A Checklist for Energy-Efficiency Requirements in Web Applications

Ana Carolina P. Rodrigues
Universidade Federal do Pampa
Alegrete, RS, Brazil
anapoltronieri.aluno@unipampa.edu.br

Gláucia D. Tavares
Universidade Federal do Pampa
Alegrete, RS, Brazil
glauciatavares.aluno@unipampa.edu.br

Juliano B. Sobroza
Universidade Federal do Pampa
Alegrete, RS, Brazil
julianosobroza.aluno@unipampa.edu.br

Kelvin M. Camargo
Universidade Federal do Pampa
Alegrete, RS, Brazil
kelvincamargo.aluno@unipampa.edu.br

Gustavo O. Almeida
Universidade Federal do Pampa
Alegrete, RS, Brazil
gustavodoa.aluno@unipampa.edu.br

Álvaro D. S. Mota
Universidade Federal do Pampa
Alegrete, RS, Brazil
alvaromota.aluno@unipampa.edu.br

Anielle Andrade
Universidade Federal do Pampa
Alegrete, RS, Brazil
anielleandrade@unipampa.edu.br

Maicon Bernardino
Universidade Federal do Pampa
Alegrete, RS, Brazil
bernardino@acm.org

Resumo

Sistemas de software consomem recursos computacionais substanciais, e seus custos ambientais têm recebido atenção crescente na literatura. A Engenharia de Requisitos tem sido investigada como um ponto de intervenção para abordar a eficiência energética ainda nas fases iniciais do desenvolvimento, porém ainda há uma lacuna quanto a mecanismos práticos que apoiem essa elicitação especificamente no contexto de aplicações web. Este trabalho propõe um *checklist* para apoiar a elicitação e a organização de requisitos de eficiência energética em aplicações web, estruturado em quatro camadas: *frontend*, *backend*, banco de dados e infraestrutura. O *checklist* é composto por 16 itens verificáveis, derivados de padrões recorrentes na literatura sobre software sustentável e requisitos verdes. A validade de conteúdo foi avaliada por meio de um painel de especialistas, utilizando o método *Content Validity Ratio* (CVR). O *checklist* oferece uma referência estruturada para que equipes de desenvolvimento considerem restrições de eficiência energética durante a especificação de requisitos, com potenciais implicações para a redução do impacto ambiental de aplicações web.

Palavras-chave

Engenharia de Requisitos, Engenharia de Software, Software Verde, Eficiência Energética, Aplicações Web, Checklist

1 Introdução

A crescente expansão do uso de sistemas de software em diversos setores da sociedade tem ampliado significativamente sua influência não apenas em processos organizacionais, mas também no meio ambiente. Infraestruturas digitais, como data centers e aplicações em nuvem, demandam elevado consumo de energia e recursos computacionais, contribuindo diretamente para impactos ambientais [23]. Nessa direção, emerge a necessidade do desenvolvimento de sistemas mais eficientes, impulsionando o conceito de software sustentável e sistemas verdes [9].

A literatura evidencia que o impacto do software pode ser analisado em diferentes níveis, considerando efeitos diretos relacionados ao consumo de recursos computacionais, bem como impactos indiretos e de longo prazo decorrentes do seu uso contínuo [22]. Esses aspectos reforçam a necessidade de considerar a sustentabilidade desde as fases iniciais do desenvolvimento, e não apenas em etapas posteriores.

Apesar dos avanços na literatura a respeito da necessidade de integrar aspectos de sustentabilidade ao longo de todo o ciclo de vida do software [18], ainda há desafios significativos na adoção prática desses conceitos na indústria [1], sobretudo pela ausência de mecanismos práticos que apoiem equipes de desenvolvimento na elicitação e organização de requisitos de eficiência energética especificamente para aplicações web.

Diante dessa lacuna, este trabalho tem como objetivo propor um checklist de apoio à elicitação, organização e verificação de requisitos verdes voltados à eficiência energética em aplicações web, estruturado segundo as camadas frontend, backend, banco de dados e infraestrutura. Espera-se que essa proposta contribua para aproximar os princípios de software verde da prática de desenvolvimento, oferecendo às equipes um instrumento estruturado para considerar restrições de eficiência energética desde as fases iniciais da especificação de requisitos.

2 Referencial Teórico e Trabalhos Relacionados

2.1 Sistemas Verdes e Software Sustentável

O conceito de Sistemas Verdes e de Software Sustentável evoluiu ao longo das últimas décadas. Inicialmente, a literatura buscou estabelecer uma perspectiva mais definicional e normativa, com ênfase na redução de impactos negativos e na consideração do ciclo de vida do software. Essa orientação pode ser observada em [10], que define software sustentável como aquele cujos impactos negativos diretos e indiretos sobre a economia, a sociedade, os seres humanos e o meio ambiente, decorrentes do desenvolvimento, da implantação e do uso do *software*, são mínimos ou ainda produzem efeitos positivos para o desenvolvimento sustentável.

Após essas definições iniciais, a área passou a concentrar esforços em como medir, avaliar e aplicar a sustentabilidade em software,

voltando-se ao mapeamento de medidas e formas de avaliação [16]. Dentre os resultados está o modelo GREENSOFT, que organiza características do software para apoiar desenvolvedores a projetá-lo de forma mais sustentável [17], e mais recentemente observa-se um movimento em direção a um entendimento comum sobre *Sustainable Software Development* (Desenvolvimento de Software Sustentável, SSD), buscando reduzir a fragmentação terminológica [13] o que demonstra a maturidade crescente da área, indicando uma transição de abordagens teóricas para aplicações mais práticas.

