Tô De Olho: Democratizando a Transparência do Senado Federal através de Dados Abertos

Trabalho de Conclusão de Curso

Pedro Batista de Almeida Filho

Pablo Vieira Florentino
Orientador

Instituto Federal da Bahia - IFBA
Curso de Análise e Desenvolvimento de Sistemas
Campus Salvador

Salvador, Bahia, Brasil
Fevereiro de 2026

Sumário

1 Visão geral

O *Tô De Olho* é uma plataforma *web* de transparência parlamentar focada no Senado Federal. Sua proposta é aproximar cidadãos dos dados legislativos oficiais, convertendo informação dispersa em conhecimento fiscalizável e de fácil compreensão. O projeto vai além dos dados abertos básicos, integrando fontes complexas como a Cota para o Exercício da Atividade Parlamentar dos Senadores (CEAPS) e as "emendas PIX". Ao combinar uma arquitetura de **monolito modular** em Go, ingestão via APIs oficiais (Senado e Portal da Transparência) e um *front-end* em Next.js, a plataforma busca reduzir a assimetria de informação sobre os 81 senadores da República [1].

A literatura de democracia digital evidencia que TICs ampliam possibilidades de participação, mas só geram valor quando articuladas a contextos de uso e inclusão. Avelino et al. mapeiam iniciativas de dados abertos, reforçando que tecnologias precisam ser mediadas por visualizações claras para efetivação do controle social [2]. Com um corpo legislativo menor e mais "caro" *per capita* que a Câmara, o Senado carece de ferramentas focadas que cruzem votações nominais com a execução orçamentária de emendas. À luz desses estudos, o *Tô De Olho* procura transformar a transparência passiva em *accountability* ativa, oferecendo rankings e métricas objetivas de desempenho parlamentar [3, 4].

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver uma plataforma *web* de transparência política que centralize, organize e simplifique o acesso aos dados públicos do Senado Federal, fomentando a fiscalização cidadã e o debate qualificado sobre a atuação dos 81 senadores, com ênfase no monitoramento de gastos e emendas parlamentares.

1.1.2 Objetivos Específicos

- Implementar um **backend em Go** com arquitetura de monolito modular para ingestão de dados das APIs oficiais do Senado e do Portal da Transparência;
- Desenvolver rotinas ETL para consumir as APIs Legislativa, Administrativa e do Portal da Transparência, priorizando fontes estruturadas;
- Criar algoritmos de *Ranking* para avaliar senadores com base em presença em votações, produtividade legislativa, economia na cota parlamentar, participação em comissões e transparência de dados;
- Construir uma interface *front-end* responsiva utilizando Next.js, permitindo a visualização intuitiva de perfis, despesas e *scorecards* de fiscalização.

1.2 Declaração do Problema

O Senado Federal disponibiliza dados públicos por meio de APIs próprias, enquanto a Controladoria-Geral da União (CGU) mantém o Portal da Transparência com dados de emendas parlamentares. Contudo, essas fontes encontram-se fragmentadas em órgãos distintos: a API Legislativa do Senado concentra informações sobre matérias e votações; a API Administrativa do Senado reúne dados da CEAPS e remunerações de gabinete; e o Portal da Transparência da CGU hospeda os registros de emendas e transferências federais. Para construir uma visão completa de um único senador, o cidadão precisaria consultar três sistemas de dois órgãos diferentes, com interfaces, formatos e periodicidades de atualização distintos.

Essa fragmentação adquire contornos mais graves quando analisamos as "Transferências Especiais" — popularmente conhecidas como "emendas PIX". Criada em 2019, essa modalidade dispensa convênio e transfere recursos federais diretamente a estados e municípios. Alencar [5] demonstra que, do total de R$ 20,5 bilhões transferidos por essa via, apenas R$ 933 milhões tiveram prestação de contas adequada — menos de 5%. Em 2020, primeiro ano de vigência, as transferências especiais representavam 6,4% das emendas individuais; em 2023, esse percentual saltou para 32,4%. A distribuição é ainda mais desigual: no mesmo estado, alguns municípios receberam mais de R$ 4.500 *per capita*, enquanto outros receberam menos de R$ 1 — sem qualquer justificativa pública dos parlamentares.

Além da barreira técnica imposta pela fragmentação dos dados, há uma barreira social igualmente relevante. Segundo o Indicador de Alfabetismo Funcional [6], 29% da população brasileira entre 15 e 64 anos é funcionalmente analfabeta, o que limita severamente a capacidade de interpretar planilhas, gráficos e relatórios disponibilizados nos portais oficiais. Nesse contexto, a simples disponibilização de dados brutos não garante transparência efetiva: é necessária uma ferramenta que consolide as informações dispersas e as apresente de forma visual e acessível, permitindo ao cidadão comum avaliar qualitativamente seus representantes [2].

1.3 Proposta de Solução de Software

Diante da fragmentação de dados descrita e da barreira de letramento que impede o cidadão comum de interpretar planilhas e relatórios oficiais, propõe-se o *Tô De Olho*: uma plataforma *web* de código aberto concebida para centralizar a fiscalização do Senado Federal. A solução integra três APIs oficiais distintas — Legislativa do Senado, Administrativa do Senado e Portal da Transparência da CGU — consolidando informações dispersas em uma interface única e acessível.

O sistema organiza os dados em três dimensões complementares do mandato parlamentar:

- **Atividade Legislativa**: votações nominais, participação em comissões, proposições de autoria e relatorias;
- **Gestão de Recursos**: despesas detalhadas da Cota Parlamentar (CEAPS), com identificação de fornecedores e categorias de gasto;

- **Articulação Orçamentária**: emendas parlamentares com destaque para Transferências Especiais (“emendas PIX”), permitindo rastrear o destino dos recursos.

