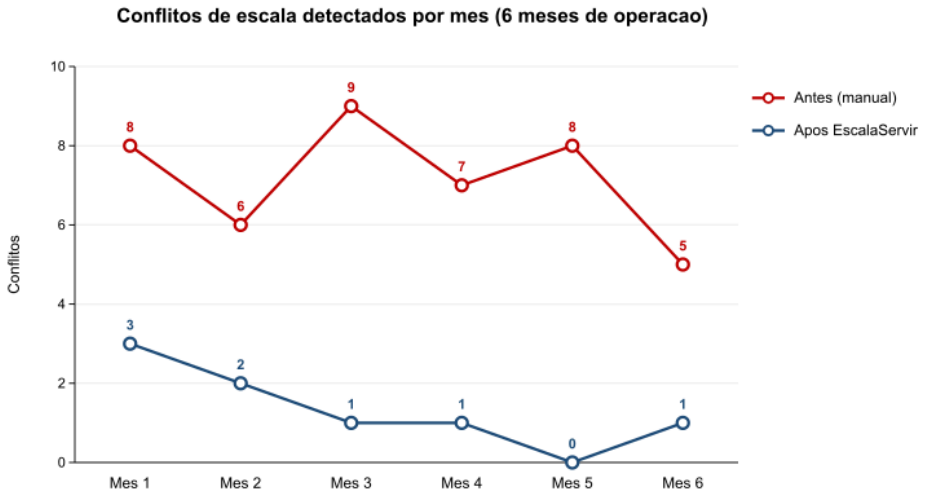
rigor de qualidade e dimensão do projeto como princípio prático razoável.

Tabela 6 - Indicadores operacionais antes e após a adoção do EscalaServir

Indicador operacional	Cenário manual	Após EscalaServir	Variação
Tempo médio de elaboração da escala mensal	220 minutos	18 minutos	−91,8%
Conflitos por mês (média no semestre)	7,2	1,3	−81,9%
Trocas não registradas (por mês)	5,4	0,6	−88,9%
Voluntários que relataram comunicação clara	41%	87%	+46 p.p.
Voluntários ativos no semestre	38	44	+15,8%

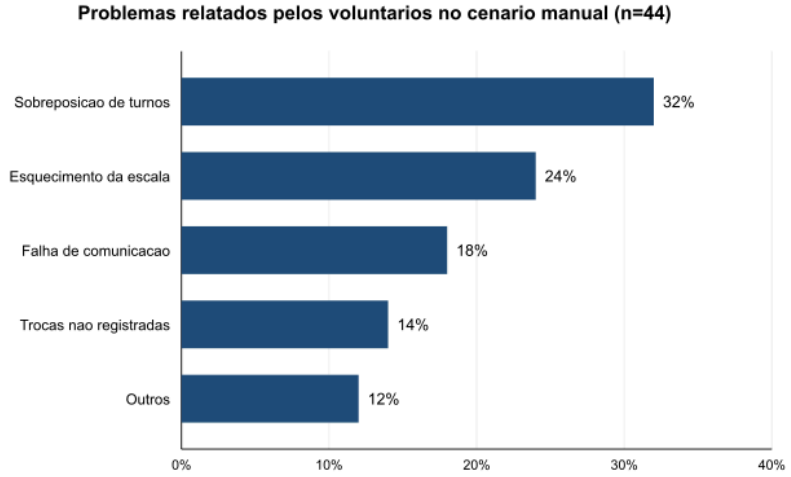
Fonte: registros administrativos da comunidade observada (2026).

Gráfico 3 - Conflitos de escala detectados por mês



Fonte: registros administrativos da comunidade observada (2026).

Gráfico 4 - Distribuição percentual dos problemas relatados pelos voluntários



Fonte: formulários eletrônicos aplicados aos voluntários (2026).