2.2 Engenharia de Requisitos e Requisitos Verdes

A Engenharia de Requisitos (RE) é uma área da engenharia de *software* na qual são estabelecidos os limites, as funções e os objetivos do sistema a partir das necessidades e preocupações dos stakeholders e dos aspectos sociais, econômicos e físicos [5]. A RE possui quatro etapas: elicitação, cujo foco é a identificação dos requisitos por meio das necessidades dos *stakeholders*; modelagem e análise, voltada à estruturação e compreensão de requisitos; asseguração, focada na verificação de qualidade; gerenciamento e evolução de requisitos que os adapta diante das mudanças que ocorrem durante o desenvolvimento [5].

Nesse contexto, RE torna-se especialmente relevante para a sustentabilidade, pois permite incorporar preocupações ambientais ainda nas fases iniciais do desenvolvimento, influenciando diretamente decisões arquiteturais e de implementação.

Para a criação de Sistemas Verdes, pode-se utilizar um ramo específico da Engenharia de Requisitos, denominado *Green Requirements*, ou Requisitos Verdes. Esses requisitos delimitam objetivos voltados tanto aos impactos ambientais diretos do sistema, como consumo de energia e recursos computacionais, quanto aos impactos indiretos influenciados por seu uso. A elicitação de tais requisitos durante o planejamento do sistema amplia a compreensão do que é um software sustentável, assim contribuindo, desde o início, para a consideração de aspectos ambientais relevantes à concretização de um software sustentável [18]. Apesar de não estar muito presente na indústria, a literatura em RE voltada a sustentabilidade já apresenta, há mais de uma década, abordagens para esse tema [1].

É importante diferenciar os dois termos utilizados ao longo deste artigo. Software sustentável refere-se a um conceito mais amplo, que abrange impactos econômicos, sociais e ambientais do software ao longo de seu ciclo de vida [10]. Requisitos verdes, por sua vez, constituem um subconjunto mais específico desse conceito, voltado exclusivamente aos impactos ambientais diretos e indiretos do software, como consumo de energia e de recursos computacionais [17, 18]. Este trabalho concentra-se especificamente em requisitos verdes voltados à eficiência energética, sem tratar de forma abrangente as dimensões econômica e social da sustentabilidade de software.

Neste artigo, adota-se a nomenclatura elicitação, organização e verificação para descrever o processo de aplicação do checklist proposto, mapeando-se, respectivamente, às fases de elicitação, modelagem e análise e asseguração da RE tradicional [5]. O termo "organização"é empregado aqui para enfatizar o agrupamento dos itens por camadas da aplicação web, e não como uma fase adicional à literatura clássica de RE.

2.3 Padrões de Software Verde para Aplicações Web

A partir da definição de requisitos verdes, torna-se importante compreender como esses requisitos podem ser aplicados na prática. Nesse contexto, destacam-se os padrões de software verde (green software patterns), termo utilizado neste trabalho para se referir a um conjunto de boas práticas recorrentes voltadas ao desenvolvimento de software com menor consumo de recursos e menor impacto ambiental [2, 5, 18]. Para efeito de consistência terminológica, este artigo adota o termo "padrão"para se referir a essas boas práticas ao longo de todo o texto, evitando variações como "estratégia"ou "prática"isoladamente.

Para melhor organização e facilitar a compreensão e aplicação, esses padrões são classificados de acordo com as camadas de uma aplicação web:

Frontend, envolve práticas voltadas à interface com o usuário, como a redução do tamanho de arquivos, compressão de imagens, minimização de *scripts* e diminuição do número de requisições HTTP, contribuindo para menor consumo de energia no lado do cliente [25]; **Backend**, inclui padrões relacionados ao processamento de dados, como a otimização de algoritmos, eliminação de operações redundantes e uso de mecanismos de cache, reduzindo o consumo de CPU e aumentando a eficiência do sistema [20, 25]; **Banco de Dados**, abrange práticas como otimização de consultas, indexação [20, 25] e remoção de dados obsoletos, contribuindo para menor uso de armazenamento e processamento [24]; **Infraestrutura**, refere-se ao uso eficiente de recursos computacionais, como escalabilidade automática, virtualização e uso de serviços em nuvem, possibilitando melhor aproveitamento energético dos sistemas [11, 17, 25].

A opção por incluir banco de dados e infraestrutura, além de frontend e backend, decorre do fato de que o consumo energético de aplicações web não se restringe ao código da aplicação: práticas de persistência de dados (e.g., indexação, remoção de dados obsoletos) e decisões de provisionamento de recursos computacionais (e.g., escalabilidade, hospedagem) têm impacto direto e frequentemente reportado na literatura sobre eficiência energética [11, 20, 24, 25]. A divisão em quatro camadas busca, portanto, refletir uma visão de arquitetura de alto nível que cobre os principais pontos de decisão de uma aplicação web onde requisitos de eficiência energética podem ser aplicados.

A adoção desses padrões traz benefícios como redução do consumo energético e de custos operacionais, mas pode envolver trocas (trade-offs), como maior complexidade ou esforço de desenvolvimento [2].