O diferencial da plataforma reside em quatro pilares:

1. **Ranking Metodologicamente Fundamentado**: inspirado no *State Legislative Effectiveness Score* (SLES) de Volden e Wiseman [7], o algoritmo de avaliação pondera produtividade legislativa (35%), presença em votações (25%), economia na cota parlamentar (20%) e participação em comissões (20%). Os critérios e pesos são públicos, permitindo ao cidadão compreender — e questionar — a metodologia;

2. **Visualização Orientada à Ação**: seguindo os princípios de retórica visual de Hullman [8], cada dado absoluto é contextualizado com médias comparativas, reduzindo a possibilidade de interpretações manipuladas e estimulando conclusões informadas;

3. **Acessibilidade como Requisito**: a interface segue as diretrizes WCAG 2.1 nível AA, garantindo navegação por leitores de tela, contraste adequado e operação via teclado — essencial para atingir os 29% de brasileiros funcionalmente analfabetos identificados pelo INAF [6];

4. **Consolidação Multi-Fonte**: ao integrar dados de três órgãos distintos em uma única consulta, a plataforma elimina a necessidade de o cidadão navegar por sistemas heterogêneos com formatos e interfaces incompatíveis.

Em síntese, o *Tô De Olho* atua como um “auditor digital”, automatizando cruzamentos de dados que, manualmente, seriam inviáveis para o eleitor comum. O objetivo não é substituir a análise crítica do cidadão, mas fornecer-lhe ferramentas para exercê-la de forma qualificada.

1.4 Trabalhos Relacionados

Diversas iniciativas no Brasil e no mundo buscam promover a transparência política por meio da tecnologia. À luz da Escada de Participação de Arnstein [9], podemos classificar essas ferramentas conforme o grau de poder que conferem ao cidadão.

1.4.1 Portais Oficiais (Nível Informação)

Portal da Transparência (CGU): Lançado em novembro de 2004 pela Controladoria-Geral da União, o Portal da Transparência consolidou-se como a principal ferramenta oficial do governo federal para acesso a dados de gastos públicos, servidores e transferências [10]. Em 2018, o portal passou por reformulação completa para tornar a navegação mais intuitiva, e em 2024, ao completar 20 anos, recebeu novas atualizações que reafirmaram seu papel central no controle social. Os dados disponíveis abrangem execução orçamentária detalhada por órgão, remuneração individualizada de servidores, pagamentos de programas sociais (Bolsa Família, Auxílio Gás, Pé-de-Meia), licitações, contratos e — particularmente relevante para este trabalho — emendas parlamentares, incluindo registros relacionados à ADPF 854 sobre transparência orçamentária.

O portal registra entre 1,3 e 1,5 milhão de usuários únicos mensais, com aproximada-

mente 14 a 19 milhões de visualizações de página, demonstrando alto engajamento da sociedade civil e órgãos de controle. Para desenvolvedores, oferece uma API REST com limites de 90 requisições por minuto em horário comercial e 300 requisições por minuto durante a madrugada. Contudo, algumas limitações persistem: a periodicidade de atualização varia significativamente entre conjuntos de dados — enquanto despesas e emendas são atualizadas diariamente, dados de imóveis funcionais podem apresentar defasagem superior a seis meses. Além disso, a granularidade contábil (com termos técnicos do Siafi — Sistema Integrado de Administração Financeira do Governo Federal) representa barreira para cidadãos sem conhecimento em contabilidade pública.

Portal de Dados Abertos do Senado Federal: Lançado em 2012 em conformidade com a Lei de Acesso à Informação [11], o portal foi institucionalizado pelo Ato da Comissão Diretora n. 14 de 2013, que estabeleceu a Política de Dados Abertos do Senado [12]. O ecossistema divide-se em duas APIs principais: a **API Legislativa**, que oferece dados sobre matérias, votações nominais, senadores e atividades de comissões; e a **API Administrativa**, focada em transparência de gastos (CEAPS), gestão de pessoas, orçamento e contratos. Os formatos suportados incluem JSON, XML e CSV, com documentação técnica via Swagger UI. A API possui limite de 10 requisições por segundo para garantir estabilidade.

Portal de Dados Abertos da Câmara dos Deputados: O fornecimento de dados legislativos pela Câmara iniciou-se em 2006 através do sistema SIT Câmara (Web Services) [13]. Com a Lei de Acesso à Informação em 2011, o portal foi rebatizado como “Dados Abertos”, eliminando a obrigatoriedade de cadastro prévio. Em 2017, lançou-se a API RESTful v2, substituindo os antigos Web Services por arquitetura mais moderna. O portal oferece *endpoints* para deputados (perfis biográficos, discursos, frentes parlamentares), proposições (texto integral, tramitação), votações (incluindo voto individual de cada parlamentar) e cotas parlamentares (CEAP). Os formatos incluem JSON e XML via API, além de CSV, XLSX e ODS para *downloads* em massa. O portal é referência em transparência legislativa, alimentando projetos como a Operação Serenata de Amor, Radar Governamental e VotoBom.

Embora esses portais representem avanços significativos na transparência passiva, situam-se no degrau mais básico da Escada de Arnstein — informação bruta sem mediação interpretativa. O cidadão comum, sem conhecimento técnico sobre APIs ou contabilidade pública, enfrenta barreiras substanciais para transformar dados dispersos em fiscalização efetiva.