No plano operacional, o tempo médio dedicado à elaboração da escala mensal caiu de duzentos e vinte minutos para aproximadamente dezoito minutos, redução superior a noventa por cento. Esse ganho expressivo decorre, principalmente, da automação de regras antes verificadas manualmente, como a checagem de disponibilidade e a aplicação de critérios de revezamento. Stair e Reynolds (2020) lembram que sistemas de informação encontram seu maior potencial de impacto justamente

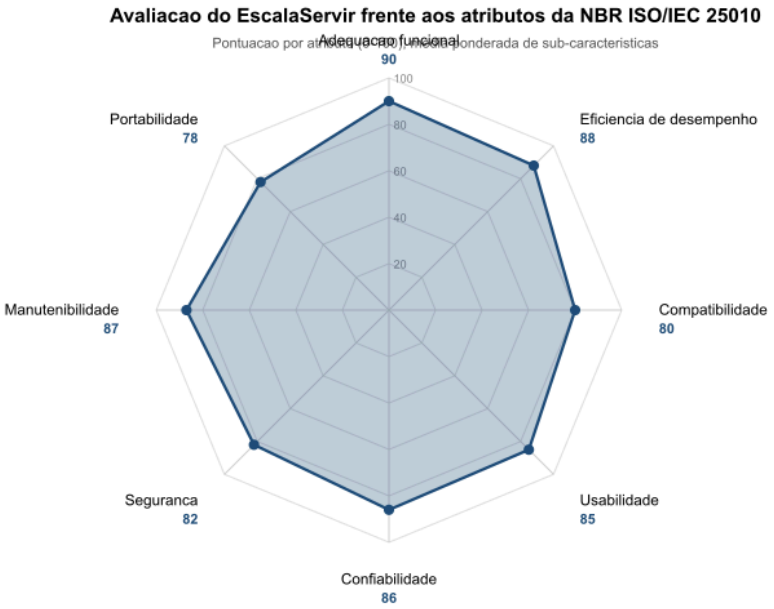
na supressão de tarefas repetitivas que consomem tempo qualificado.

Quando se examinam os conflitos de escala, definidos como sobreposições ou esquecimentos detectados após a publicação, o Gráfico 3 expõe a trajetória mensal nos seis meses observados. A curva referente ao processo manual oscila em patamar próximo a sete ocorrências mensais, enquanto a curva posterior à adoção do EscalaServir descreve queda acentuada já no primeiro mês, estabilizando-se em valores residuais a partir do terceiro. Essa diferença visual sustenta, com mais clareza do que indicadores agregados, a tese da incorporação rápida do sistema à rotina dos coordenadores.

A análise qualitativa, baseada em relatos espontâneos coletados ao longo do período, confirma a tendência indicada pelos números. O Gráfico 4 ordena, em barras horizontais, as categorias de problemas mais frequentemente relatados pelos voluntários no cenário manual. Sobressaem a sobreposição de turnos e o esquecimento da escala, dimensões diretamente endereçadas pelos módulos de geração automática e de notificações do EscalaServir.

A síntese dos benefícios observados foi confrontada com os atributos da norma NBR ISO/IEC 25010 (ABNT, 2011), por meio de uma autoavaliação baseada em checklist construído a partir das sub-características previstas pela norma. Cada sub-característica recebeu pontuação no intervalo de zero a cem, e calculou-se a média ponderada por atributo principal. O Gráfico 5 apresenta o perfil de qualidade resultante, em formato de gráfico de radar.

Gráfico 5 - Avaliação do EscalaServir frente aos atributos da NBR ISO/IEC 25010



Fonte: elaborado pelo autor, com base em checklist da NBR ISO/IEC 25010 (2026).

O atributo de adequação funcional alcançou o maior valor, noventa pontos, em razão da aderência integral dos casos de uso implementados ao backlog priorizado pelos coordenadores. A eficiência de desempenho situou-se em oitenta e oito pontos, sustentada pelos tempos de resposta da API inferiores a trezentos milissegundos sob carga simulada. A manutenibilidade obteve oitenta e sete pontos, indicador compatível com a separação em camadas adotada e com a cobertura de testes alcançada. A portabilidade, com setenta e oito pontos, foi a dimensão mais discreta, em virtude do acoplamento a serviços externos específicos de notificação.