2.4 Trabalhos Relacionados

Diante da necessidade de incorporar a sustentabilidade nas fases iniciais do desenvolvimento de software, o trabalho de [3] amplia a visão do Manifesto de Karlskrona [4] por meio do *Sustainability Awareness Framework* (SusAF). O estudo propõe que a sustentabilidade seja tratada como um requisito de primeira classe. A aplicação do *framework* permite identificar impactos de longo prazo (terceira ordem) ainda na fase de elicitação, possibilitando que partes interessadas compreendam melhor as consequências das decisões de projeto nas dimensões social e ambiental.

Para avaliar a sustentabilidade ao longo do ciclo de vida do software, [17] propõe o modelo GREENSOFT, já mencionado na fundamentação teórica, destacando que a adoção de requisitos técnicos e econômicos contribui para reduzir o impacto ambiental e promover uso mais eficiente de recursos, alinhando-se aos princípios de SSD.

Complementarmente, [1] realizaram um mapeamento sistemático com 55 publicações, identificando 29 abordagens de ER para sustentabilidade desde 2000, com aumento recente de publicações mas persistente falta de aplicação na indústria, reforçando a necessidade de mecanismos mais práticos.

Nesse contexto, [22] propõem o catálogo preliminar NFR4SUSTAIN, abordando dimensões econômica, social e técnica, para apoiar desenvolvedores na incorporação de NFRs de sustentabilidade e a seleção de sistemas por partes interessadas.

Complementarmente, [8] reúnem métodos e práticas voltados à construção eficiente de programas, abordando mecanismos de análise de eficiência energética do código.

No contexto de desenvolvimento web, [21] apresentam o *Green Web Meter*, sistema de monitoramento de sustentabilidade digital em aplicações web baseado em KPIs (*Key Performance Indicators*, e.g., consumo de recursos, emissões), reforçando a relevância de requisitos embasados em métricas para eficiência energética e redução de impacto ambiental.

Além disso, [25] reúnem práticas sustentáveis para as etapas de design, desenvolvimento e implantação web, aproximando a discussão de sustentabilidade de práticas concretas, embora sem organizá-las como requisitos ou *checklist* de apoio à ER.

De forma complementar, [12] discutem a sustentabilidade de *websites* sob a ótica de decisões de design, conteúdo, linguagem, organização e formatação, ampliando os requisitos de sustentabilidade para além da infraestrutura e do código.

Por fim, cabe mencionar que o objetivo do estudo de [19] é evidenciar o uso de *checklists* como suporte estruturado para análise e verificação de software. Embora o trabalho não trate diretamente de sustentabilidade ou aplicações web, ele reforça a utilidade desse tipo de abordagem. Nesse sentido, serve como base para a proposta deste artigo, que utiliza um *checklist* como instrumento para elicitar, organizar e aplicar práticas voltadas à eficiência energética no desenvolvimento web.

Em uma perspectiva mais ampla, a literatura pode ser organizada em três linhas: frameworks conceituais (e.g., SusAF, GREENSOFT); estudos analíticos, como revisões e mapeamentos, que identificam tendências e lacunas; e propostas aplicadas, como catálogos e práticas de desenvolvimento sustentável.

Apesar dos avanços, ainda há uma lacuna na integração dessas abordagens, especialmente quanto à falta de mecanismos práticos para elicitação e organização de requisitos de sustentabilidade em aplicações web. Nesse sentido, este artigo busca contribuir com uma proposta de *checklist* orientado à eficiência energética e à redução de impacto ambiental nesse contexto.

A Tabela 1 sintetiza os principais estudos discutidos nesta seção, destacando sua cobertura em relação à ER, ao contexto de aplicações web e ao uso de *checklist* ou artefato prático. A Tabela 1 evidencia que os estudos existentes cobrem de forma fragmentada os elementos centrais deste trabalho. Parte deles concentra-se em ER, parte em aplicações web e parte em artefatos de apoio, como *checklists*. Entretanto, não foi identificado um estudo que integre explicitamente esses três aspectos na forma de um *checklist* voltado à elicitação, organização e verificação de requisitos de sustentabilidade para aplicações web.

Tabela 1: Matriz sintética dos trabalhos relacionados

Estudo	ER	Web	Artefato
Becker *et al.* (2019) [3]	Sim	Não	Não
Mancebo *et al.* (2021) [15]	Sim	Não	Não
Naumann *et al.* (2011) [17]	Parcial	Não	Não
Bambazek *et al.* (2023) [1]	Sim	Não	Não
Santos *et al.* (2024) [22]	Sim	Não	Parcial
Souza *et al.* (2024) [8]	Parcial	Não	Não
Sala *et al.* (2024) [21]	Não	Sim	Parcial
González-Sordé (2025) [12]	Não	Sim	Não
Wimalasena e Arambepola (2025) [25]	Não	Sim	Não
Poon *et al.* (2011) [19]	Não	Não	Sim

3 Metodologia da Pesquisa

Este artigo caracteriza-se como uma pesquisa de natureza aplicada, com abordagem qualitativa e caráter exploratório [26], tendo como objetivo propor um checklist de apoio à elicitação e organização de requisitos de sustentabilidade em aplicações web. A escolha por uma abordagem qualitativa justifica-se pelo objetivo de construir e refinar um artefato de pesquisa e não de testar hipóteses estatísticas, alinhando-se a estratégias de métodos mistos para a etapa de avaliação com especialistas, que combina dados quantitativos (CVR) e qualitativos (comentários abertos) [7]. A condução da pesquisa foi organizada em duas etapas: a construção do checklist e sua avaliação por um painel de especialistas.