1.4.2 Ferramentas de Fiscalização da Câmara dos Deputados

O ecossistema de transparência para a Câmara dos Deputados é mais desenvolvido que para o Senado, contando com diversas iniciativas consolidadas:

Operação Serenata de Amor: Projeto pioneiro de código aberto, lançado em 2016 via financiamento coletivo, que utiliza inteligência artificial para detectar irregularidades em gastos parlamentares [14]. Desenvolvido pela Open Knowledge Brasil, é composto por dois sistemas complementares:

- **Rosie:** Algoritmo de *machine learning* desenvolvido em Python que audita a Cota para Exercício da Atividade Parlamentar (CEAP). Opera através de cinco classificadores prin-

cipais: (1) *meal price outlier* — identifica refeições com valores acima da média para o local; (2) *irregular companies* — detecta gastos em empresas com situação cadastral irregular na Receita Federal; (3) *traveled speeds* — cruza gastos para identificar deslocamentos fisicamente impossíveis; (4) *monthly subquota limit* — verifica excesso nos limites mensais por categoria; e (5) *election expenses* — identifica uso indevido da cota para financiar campanhas.

- **Jarbas:** Interface *web* desenvolvida em Django que permite aos cidadãos navegar pelos casos suspeitos identificados pela Rosie, visualizar notas fiscais digitalizadas e formalizar denúncias.

Até 2018, o projeto identificou **8.276 reembolsos suspeitos** envolvendo 735 deputados, totalizando aproximadamente **R$ 3,6 milhões** em potenciais irregularidades. Um mutirão inicial resultou em 629 denúncias formais ao Congresso. Atualmente, a equipe principal migrou o foco para o projeto "Querido Diário", que aplica técnicas similares a diários oficiais municipais.

De Olho no Congresso: Plataforma *web* moderna focada em gastos de Deputados Federais [15]. Com mais de **55 mil visitantes** e **95 mil consultas** realizadas, a ferramenta oferece interface acessível que inclui:

- **Rankings Múltiplos:** Top 50 deputados com maiores gastos, ranking de partidos por consumo da cota e ranking de empresas fornecedoras — incluindo filtro específico para "empresas com sanções" administrativas;
- **Painel de Alertas:** Sistema de detecção automática de despesas atípicas, incluindo: valores significativamente acima da média geral, pagamentos idênticos repetidos ao mesmo fornecedor, notas fiscais emitidas em finais de semana, e intervalos menores que 3 dias entre pagamentos;
- **Histórico Completo:** Gastos anuais e mensais com filtros por fornecedor, categoria e período, além de detalhamento de benefícios (auxílio-moradia, imóvel funcional) e equipe de gabinete.

A plataforma ressalta que os alertas são "indicativos que merecem investigação", servindo como guia para auditoria cidadã. Limita-se, porém, à Câmara dos Deputados e não oferece métricas de desempenho legislativo.

De Olho em Você: Plataforma *web* com foco em "transparência que dá para entender", abrangendo aproximadamente 549 parlamentares da Câmara dos Deputados [16]. A plataforma integra dados da API da Câmara e do Portal da Transparência, destacando-se pela cobertura das **Emendas PIX** (Transferências Especiais). Entre suas funcionalidades principais:

- **Mapas de Distribuição:** Cada perfil de deputado apresenta visualização geoespacial do destino de suas emendas parlamentares, permitindo identificar concentração de recursos por município;
- **Ranking de Cidades:** Classificação por faixa populacional das cidades que mais rece-

beram Transferências Especiais (ex: municípios de até 20 mil habitantes);

- **Comparador de Parlamentares:** Ferramenta que permite selecionar de 2 a 5 deputados para comparação lado a lado de gastos de cota, equipe de gabinete, emendas enviadas e fornecedores em comum;
- **Painel de Fornecedores:** Ranking das empresas que mais recebem recursos, identificando padrões de concentração de gastos.

Entretanto, os rankings do "De Olho em Você" baseiam-se em métricas agregadas diretas (quem mais gastou, quem mais enviou emendas), **sem metodologia explícita de efetividade legislativa**. Além disso, a plataforma não contempla o Senado Federal.

1.4.3 Experiências Internacionais

TheyWorkForYou (Reino Unido): Lançada em 2004 e operada pela organização sem fins lucrativos mySociety [17], a plataforma monitora cinco parlamentos britânicos: Câmara dos Comuns, Câmara dos Lordes, Parlamento Escocês, Senedd (País de Gales) e Assembleia da Irlanda do Norte [18]. Em 2023/24, registrou mais de **4,8 milhões de visitas**. Entre suas funcionalidades:

- **Hansard Pesquisável:** Arquivo completo de todos os discursos e debates parlamentares;
- **Alertas por E-mail:** Notificações automáticas quando um parlamentar específico discursa ou quando uma palavra-chave é mencionada;
- **Busca por Código Postal:** Identificação imediata do representante local.

A plataforma consolidou-se como referência mundial em *civic tech* parlamentar, inspirando iniciativas em diversos países.

OpenSecrets (Estados Unidos): Principal organização de pesquisa sobre dinheiro na política americana, resultante da fusão em 2021 entre o *Center for Responsive Politics* (fundado em 1983 por dois ex-senadores) e o *National Institute on Money in State Politics* [19]. Vencedora de múltiplos *Webby Awards*, a plataforma rastreia:

- **Financiamento de Campanhas:** Contribuições individuais, PACs e Super PACs;
- **Lobbying:** Gastos de empresas e grupos de interesse para influenciar legislação;
- **Revolving Door:** Monitoramento de ex-congressistas que se tornaram lobistas;
- **Dark Money:** Análise de fundos de origem não divulgada que influenciam eleições.

É fonte primária para veículos como *The New York Times* e *The Washington Post*, oferecendo APIs e exportações de dados para pesquisadores acadêmicos.