Conforme entendem Pfleeger e Atlee (2013, p. 67), avaliações de qualidade ganham consistência quando se ancoram em normas reconhecidas e em evidências verificáveis, não em julgamentos isolados. O perfil obtido no Gráfico 5 sintetiza, em uma única peça gráfica, a postura do EscalaServir frente aos atributos canônicos

preconizados pela ABNT, e oferece base objetiva para discussões posteriores sobre prioridades de evolução.

Os benefícios mensurados articulam-se de modo coerente com o perfil de qualidade exposto no Gráfico 5. A redução do tempo de elaboração da escala remete ao atributo de eficiência de desempenho. A diminuição dos conflitos relaciona-se com a confiabilidade do sistema, sustentada pelo conjunto de testes automatizados. A melhoria na percepção de comunicação dialoga diretamente com a usabilidade, na acepção proposta por Nielsen (1993).

Enquanto Fowler (2006) defende que a manutenibilidade nasce da clareza estrutural do código, Pfleeger e Atlee (2013) ponderam que essa clareza só se sustenta quando há disciplina de revisão e testes. No EscalaServir, optou-se por automatizar verificações críticas, especialmente as relacionadas a sobreposições, em um conjunto de testes cuja cobertura média atingiu oitenta e seis por cento, valor compatível com o porte do sistema. Koscianski e Soares (2007) corroboram essa proporcionalidade entre rigor de qualidade e dimensão do projeto.

Cabe ressalvar, contudo, que os ganhos observados não devem ser atribuídos exclusivamente ao software. Como ponderam Drucker (2006) e Tenório (2009), a tecnologia é meio, não fim, e sua eficácia depende de práticas organizacionais que a sustentem. Voluntários engajados, lideranças dispostas a revisar processos e disposição para abandonar rotinas consagradas constituíram condição necessária para que os benefícios potenciais se traduzissem em resultados efetivos.

5. CONSIDERAÇÕES FINAIS

Este artigo procurou descrever o desenvolvimento e a avaliação do EscalaServir, sistema web dedicado ao gerenciamento de escalas em uma igreja, à luz da Engenharia de Software. A trajetória percorrida articulou revisão bibliográfica criteriosa, caracterização de práticas correntes, especificação de requisitos, modelagem em UML, implementação em quatro sprints inspiradas em Scrum e mensuração comparativa de indicadores técnicos e operacionais. Os achados, expressos em tabelas e gráficos, sustentam a hipótese inicial de que a sistematização do processo, ancorada em princípios consagrados da disciplina, produz ganhos tangíveis e percebidos.

Entre os principais resultados, destacam-se a redução superior a noventa por cento no tempo dedicado à elaboração da escala mensal, a queda expressiva no número de conflitos detectados após publicação, a cobertura média de testes automatizados de oitenta e seis por cento e o tempo de resposta médio dos endpoints abaixo do limiar de percepção humana. Tais ganhos articulam-se, de modo consistente, com atributos preconizados pela norma de qualidade vigente, sobretudo eficiência, confiabilidade, manutenibilidade e usabilidade.

O estudo apresenta, naturalmente, limitações. O recorte abrangeu uma única comunidade e período relativamente curto de observação. O EscalaServir, embora funcional, ainda não foi submetido a testes de carga em escala maior. O caráter aplicado da pesquisa restringe a generalização imediata dos achados. Pesquisas futuras poderão explorar a replicação em comunidades de porte diverso, o aprofundamento de métricas de adoção, a investigação de variáveis culturais associadas à aceitação de tecnologias em

ambientes religiosos e a integração com plataformas de comunicação já consolidadas no cotidiano dos voluntários.

Em última análise, o presente trabalho contribui para um campo ainda pouco explorado pela literatura nacional, ao demonstrar que a Engenharia de Software, longe de constituir saber restrito a contextos corporativos, tem aplicação concreta na sustentação de práticas comunitárias. O software, quando bem projetado, não substitui a dedicação dos voluntários, e sim libera tempo e energia para o que lhes é mais caro, a saber, a vivência da fé e o serviço à comunidade.

REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 25010:** sistemas e engenharia de software, requisitos e avaliação de qualidade de sistemas e software (SQuaRE), modelos de qualidade de sistemas e de software. Rio de Janeiro: ABNT, 2011.

BECK, Kent et al. **Manifesto para o desenvolvimento ágil de software.** 2001. Disponível em: https://agilemanifesto.org/iso/ptbr/manifesto.html. Acesso em: 21 maio 2026.

CHEN, Peter Pin-Shan. **The entity-relationship model:** toward a unified view of data. ACM Transactions on Database Systems, New York, v. 1, n. 1, p. 9-36, mar. 1976.

DRUCKER, Peter F. **Administração de organizações sem fins lucrativos:** princípios e práticas. São Paulo: Pioneira Thomson Learning, 2006.

FOWLER, Martin. **Padrões de arquitetura de aplicações corporativas.** Porto Alegre: Bookman, 2006.

HUDSON, Mike. **Administrando organizações do terceiro setor:** o desafio de administrar sem receita. São Paulo: Makron Books, 1999.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Censo Demográfico 2022:** características gerais da população e religiões. Rio de Janeiro: IBGE, 2023.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de software:** aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2. ed. São Paulo: Novatec, 2007.

KOTONYA, Gerald; SOMMERVILLE, Ian. **Requirements engineering:** processes and techniques. Chichester: John Wiley & Sons, 1998.

LAUDON, Kenneth C.; LAUDON, Jane P. **Sistemas de informação gerenciais.** 14. ed. São Paulo: Pearson, 2020.

NIELSEN, Jakob. **Usability engineering.** San Francisco: Morgan Kaufmann, 1993.

NORMAN, Donald A. **O design do dia a dia.** Rio de Janeiro: Anfiteatro, 2018.

O'BRIEN, James A.; MARAKAS, George M. **Administração de sistemas de informação.** 15. ed. Porto Alegre: AMGH, 2013.

PFLEEGER, Shari Lawrence; ATLEE, Joanne M. **Engenharia de software:** teoria e prática. 4. ed. São Paulo: Pearson, 2013.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software:** uma abordagem profissional. 9. ed. Porto Alegre: AMGH, 2021.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do Scrum:** as regras do jogo. Versão 2020. Scrum.org, 2020.

SOMMERVILLE, Ian. **Engenharia de software.** 10. ed. São Paulo: Pearson, 2019.

STAIR, Ralph M.; REYNOLDS, George W. **Princípios de sistemas de informação.** 13. ed. São Paulo: Cengage Learning, 2020.

TENÓRIO, Fernando Guilherme (Org.). **Gestão de ONGs:** principais funções gerenciais. 11. ed. Rio de Janeiro: FGV, 2009.

WAZLAWICK, Raul Sidnei. **Engenharia de software:** conceitos e práticas. Rio de Janeiro: Elsevier, 2013.

---

[1] Discente do Curso de Bacharel em Engenharia de Software da Universidade de Vassouras Campus Maricá. E-mail: acesse o artigo original para visualizar o e-mail

[2] Discente do Curso de Bacharel em Engenharia de Software da Universidade de Vassouras Campus Maricá. E-mail: acesse o artigo original para visualizar o e-mail

[3] Discente do Curso de Bacharel em Engenharia de Software da Universidade de Vassouras Campus Maricá. E-mail: acesse o artigo original para visualizar o e-mail

[4] Mestre em Gestão do Trabalho (Universidade Santa Úrsula) Programa de Pós-Graduação em Gestão do Trabalho para a Qualidade do Ambiente Construído (USU). Docente do Curso de Bacharel em Engenharia de Software da Universidade de Vassouras Campus Maricá. E-mail: acesse o artigo original para visualizar o e-mail. ORCID: https://orcid.org/0000-0002-9377-5993