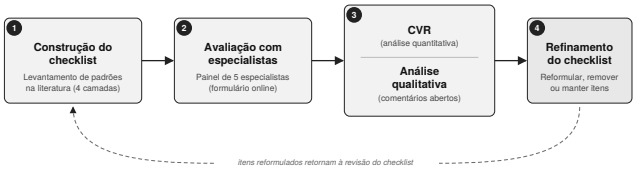

Figura 1: Fluxo metodológico da pesquisa

O processo de construção do *checklist*, ilustrado na Figura 1, seguiu três etapas: identificação de práticas sustentáveis na literatura, organização por camadas da aplicação web e transformação em itens verificáveis.

A construção da proposta foi baseada em um levantamento e análise da literatura relacionada à ER para sustentabilidade, sistemas verdes e práticas de eficiência energética em software [1, 17, 18]. Foram considerados estudos que abordam tanto aspectos conceituais, como *frameworks* e modelos, quanto trabalhos voltados a práticas concretas de desenvolvimento sustentável. Além disso, analisamos artigos que utilizaram ou exemplificaram práticas de como criar um *checklist* [19].

A partir dessa análise, foram identificados padrões recorrentes relacionados à redução do consumo de recursos computacionais e à melhoria da eficiência energética em aplicações web. Esses padrões, apresentados na subseção 2.3, foram extraídos, organizados e sintetizados de acordo com as principais camadas de uma aplicação

web: frontend, *backend*, banco de dados e infraestrutura. Com base nesses padrões, foi elaborado um *checklist* estruturado e verificável com o objetivo de apoiar engenheiros de software na identificação, análise e aplicação de práticas sustentáveis no desenvolvimento de sistemas web.

A partir dessa análise, destacam-se as práticas sustentáveis que orientaram a construção do *checklist*, tais como: (i) redução do tamanho e da quantidade de recursos no frontend [25]; (ii) minimização de requisições entre cliente e servidor [25]; (iii) otimização de algoritmos e evitar processamento desnecessário [20]; (iv) utilização de consultas eficientes no banco de dados [20]; (v) remoção frequente de dados desatualizados [24]; e (vi) uso de escalabilidade automática na infraestrutura [11]. Com base nessas práticas, cada item foi transformado em uma ou mais perguntas verificáveis, compondo o *checklist* apresentado na seção resultados.

3.1 Coleta e Análise de Dados

A segunda etapa da pesquisa consistiu na avaliação do checklist por um painel de especialistas, com o objetivo de verificar a validade de conteúdo dos itens propostos. A avaliação foi realizada por meio de um formulário online, respondido por cinco especialistas provenientes da academia e da indústria. Os participantes possuíam experiência relacionada à Engenharia de Software, Engenharia de Requisitos, desenvolvimento web ou sustentabilidade em software, contemplando perfis como pesquisadores, professores e profissionais atuantes no desenvolvimento de software.

O instrumento de coleta foi estruturado em três partes: caracterização do perfil dos especialistas, avaliação dos 16 itens do checklist e comentários abertos. Para cada item, os especialistas classificaram sua essencialidade em uma escala de três níveis: essencial, útil mas não essencial ou desnecessário. Além disso, foi disponibilizado um campo opcional para comentários e sugestões de reformulação.

Para a análise quantitativa, foi utilizado o Content Validity Ratio (CVR), proposto por Lawshe [14], calculado pela fórmula $CVR = (N_e - N/2)/(N/2)$, em que N_e representa o número de especialistas que classificaram o item como essencial e N o total de participantes. Considerando os cinco respondentes, os valores de CVR foram calculados individualmente para cada item do checklist. Itens com CVR abaixo do limiar adotado foram submetidos à revisão com base nos comentários qualitativos dos especialistas.

As respostas abertas foram analisadas qualitativamente por meio do agrupamento de comentários por temas recorrentes, como ambiguidade na redação dos itens, ausência de práticas relevantes e sugestões de reformulação. A partir dessa análise, o grupo discutiu colaborativamente os itens avaliados, decidindo entre reformular, remover ou manter cada item com justificativa. As decisões foram registradas para garantir rastreabilidade entre as avaliações recebidas e a versão final do checklist.

4 Resultados

O *checklist* apresentado nesta seção teve como base os estudos analisados ao longo deste artigo, e sua construção seguiu as três etapas descritas na Seção 3: identificação das práticas sustentáveis na literatura, organização por camadas e transformação em perguntas verificáveis. As práticas listadas na metodologia serviram como ponto de partida, sendo expandidas em perguntas adicionais embasadas nas referências utilizadas. O resultado é um *checklist* voltado à ER, organizado segundo as quatro camadas de uma aplicação web (*frontend*, *backend*, banco de dados e infraestrutura), verificável e baseado na recorrência das práticas encontradas na literatura.