1.4.4 Lacuna Identificada e Diferencial do Tô De Olho

A análise sistemática dos trabalhos relacionados evidencia um cenário paradoxal: enquanto a Câmara dos Deputados — com 513 parlamentares — dispõe de ao menos três plataformas consolidadas de fiscalização cidadã, o Senado Federal permanece como uma "caixa preta" digital. Essa lacuna não é trivial. Os 81 senadores exercem mandatos de oito anos, atuam como câmara revisora de toda legislação federal e detêm competências exclusivas de alto impacto: confirmação de ministros do STF, julgamento de presidentes da República e aprovação de dívidas externas. A ausência de ferramentas de monitoramento específicas representa, portanto, uma falha sistêmica no ecossistema de *accountability* brasileiro.

Mais do que apenas replicar soluções existentes para o âmbito senatorial, o *Tô De Olho* propõe-se a **sintetizar o melhor de cada iniciativa** analisada, superando limitações identificadas:

- **Do "De Olho em Você"**, incorporamos a **visualização geoespacial de Emendas PIX** — permitindo que o cidadão identifique, em mapas interativos, quais municípios receberam recursos de cada senador — e o **comparador de parlamentares**, que possibilita análise lado a lado de até cinco senadores em múltiplas dimensões;
- **Do "De Olho no Congresso"**, adotamos o **painel de alertas automáticos** para despesas atípicas — notas fiscais em finais de semana, valores acima da média, pagamentos repetidos em intervalos curtos — e o **ranking de fornecedores**, incluindo cruzamento com empresas sob sanção administrativa;
- **Do "Serenata de Amor"**, inspiramo-nos na abordagem de **código aberto** e **documentação transparente**, permitindo que pesquisadores e jornalistas repliquem e validem os resultados.

O diferencial central do *Tô De Olho*, entretanto, reside em uma contribuição original: a implementação de um **Índice de Efetividade Legislativa** adaptado ao contexto brasileiro. Enquanto as ferramentas existentes limitam-se a ordenar parlamentares por *volume de gastos* — métrica que penaliza a parcimônia — ou *quantidade de proposições* — que ignora a qualidade e o impacto legislativo —, propomos um modelo multidimensional inspirado no *State Legislative Effectiveness Score* (SLES) de Volden e Wiseman [7].

Nosso índice pondera quatro dimensões objetivas, com pesos públicos e metodologia reprodutível:

1. **Produtividade Legislativa (35%)**: Avalia a capacidade de transformar proposições em leis, com multiplicadores por tipo (PEC: 3x, PLP: 2x) e estágio de tramitação alcançado;
2. **Presença em Votações (25%)**: Mensura o comparecimento efetivo às sessões deliberativas, descontando ausências justificadas por licença médica ou missão oficial;
3. **Economia na Cota Parlamentar (20%)**: Compara o uso individual da CEAPS com a mediana do Senado, premiando a eficiência no uso de recursos públicos;
4. **Participação em Comissões (20%)**: Pondera o engajamento em comissões permanen-

tes e especiais, com bônus para cargos de liderança (presidente, relator).

A exposição pública de critérios e pesos não é mera formalidade: representa um compromisso ético com a **transparência metodológica**. Diferente de rankings opacos, o cidadão poderá compreender — e questionar — os fundamentos da classificação, evitando que a plataforma seja percebida como veículo de viés político.

Em síntese, o *Tô De Olho* posiciona-se como a **primeira plataforma integrada** de fiscalização cidadã voltada ao Senado Federal, combinando três vertentes complementares:

1. **Consolidação Multi-Fonte**: Integra dados de três APIs oficiais (Legislativa, Administrativa e Portal da Transparência) em interface única;
2. **Inteligência de Dados**: Oferece alertas automáticos, rankings comparativos e visualizações geoespaciais que transformam dados brutos em informação acionável;
3. **Rigor Metodológico**: Fundamenta-se em literatura acadêmica sobre efetividade legislativa, com metodologia aberta a auditoria pública.

2 Fundamentação Teórica

O desenvolvimento de uma plataforma de transparência parlamentar situa-se na interseção de múltiplos campos do conhecimento: direito à informação, ciência política, teoria democrática, design de interação e engenharia de software. Compreender essas bases teóricas é essencial não apenas para fundamentar as decisões de projeto, mas também para posicionar a ferramenta no debate mais amplo sobre controle social e qualidade da democracia.

Esta seção organiza-se em três eixos complementares. O primeiro eixo — **normativo-institucional** — examina os marcos legais e conceituais que sustentam a transparência governamental no Brasil, desde a Lei de Acesso à Informação até os compromissos internacionais de governo aberto. O segundo eixo — **político-participativo** — aborda as teorias de democracia digital e accountability, investigando como tecnologias podem (ou não) fortalecer a participação cidadã e a responsabilização de agentes públicos. O terceiro eixo — **técnico-metodológico** — apresenta os fundamentos de mensuração de efetividade legislativa, visualização de dados e arquitetura de sistemas que orientam a implementação do *Tô De Olho*.

2.1 Eixo Normativo-Institucional

Este eixo examina os fundamentos jurídicos e conceituais que regulamentam o acesso à informação pública no Brasil, bem como os desafios de transparência em modalidades orçamentárias específicas.

2.1.1 Transparência Pública e Dados Abertos

A transparência governamental constitui pilar fundamental do Estado Democrático de Direito. No Brasil, a Lei de Acesso à Informação (LAI — Lei nº 12.527/2011) estabelece que o

acesso é a regra e o sigilo, a exceção, garantindo aos cidadãos o direito de solicitar e receber informações públicas sem necessidade de justificativa [11]. A LAI impõe aos órgãos públicos o dever de divulgação proativa de informações de interesse coletivo, incluindo dados sobre despesas, contratos e remunerações de servidores.