Tabela 2: *Checklist* para a camada de *frontend*

Categoria	Item de verificação
Efeitos visuais	O *frontend* adota seleção otimizada de efeitos visuais, evitando animações e recursos visuais desnecessários? [25]
Dependências e recursos	O sistema reduz o uso de dependências JavaScript e outros recursos que aumentam o tamanho da página? [25]
Imagens	O *frontend* utiliza formatos modernos de imagem como WebP e AVIF para reduzir o volume de dados transferidos? [25]
Requisições HTTP	O sistema minimiza o número de requisições HTTP entre cliente e servidor?[25]

Tabela 3: *Checklist* para a camada de *backend*

Categoria	Item de verificação
Cache	O *backend* utiliza mecanismos de *cache* para evitar processamento redundante e reduzir o consumo de recursos computacionais? [20, 25]
Processamento no servidor	O sistema evita operações desnecessárias no servidor, como processamento repetitivo ou consultas redundantes, visando melhorar a eficiência energética? [20]
Respostas de API	As respostas das APIs foram projetadas para transferir apenas os dados necessários, evitando sobrecarga de rede e processamento desnecessário no servidor? [25]
Ciclo de vida dos dados	O sistema possui uma política definida de ciclo de vida dos dados, estabelecendo critérios para retenção, arquivamento e exclusão de informações ao longo do tempo? [24]

Tabela 4: *Checklist* para a camada de banco de dados

Categoria	Item de verificação
Otimização de consultas	O banco de dados utiliza otimização de consultas para reduzir consumo de energia e uso de recursos? [20]
Eficiência de processamento	As consultas são revisadas periodicamente para evitar processamento desnecessário e melhorar a eficiência energética [20]
Dados obsoletos	O banco de dados realiza a remoção ou arquivamento periódico de dados obsoletos? [24]
Indexação	O banco de dados utiliza indexação adequada para reduzir tempo de processamento e consumo de recursos? [25]

Os resultados deste trabalho indicam que é possível organizar requisitos de sustentabilidade para aplicações web de forma estruturada e verificável por meio de um *checklist* orientado às camadas de aplicação [1, 18]. Para assegurar a qualidade do instrumento, a revisão colaborativa entre os integrantes do grupo contribuiu para garantir perguntas abrangentes e verificáveis, alinhando-se à abordagem de [19].

Em conjunto, os 16 itens apresentados (Tabelas 2, 3, 4, 5) compõem um instrumento prático e verificável que pode ser adotado por equipes de desenvolvimento com foco em aplicações web. Ao organizar as práticas sustentáveis por camada, o *checklist* busca tornar concretos os princípios de software verde, aproximando a teoria da prática cotidiana do desenvolvimento [1, 19]. Espera-se

Tabela 5: *Checklist* para a camada de infraestrutura

Categoria	Item de verificação
Dimensionamento de instâncias	O sistema utiliza dimensionamento adequado de instâncias para evitar o superprovisionamento de recursos computacionais? [11]
Escalabilidade automática	A infraestrutura utiliza mecanismos de escalabilidade automática conforme a demanda? [11]
Monitoramento de recursos	O consumo de recursos (CPU, memória) é monitorado regularmente para identificar desperdícios energéticos? [11, 17]
Hospedagem sustentável	Os serviços de hospedagem ou nuvem consideram práticas de eficiência energética ou uso de energia renovável? [25]

que sua adoção contribua para que aspectos como eficiência energética, uso consciente de recursos computacionais e redução do impacto ambiental sejam considerados desde as fases iniciais do desenvolvimento de aplicações web, conforme preconizado pela Engenharia de Requisitos para sustentabilidade [18].

4.1 Validação do *Checklist* por Especialistas

Para avaliar a validade de conteúdo do *checklist*, foi conduzida uma validação com cinco especialistas, cujos perfis estão descritos na Tabela 6.

Tabela 6: Perfil dos especialistas

Especialista	Experiência	Perfil	Exp. com RNF/ Sustentabilidade
E1	6–10 anos	Desenvolvedor de Software	Parcialmente
E2	6–10 anos	Pesquisador / Professor	Sim
E3	Menos de 3 anos	Desenvolvedor de Software	Não
E4	Menos de 3 anos	Desenvolvedor de Software	Sim
E5	Mais de 10 anos	Arquiteto de Software	Sim

4.1.1 Análise Quantitativa — CVR por Item. O CVR (*Content Validity Ratio*) foi calculado conforme [14]: $\text{CVR} = (N_e - N/2)/(N/2)$, onde N_e é o número de especialistas que marcaram "Essencial" e N é o total de respostas válidas por item, excluindo a opção "Não me sinto apto a avaliar". O limiar mínimo varia conforme N ($N = 5$: 0,59; $N = 4$: 0,75; $N = 3$: 0,99).

Cinco itens atingiram o limiar de CVR (BE2, BE3, BD1, BD4, IN1); os demais onze foram submetidos à análise qualitativa (Seção 4.1.2), dada a alta sensibilidade do CVR a mudanças pontuais em uma amostra pequena ($N = 5$), conforme discutido na Seção 5.2 (Validade de Conclusão).[1]

4.1.2 Análise Qualitativa — Revisão dos Itens Não Validados. Para cada item com CVR abaixo do limiar, os comentários dos especialistas foram analisados e uma decisão registrada, conforme a Tabela 7.

[1]Os valores individuais de CVR por item estão disponíveis no material suplementar no Zenodo (ver 6.1).

Tabela 7: Revisão dos itens não validados

Item	Problema identificado	Decisão
FE1	Subjetividade do enunciado ("desnecessários")	Reformular
FE2	CVR baixo sem justificativa qualitativa	Manter e reavaliar
FE3	Dependência de tecnologias específicas (WebP/AVIF)	Reformular
FE4	Rigidez prescritiva; nem sempre menos requisições é melhor	Reformular
BE1	*Cache* não se aplica universalmente	Reformular
BE4	Aplicabilidade restrita; associado à LGPD	Reformular
BD2	Periodicidade inadequada para todos os contextos	Reformular
BD3	Inaplicável em domínios com retenção obrigatória	Reformular
IN2	Risco de custo exponencial com *auto-scaling* ignorado	Remover
IN3	CVR abaixo do limiar ($N = 3$); sem sugestão de reformulação	Manter e reavaliar
IN4	Aplicabilidade restrita por dependência de provedor	Reformular

4.1.3 Análise Temática das Respostas Abertas. A análise temática das respostas abertas evidenciou oportunidades de refinamento do checklist. Um dos pontos levantados foi a subjetividade de alguns enunciados, dificultando a verificação objetiva. Também foram feitas observações sobre a dependência de tecnologias específicas em certas perguntas, como a menção direta a WebP e AVIF, sugerindo-se uma formulação mais geral. Outro aspecto recorrente foi a aplicabilidade condicionada ao contexto, alguns itens foram considerados válidos apenas em cenários específicos.

Além disso, os comentários indicaram que determinados itens possuem implicações que extrapolam a sustentabilidade técnica, como o alinhamento com exigências regulatórias, a exemplo da LGPD. Também foram registradas críticas à rigidez de enunciados, especialmente em casos em que práticas como escalabilidade automática ou minimização de requisições não podem ser tratadas como regras absolutas, pois envolvem trocas (*trade-offs*) de custo, arquitetura e contexto de uso.

As decisões de reformulação priorizaram a manutenção da testabilidade objetiva do item, enquanto itens com aplicabilidade excessivamente restrita ou risco de efeito colateral não intencional (e.g., IN2) foram removidos. Os demais itens que não estão apresentados na tabela 8 não foram modificados ou, como no caso do item do IN2, foram modificados devido a CVR negativo e por ter sido classificado como "desnecessário"pelos especialistas.

Apesar das observações, a percepção geral sobre a estrutura do instrumento foi positiva. A divisão por camadas foi considerada adequada pela maioria dos especialistas, com apenas uma avaliação parcial associada à ausência de uma camada específica de segurança. De modo semelhante, a aplicabilidade prática do *checklist* também foi bem avaliada: a maior parte dos especialistas indicou que utilizaria o instrumento em projetos reais.

5 Discussão

5.1 Desafios e Tendências

A adoção de práticas de software verde ainda esbarra em obstáculos significativos, sendo a lacuna entre a pesquisa acadêmica e a aplicação industrial a mais proeminente. Conforme apontado por [1], embora dezenas de abordagens tenham sido propostas na última década, a adoção efetiva pelas empresas permanece restrita. Esse cenário é agravado pela fragmentação de metodologias e pela escassez

Tabela 8: Itens modificados do *checklist*

ID	Versão original	Versão revisada
FE1	O *frontend* adota seleção otimizada de efeitos visuais, evitando animações e recursos visuais desnecessários?	O *frontend* utiliza efeitos visuais de forma otimizada, evitando recursos desnecessários que comprometam o desempenho?
FE3	O *frontend* utiliza formatos modernos de imagem como WebP e AVIF para reduzir o volume de dados transferidos?	O *frontend* utiliza formatos e estratégias de otimização de imagens para reduzir o volume de dados transferidos?
FE4	O sistema minimiza o número de requisições HTTP entre cliente e servidor?	O sistema otimiza o número de requisições HTTP entre cliente e servidor?
BE1	O *backend* utiliza mecanismos de *cache* para evitar processamento redundante e reduzir o consumo de recursos computacionais?	O *backend* utiliza estratégias para reduzir processamento redundante e otimizar o uso de recursos computacionais, como mecanismos de *cache* quando aplicável?
BE4	O sistema possui uma política definida de ciclo de vida dos dados, estabelecendo critérios para retenção, arquivamento e exclusão de informações ao longo do tempo?	O sistema possui uma política definida de ciclo de vida dos dados, estabelecendo critérios para retenção, arquivamento e exclusão de informações ao longo do tempo seguindo critérios legais como a LGPD?
BD1	O banco de dados utiliza otimização de consultas para reduzir consumo de energia e uso de recursos?	O banco de dados utiliza consultas otimizadas para reduzir o uso de recursos computacionais?
BD2	As consultas são revisadas periodicamente para evitar processamento desnecessário e melhorar a eficiência energética.	As consultas críticas são monitoradas e revisadas para evitar processamento desnecessário?
BD3	O banco de dados realiza a remoção ou arquivamento periódico de dados obsoletos?	O banco de dados realiza remoção ou arquivamento periódico de dados obsoletos quando necessário?
IN4	Os serviços de hospedagem ou nuvem consideram práticas de eficiência energética ou uso de energia renovável?	Os serviços de hospedagem ou nuvem consideram práticas de eficiência energética ou uso de energia renovável quando possível?

de métricas padronizadas, como o Software Carbon Intensity (SCI, ISO/IEC 21031:2024), cuja adoção na indústria ainda é incipiente [1, 6].