A literatura distingue duas modalidades complementares de transparência. A **transparência ativa** ocorre quando o Estado disponibiliza informações de forma proativa em portais e bases de dados, independentemente de solicitação — exemplificada pelos Portais de Dados Abertos do Senado Federal e da Câmara dos Deputados. A **transparência passiva**, por sua vez, responde às solicitações dos cidadãos via canais específicos como o Sistema Eletrônico do Serviço de Informação ao Cidadão (e-SIC). Enquanto a primeira amplia o acesso massivo a dados estruturados, a segunda garante o direito individual à informação específica [2]. Pesquisas de Michener *et al.* demonstram que, cinco anos após a promulgação da LAI, a transparência permanece significativamente mais fraca nos níveis estadual e municipal, com índices de compliance inferiores a 50% em muitos entes federativos [20].

O conceito de *Open Government Data* (Dados Governamentais Abertos) vai além da simples disponibilização: preconiza que as informações públicas devem ser liberadas em formatos abertos, processáveis por máquina e livres de licenças restritivas. Tim O'Reilly, em ensaio seminal que cunhou o termo "Government as a Platform", argumenta que governos devem atuar como **plataformas** que habilitam a inovação cidadã, em vez de provedores diretos de serviços isolados [21]. Nessa visão, APIs de dados abertos funcionam como infraestrutura sobre a qual a sociedade civil constrói ferramentas de fiscalização — exatamente o que o *Tô De Olho* realiza. Tim Berners-Lee, inventor da *World Wide Web*, complementa essa perspectiva ao propor em 2010 uma escala de cinco estrelas para avaliar a qualidade dos dados abertos [22]:

1. **Uma estrela**: Dados disponíveis na web em qualquer formato, sob licença aberta (ex: PDF escaneado);
2. **Duas estrelas**: Dados estruturados e legíveis por máquina (ex: planilha Excel);
3. **Três estrelas**: Dados em formato aberto não-proprietário (ex: CSV ao invés de Excel);
4. **Quatro estrelas**: Dados identificados por URIs, seguindo padrões W3C como RDF;
5. **Cinco estrelas**: Dados linkados a outras fontes, formando uma rede de conhecimento interoperável (*Linked Open Data*).

Os portais brasileiros de dados abertos situam-se predominantemente entre duas e três estrelas: oferecem arquivos CSV e JSON processáveis por máquina, mas raramente implementam identificadores únicos (URIs) ou linkagem semântica entre bases de dados de diferentes órgãos. A fragmentação identificada na Declaração do Problema — três APIs de dois órgãos distintos — exemplifica essa limitação: embora os dados sejam tecnicamente "abertos", a ausência de interoperabilidade impõe barreiras significativas à consolidação e análise integrada [2].

O Brasil integra a *Open Government Partnership* (OGP) desde 2011, tendo desenvolvido

seis planos de ação nacionais com participação da sociedade civil. Essas iniciativas resultaram em 130 reformas voltadas à melhoria da governança e ao fortalecimento da Lei de Acesso à Informação [23]. Contudo, como demonstram as pesquisas sobre dados abertos, a mera disponibilização de informações não garante transparência efetiva: é necessário que os dados alcancem o público, que este tenha capacidade de processá-los, e que existam mecanismos institucionais para responsabilização dos agentes públicos.

2.1.2 Emendas PIX e Desafios de Transparência Orçamentária

As Transferências Especiais, popularmente conhecidas como "emendas PIX", constituem modalidade de repasse de recursos federais a estados e municípios criada pela Emenda Constitucional nº 105/2019. Diferentemente de convênios tradicionais, que exigem plano de trabalho prévio e prestação de contas detalhada, as emendas PIX transferem recursos diretamente às contas dos entes federativos com discricionariedade ampla sobre sua aplicação.

A pesquisa de Alencar [5] revela deficiências graves de transparência fiscal nesta modalidade. Do total de R$ 20,5 bilhões transferidos via emendas PIX até 2023, apenas R$ 933 milhões — menos de 5% — tiveram prestação de contas adequada. A evolução do instrumento é expressiva: em 2020, primeiro ano de vigência, as transferências especiais representavam 6,4% das emendas individuais; em 2023, esse percentual saltou para 32,4%.

A distribuição territorial revela disparidades extremas: no mesmo estado, alguns municípios receberam mais de R$ 4.500 *per capita* via emendas PIX, enquanto outros receberam menos de R$ 1 — sem qualquer justificativa pública dos parlamentares sobre os critérios de alocação. Esta opacidade compromete as três vertentes de *accountability* discutidas posteriormente nesta seção: a **vertical** (eleitores não conseguem avaliar escolhas de seus representantes), a **horizontal** (tribunais de contas enfrentam dificuldades de fiscalização) e a **social** (jornalistas e pesquisadores encontram dados fragmentados e incompletos).

O *Tô De Olho* aborda esta lacuna ao integrar dados de emendas do Portal da Transparência com informações legislativas do Senado, permitindo que o cidadão visualize, para cada senador: o total de recursos destinados via transferências especiais, os municípios beneficiados e a evolução temporal dos repasses. Ao consolidar informações dispersas em interface única, a plataforma contribui para reduzir a opacidade que caracteriza esta modalidade orçamentária.

2.2 Eixo Político-Participativo

Este eixo investiga as bases teóricas que fundamentam a relação entre tecnologia, participação cidadã e responsabilização de agentes públicos, examinando como ferramentas digitais podem fortalecer — ou apenas simular — o controle social.

2.2.1 Democracia Digital e Participação Cidadã

O conceito de democracia digital refere-se ao emprego de tecnologias de informação e comunicação (TICs) para produzir "mais democracia e melhores democracias" [1]. Wilson

Gomes identifica três fases históricas neste campo: a **teledemocracia** (anos 1970-90), marcada por experimentos com televisão interativa e enquetes eletrônicas; a **fase da internet** (1995-2005), caracterizada pelo debate sobre potenciais e limites da rede para a participação política; e a **autonomização contemporânea**, onde subtemas como governo aberto, *smart cities* e parlamento digital desenvolvem-se de forma independente, com metodologias e agendas próprias [4].