No escopo específico das aplicações web, a gestão de trocas (*trade-offs*) desponta como outro desafio arquitetural constante. A implementação de padrões verdes frequentemente esbarra no aumento da complexidade técnica do sistema ou em restrições de flexibilidade na interface [25]. Somado a isso, as equipes enfrentam o risco do chamado efeito *rebound*, fenômeno no qual os ganhos de eficiência energética são rapidamente anulados pelo aumento proporcional no volume de uso do sistema ou tráfego de rede [22]. Isso exige que a sustentabilidade não seja apenas uma meta estática, mas um requisito monitorado continuamente.

5.2 Ameaças à Validade e Limitações

A condução deste estudo está sujeita a ameaças à validade que devem ser consideradas na interpretação de seus resultados. Tais limitações foram mitigadas na medida do possível, conforme detalhado a seguir.

Validade de Construto. A principal ameaça refere-se à métrica de avaliação utilizada pelo painel. O cálculo do CVR baseia-se na classificação dos itens nas categorias essenciais, úteis, não essenciais e desnecessários. A interpretação desses termos pode variar entre os avaliadores, introduzindo subjetividade. Para atenuar essa ameaça, o formulário incluiu comentários qualitativos, usados para orientar o refinamento dos itens rejeitados quantitativamente.

Validade Interna. As ameaças internas estão associadas à composição do painel de especialistas. O número de participantes ($N = 5$) é uma limitação que afeta a estabilidade dos resultados quantitativos. Adicionalmente, o processo de seleção por conveniência pode introduzir viés de amostragem. Buscou-se mitigar essa questão compondo um painel heterogêneo, contemplando perfis da indústria e da academia, com diferentes níveis de experiência em desenvolvimento de software e em sustentabilidade.

Validade Externa. A generalização dos resultados é restrita pelo escopo definido na pesquisa. O *checklist* foi construído e avaliado especificamente para o contexto de aplicações web. Consequentemente, as práticas e os itens de verificação não podem ser diretamente extrapolados para outros domínios, como sistemas embarcados ou aplicações móveis nativas, sem adaptação e nova validação. Ademais, painéis compostos por profissionais de outros nichos poderiam avaliar a essencialidade dos itens de forma distinta.

Validade de Conclusão. Esta validade refere-se à relação entre o tratamento e o resultado. Devido ao tamanho reduzido do painel, a concordância quantitativa, medida pelo CVR, apresenta alta sensibilidade. Uma mudança pontual na avaliação de um único especialista tem impacto significativo no alcance dos limiares de aceitação. Por esse motivo, as decisões finais sobre a manutenção ou reformulação dos itens não se basearam exclusivamente no limiar matemático, sendo estritamente atreladas a uma rodada de análise qualitativa das respostas abertas para garantir a robustez metodológica das conclusões.

6 Considerações Finais

Este estudo discutiu a relevância da incorporação de requisitos verdes no desenvolvimento de software, com foco em aplicações web, propondo um *checklist* estruturado no formato de perguntas para apoiar a elicitação, organização e verificação de requisitos de sustentabilidade. Sua principal contribuição é aproximar os conhecimentos teóricos discutidos da prática de desenvolvimento, auxiliando equipes na consideração de eficiência energética, uso consciente de recursos computacionais e redução do impacto ambiental.

6.1 Lições Aprendidas e Trabalhos Futuros

O desenvolvimento deste trabalho evidenciou a interdependência entre as camadas; otimizações isoladas podem ser anuladas por ineficiências em outras e a ausência de padronização terminológica na literatura, já apontada por [1], além da competição entre requisitos de sustentabilidade e restrições de desempenho e custo. Como trabalhos futuros, destaca-se a validação empírica do *checklist* em projetos reais, associando seus itens a métricas quantitativas como o *Software Carbon Intensity* (SCI) [6], e a investigação de sua integração a ferramentas de ER ou *pipelines* de integração contínua.

Disponibilidade de Artefatos

O pacote de artefatos anonimizado que apoia este estudo está disponível no Zenodo em https://doi.org/10.5281/zenodo.21362388. O material contém o protocolo de pesquisa, as fórmulas aplicadas durante a análise, o modelo de formulário e a análise das respostas.

Referências

[1] Peter Bambazek, Iris Groher, and Norbert Seyff. 2023. Requirements Engineering for Sustainable Software Systems: A Systematic Mapping Study. *Requirements Engineering* 28, 3 (2023), 481–505. doi:10.1007/s00766-023-00402-1

[2] Christoph Becker et al. 2016. Requirements: The Key to Sustainability. *IEEE Software* 33, 1 (2016), 56–65.

[3] Christoph Becker, Stefanie Betz, Rina Jain, Karine Bashar, Birgit Penzenstadler, Colin C. Venters, and Norbert Seyff. 2019. Sustainability Design in Software Engineering: The SusAF Method. In *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*. IEEE, Montreal, QC, Canada, 46–47. doi:10.1109/ICSE-Companion.2019.00038

[4] Christoph Becker, Ruzanna Chitchyan, Leticia Duboc, Steve Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin C. Venters. 2015. The Karlskrona Manifesto for Sustainability Design. In *Proceedings of the 37th International Conference on Software Engineering (ICSE)*. IEEE, Florence, Italy, 467–476.

[5] Amel Bennaceur, Thein Than Tun, Yijun Yu, and Bashar Nuseibeh. 2019. Requirements Engineering. In *Software Engineering*. Springer, Cham, Switzerland, 55–77.

[6] Andreas Brunnert. 2024. *Green Software Metrics*. Technical Report. Munich University of Applied Sciences.