A participação cidadã mediada por tecnologia pode assumir diferentes níveis de profundidade e poder real. Sherry Arnstein, em seu trabalho seminal de 1969, propõe a "Escada da Participação Cidadã" (*Ladder of Citizen Participation*), uma tipologia de oito degraus que classifica o grau de poder efetivamente conferido aos cidadãos [9]:

- **Não-participação** (degraus 1-2): *Manipulação* e *Terapia* — formas em que o objetivo é "educar" ou "curar" os participantes, não ouvi-los. Comitês consultivos sem poder deliberativo exemplificam essa categoria.
- **Participação simbólica** (degraus 3-5): *Informação*, *Consulta* e *Pacificação* — cidadãos podem ouvir e ser ouvidos, mas sem garantia de que suas vozes influenciem decisões. Audiências públicas e pesquisas de opinião situam-se neste nível.
- **Poder cidadão** (degraus 6-8): *Parceria*, *Delegação de Poder* e *Controle Cidadão* — redistribuição efetiva de poder decisório. Orçamentos participativos vinculantes e conselhos com poder de veto exemplificam esses degraus superiores.

Ferramentas de transparência como o *Tô De Olho* situam-se primariamente no degrau da **informação**: proveem ao cidadão dados estruturados sobre a atuação parlamentar, condição necessária — mas não suficiente — para o exercício pleno da fiscalização. Como adverte Arnstein, "participação sem redistribuição de poder é um processo vazio e frustrante para os desprovidos de poder" [9]. Reconhecendo essa limitação estrutural, o projeto busca ir além da mera disponibilização de dados: ao oferecer *rankings*, comparativos e visualizações contextualizadas, a plataforma **empodera** o cidadão para uma fiscalização mais qualificada, fornecendo-lhe ferramentas para exercer pressão informada sobre seus representantes.

No contexto brasileiro, a brecha digital representa obstáculo adicional à democracia digital. Segundo o INAF, 29% da população entre 15 e 64 anos é funcionalmente analfabeta [6], limitando severamente a capacidade de interpretar planilhas, gráficos e relatórios técnicos. Avelino et al. mapeiam iniciativas de governo aberto no âmbito federal, identificando avanços significativos na disponibilização de dados, porém alertando que tecnologias precisam ser mediadas por visualizações claras e linguagem acessível para efetivação do controle social [2].

2.2.2 Teoria Principal-Agente e Accountability

A relação entre cidadãos e representantes eleitos pode ser analisada através da lente da **teoria econômica de agência**, originalmente desenvolvida para compreender relações contratuais em organizações. Neste modelo, os cidadãos (eleitores) atuam como **principais** que delegam autoridade a **agentes** (parlamentares e burocratas) para tomar decisões em seu

nome. O problema fundamental surge da **assimetria informacional**: os agentes possuem mais informação sobre suas próprias ações, esforços e competências do que os principais que os monitoram [24].

Esta assimetria manifesta-se de duas formas principais. O **risco moral** (*moral hazard*) ocorre quando os principais não conseguem observar plenamente as ações dos agentes após a delegação: um parlamentar pode "enrolar" em suas funções, ausentar-se de votações ou priorizar interesses particulares sem que o eleitorado perceba. A **seleção adversa** (*adverse selection*) surge antes da delegação: candidatos podem exagerar suas qualificações ou ocultar incompetências durante campanhas eleitorais. Em ambos os casos, a divergência entre os interesses do principal e do agente gera "custos de agência" — ineficiências, corrupção ou políticas distorcidas.

O conceito de **accountability** — frequentemente traduzido como "prestação de contas" ou "responsabilização" — emerge como mecanismo para mitigar esses problemas. A literatura distingue três vertentes complementares [5]:

- **Accountability vertical**: Exercida pelos eleitores através do voto. Parlamentares que não atendem às expectativas podem ser punidos nas urnas. Contudo, eleições são instrumentos "grosseiros": ocorrem periodicamente (a cada 4-8 anos para senadores), envolvem múltiplas dimensões de avaliação simultâneas, e dependem de que o eleitor tenha informação suficiente para julgar o desempenho.

- **Accountability horizontal**: Exercida por instituições de controle como tribunais de contas, controladorias, Ministério Público e o próprio Poder Judiciário. Esses órgãos possuem competência técnica e acesso privilegiado a informações, mas enfrentam limitações de capacidade operacional diante do volume de atos a fiscalizar.

- **Accountability social**: Exercida por organizações da sociedade civil, imprensa investigativa, acadêmicos e cidadãos organizados. Esta modalidade complementa as anteriores ao ampliar a capacidade de monitoramento e pressionar por transparência.

A transparência é condição necessária — mas não suficiente — para a accountability. Pesquisas demonstram que para a divulgação de informações traduzir-se em responsabilização efetiva, três condições devem ser satisfeitas: (1) a informação deve efetivamente alcançar o público relevante; (2) o público deve ter capacidade de processá-la e reagir a ela; e (3) devem existir mecanismos institucionais que permitam consequências para os agentes. A simples disponibilização de dados brutos — o que alguns autores chamam de "governo nu" (*naked government*) — pode até amplificar percepções negativas sem gerar accountability efetiva.

O *Tô De Olho* posiciona-se como ferramenta de **accountability social**, reduzindo a assimetria informacional entre eleitores e senadores. Ao consolidar dados fragmentados, contextualizar valores com médias comparativas e oferecer rankings metodologicamente fundamentados, a plataforma amplia a capacidade do cidadão de monitorar seus representantes — condição prévia para o exercício tanto da accountability vertical (voto informado) quanto da pressão por accountability horizontal (denúncias fundamentadas a órgãos de controle).

2.2.3 Civic Tech e Sociedade Civil

O termo **civic tech** (tecnologia cívica) refere-se ao uso de tecnologias digitais para fortalecer a participação cidadã, a transparência governamental e a colaboração entre sociedade e Estado. Diferencia-se de *GovTech* (tecnologia para eficiência governamental interna) por seu foco na interface com o cidadão e no empoderamento da sociedade civil. Lathrop e Ruma, em coletânea seminal sobre governo aberto, reúnem contribuições de pioneiros como Tim O'Reilly, Beth Noveck e Aaron Swartz, estabelecendo três pilares para a *civic tech*: **transparência** (dados abertos), **participação** (engajamento cidadão) e **colaboração** (parcerias entre governo e sociedade) [25].

No Brasil, o ecossistema de civic tech consolidou-se a partir de 2010, impulsionado pela Lei de Acesso à Informação (2011) e pelo crescimento de organizações especializadas. Entre as iniciativas mais relevantes:

- **Transparência Brasil**: Fundada em 2000, é uma das principais organizações dedicadas à promoção da integridade e supervisão cívica no setor público. Desenvolve pesquisas, relatórios e ferramentas para monitoramento de políticas públicas [26].
- **Open Knowledge Brasil**: Responsável pela Operação Serenata de Amor e pelo projeto "Querido Diário", que aplica técnicas de inteligência artificial para auditar diários oficiais municipais.
- **Fiquem Sabendo**: Organização sem fins lucrativos que utiliza civic tech de código aberto para expor gastos governamentais não divulgados, tendo revelado mais de 500 bilhões de reais em despesas não reportadas ao longo de 27 anos.
- **Associação Brasileira de Jornalismo Investigativo (Abraji)**: Emprega jornalismo de dados e ferramentas tecnológicas para investigar corrupção e má gestão de recursos públicos.

Apesar dos avanços, o ecossistema enfrenta desafios significativos. A **sustentabilidade financeira** é precária: muitas iniciativas dependem de financiamento coletivo, bolsas internacionais ou trabalho voluntário, limitando sua capacidade de operação contínua. A **brecha digital** restringe o alcance a populações com menor acesso à internet ou habilidades tecnológicas limitadas. E a **fragmentação de esforços** — múltiplas ferramentas com escopos parcialmente sobrepostos — dispersa recursos e dificulta o engajamento do cidadão comum.

O *Tô De Olho* posiciona-se neste ecossistema com um diferencial claro: o foco exclusivo no **Senado Federal**, câmara legislativa até então carente de ferramentas específicas de fiscalização cidadã. Enquanto a Câmara dos Deputados conta com ao menos três plataformas consolidadas (Serenata de Amor, De Olho no Congresso, De Olho em Você), os 81 senadores — que exercem mandatos de oito anos e detêm competências exclusivas de alto impacto — permanecem em relativa "sombra" digital.

2.3 Eixo Técnico-Metodológico

Este eixo apresenta os fundamentos técnicos e metodológicos que orientam a implementação do *Tô De Olho*, incluindo métricas de avaliação parlamentar, princípios de visualização de dados e decisões arquiteturais.

2.3.1 Métricas de Efetividade Legislativa

A avaliação quantitativa do desempenho parlamentar constitui tema relevante na ciência política contemporânea. Craig Volden e Alan E. Wiseman, co-diretores do *Center for Effective Lawmaking*, desenvolveram o *Legislative Effectiveness Score* (LES), uma métrica que mensura a capacidade de parlamentares em conduzir suas proposições através do processo legislativo [27, 7].

A metodologia do LES fundamenta-se em cinco estágios do processo legislativo, cada qual representando um grau crescente de sucesso na agenda do parlamentar:

1. **Introdução**: O projeto é formalmente apresentado;
2. **Ação em comissão**: O projeto recebe parecer ou é debatido em comissão temática;
3. **Votação em plenário (câmara de origem)**: O projeto é levado à votação na casa onde foi apresentado;
4. **Aprovação na câmara de origem**: O projeto é aprovado e segue para a outra casa;
5. **Conversão em lei**: O projeto completa a tramitação e é sancionado.

Crucialmente, nem todos os projetos possuem igual peso na metodologia. Volden e Wiseman categorizam as proposições em três níveis de significância: **projetos comemorativos** (como denominação de logradouros), que recebem peso mínimo; **projetos substantivos**, que alteram políticas públicas de forma moderada; e **projetos substantivos e significativos**, que promovem mudanças estruturais relevantes. Esta ponderação evita que parlamentares "inflem" suas estatísticas com proposições triviais [28].

Os autores identificaram fatores consistentemente correlacionados à maior efetividade legislativa: senioridade no mandato, ocupação de posições em comissões estratégicas (especialmente presidências e relatorias), pertencimento ao partido majoritário e experiência prévia em legislaturas estaduais. Interessantemente, pesquisas subsequentes demonstraram que, controlando demais variáveis, parlamentares mulheres tendem a ser mais efetivas que seus colegas homens [7].

Embora desenvolvida para o contexto norte-americano, a metodologia oferece um *framework* adaptável para avaliar parlamentares brasileiros. No *Tô De Olho*, o "Score" do senador inspira-se nesta abordagem, combinando quatro dimensões com pesos públicos:

- **Produtividade Legislativa (35%)**: Avalia proposições de autoria e relatorias, com multiplicadores por tipo (PEC: 3x, PLP: 2x) e estágio de tramitação alcançado;

- **Presença em Votações (25%)**: Mensura comparecimento efetivo às sessões deliberativas;
- **Economia na Cota Parlamentar (20%)**: Compara uso individual da CEAPS com a mediana do Senado;
- **Participação em Comissões (20%)**: Pondera engajamento em comissões, com bônus para cargos de liderança.