[7] John W Creswell. 2014. *Qualitative, quantitative and mixed methods approaches*. Sage, Thousand Oaks, CA, USA.

[8] André de Moura e Souza, Eunice P. dos Santos Nunes, and Patrícia Cristiane de Souza. 2024. *Desenvolvimento Sustentável de Software e Eficiência Energética do Código: Um Mapeamento Sistemático*. Technical Report. Instituto de Computação – UFMT.

[9] Markus Dick and Stefan Naumann. 2010. Enhancing Software Engineering Processes towards Sustainable Software Product Design. In *Proceedings of EnviroInfo 2010: Integration of Environmental Information in Europe*. Shaker, Aachen, 706–715.

[10] Markus Dick, Stefan Naumann, and Norbert Kuhn. 2010. A Model and Selected Instances of Green and Sustainable Software. In *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience*, Jacques Berleur, Magda D. Hercheui, and Lorenz M. Hilty (Eds.). IFIP Advances in Information and Communication Technology, Vol. 328. Springer, Berlin, Heidelberg, 248–259. doi:10.1007/978-3-642-15479-9_24

[11] Brian Dougherty, Jules White, and Douglas C. Schmidt. 2012. Model-driven Autoscaling of Green Cloud Computing Infrastructure. *Future Generation Computer Systems* 28, 2 (2012), 371–378. doi:10.1016/j.future.2011.05.016

[12] Mariona González-Sordé. 2025. Exploring the Potential of Easy Language for Enhancing Website Sustainability. *Universal Access in the Information Society* 24 (2025), 1905–1915. doi:10.1007/s10209-024-01129-8

[13] Leila Karita, Brunna Caroline Mourão, and Ivan Machado. 2022. Towards a Common Understanding of Sustainable Software Development. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering (SBES '22)*. ACM, Uberlândia, MG, Brazil, 269–278. doi:10.1145/3555228.3555236

[14] C. H. Lawshe. 1975. A Quantitative Approach to Content Validity. *Personnel Psychology* 28, 4 (1975), 563–575. doi:10.1111/j.1744-6570.1975.tb01393.x

[15] Ricardo Mancebo, Coral Calero, M. Ángeles Moraga, and Miguel A. García. 2021. Requirements Engineering for Energy-Efficient Mobile Applications: A Systematic Literature Review. *Journal of Systems and Software* 172 (2021), 110846. doi:10.1016/j.jss.2020.110846

[16] Brunna C. Mourão, Leila Karita, and Ivan do Carmo Machado. 2018. Green and Sustainable Software Engineering – A Systematic Mapping Study. In *Proceedings of the 17th Brazilian Symposium on Software Quality (SBQS)*. ACM, Curitiba, Brazil, 121–130. doi:10.1145/3275245.3275258

[17] Stefan Naumann, Markus Dick, Eva Kern, and Timo Johann. 2011. The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. *Sustainable Computing: Informatics and Systems* 1, 4 (2011), 294–304. doi:10.1016/j.suscom.2011.06.004

[18] Birgit Penzenstadler. 2014. Infusing Green: Requirements Engineering for Green In and Through Software Systems. In *Proceedings of the International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy@RE)*. CEUR-WS.org, Karlskrona, Sweden, 44–53.

[19] Pak-Lok Poon, T. H. Tse, Sau-Fun Tang, and Fei-Ching Kuo. 2011. Contributions of Tester Experience and a Checklist Guideline to the Identification of Categories and Choices for Software Testing. *Software Quality Journal* 19, 1 (2011), 141–163. doi:10.1007/s11219-010-9109-4

[20] Giuseppe Procaccianti, H. H. Fernandez, and Patricia Lago. 2016. Empirical Evaluation of Two Best Practices for Energy-Efficient Software Development. *Journal of Systems and Software* 117 (2016), 185–198. doi:10.1016/j.jss.2016.02.035

[21] Antonello Sala, Lorenzo Barbetti, and Andrea Rosini. 2024. Green Web Meter: Structuring and Implementing a Real-Time Digital Sustainability Monitoring System. *Sustainability* 16, 17 (2024), 7627. doi:10.3390/su16177627

[22] Abimael Santos, Quelita Ribeiro, Jaelson Castro, and Maria Lencastre. 2024. Catálogo de Requisitos de Sustentabilidade: Um Estudo Preliminar. In *Proceedings of the 27th Workshop on Requirements Engineering (WER24)*. WER, Buenos Aires, Argentina.

[23] Rodrigo Soares V. Teixeira and Karla Donato Fook. 2024. Catálogo de Requisitos para Software Sustentável. *SBC OpenLib* (2024).

[24] Geert-Jan van Bussel, Nikki Smit, and John van de Pas. 2015. Digital Archiving, Green IT and Environment: Deleting Data to Manage Critical Effects of the Data Deluge. *Electronic Journal of Information Systems Evaluation* 18, 2 (2015), 187–198.

[25] Waruni Wimalasena and Nimasha Arambepola. 2025. Toward a Greener Web: A Systematic Literature Review of Sustainable Practices in Web Development. In *Proceedings of the 5th International Conference on Advanced Research in Computing (ICARC)*. IEEE, Belihuloya, Sri Lanka, 1–6. doi:10.1109/ICARC64760.2025.10962990

[26] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. 2012. *Experimentation in software engineering*. Vol. 236. Springer, Berlin, Heidelberg, Germany.