A transparência metodológica — expor publicamente os critérios e pesos utilizados — é fundamental para que o *ranking* seja percebido como ferramenta de informação, não de manipulação política. Diferente de classificações opacas, o cidadão poderá compreender — e questionar — os fundamentos da avaliação.

2.3.2 Visualização de Dados e Retórica Visual

A apresentação de dados ao cidadão não é neutra: escolhas de *design* influenciam profundamente a interpretação das informações. Jessica Hullman e Nicholas Diakopoulos investigaram os "efeitos de enquadramento" (*framing effects*) em visualizações narrativas, demonstrando que técnicas retóricas como seleção, omissão, ênfase e sequenciamento podem direcionar a leitura do público de forma consciente ou inconsciente [8].

Os autores identificam quatro categorias de técnicas retóricas em visualizações de dados:

1. **Proveniência**: Identificação da origem e credibilidade dos dados. Visualizações que ocultam fontes ou datas de atualização comprometem a verificabilidade. No *Tô De Olho*, cada gráfico exibe a fonte oficial (API do Senado ou Portal da Transparência) e a data da última sincronização.
2. **Mapeamento visual**: Como elementos gráficos representam variáveis numéricas. Escalas inconsistentes, truncamento de eixos ou escolhas de cores podem distorcer percepções. A plataforma adota escalas consistentes em gráficos comparativos e paletas de cores acessíveis.
3. **Anotações linguísticas**: Textos, títulos e legendas que guiam a interpretação. Valores absolutos sem contexto podem induzir conclusões equivocadas (ex: "Senador X gastou R$ 100 mil" parece muito sem saber que a média é R$ 150 mil). O *Tô De Olho* contextualiza valores com médias e percentis.
4. **Interatividade**: Controles que permitem ao usuário explorar dados por conta própria reduzem a dependência de narrativas pré-construídas. A plataforma oferece filtros por partido, estado e período, permitindo análises personalizadas.

A **literacia em visualização de dados** (*Data Visualization Literacy* — DVL) refere-se à capacidade de interpretar corretamente representações visuais de informações. Pesquisas demonstram que mesmo populações com alta escolaridade frequentemente cometem erros de interpretação em gráficos aparentemente simples. No contexto brasileiro, onde 29% da população é funcionalmente analfabeta [6], o desafio é ainda maior: visualizações complexas

podem excluir justamente os cidadãos mais vulneráveis à falta de transparência.

Para o *Tô De Olho*, esses princípios orientam decisões de *design*: priorizar visualizações simples e intuitivas; oferecer múltiplas formas de apresentação (gráficos, tabelas, textos explicativos); e testar a compreensibilidade com usuários de diferentes perfis. O objetivo é maximizar a transparência metodológica, evitando que a plataforma seja percebida como veículo de viés político.

2.3.3 Arquitetura de Software: Monolito Modular

A escolha arquitetural de um sistema de software envolve *trade-offs* fundamentais entre complexidade operacional, velocidade de desenvolvimento e capacidade de evolução. A literatura distingue três abordagens principais: o **monolito tradicional**, onde todos os componentes residem em uma única base de código sem separação clara; os **microsserviços**, que fragmentam a aplicação em serviços independentes comunicando-se via rede; e o **monolito modular**, que combina a simplicidade operacional do primeiro com a organização interna do segundo [29, 30].

O monolito modular organiza o código em módulos discretos, fracamente acoplados e orientados a domínio, porém dentro de uma única unidade de *deploy*. Diferente do monolito tradicional, que tende a se tornar uma "bola de lama" (*big ball of mud*) com o crescimento, o monolito modular impõe fronteiras lógicas explícitas entre componentes, facilitando a manutenção e permitindo que equipes trabalhem de forma relativamente independente em diferentes partes do sistema [30].

A escolha por monolito modular no *Tô De Olho* fundamenta-se em critérios técnicos e contextuais:

- **Simplicidade de Deploy**: Um único contêiner Docker simplifica a infraestrutura e reduz custos operacionais. Plataformas *serverless* como Google Cloud Run beneficiam-se particularmente desta característica, oferecendo escala automática (inclusive a zero) sem a complexidade de orquestração de múltiplos serviços;

- **Ausência de Latência de Rede**: A comunicação entre módulos ocorre via chamadas de função em memória, eliminando a latência e os riscos de falha associados a chamadas de rede. Em microsserviços, cada interação entre componentes adiciona overhead de serialização, transmissão e desserialização;

- **Consistência Transacional**: Operações que envolvem múltiplos módulos podem compartilhar uma única transação de banco de dados, garantindo atomicidade. Em arquiteturas distribuídas, esse padrão requereria implementação de *sagas* ou *two-phase commit*, aumentando significativamente a complexidade;

- **Adequação ao Contexto**: Para equipes pequenas e projetos acadêmicos, a complexidade operacional de microsserviços — monitoramento distribuído, orquestração de contêineres, *service discovery*, gerenciamento de configurações — frequentemente supera os benefícios de escalabilidade granular.

A literatura recente sobre monolito modular destaca a importância de princípios do **Domain-Driven Design** (DDD) na definição de fronteiras entre módulos. Eric Evans, criador do DDD, propõe o conceito de *Bounded Contexts* — regiões do sistema onde um modelo de domínio específico se aplica, com interfaces bem definidas para comunicação com outros contextos [31]. No *Tô De Olho*, cada módulo interno (`senador`, `ceaps`, `votacao`, `emenda`, `ranking`) representa um *bounded context* com responsabilidades claras.

Frameworks modernos têm formalizado o padrão de monolito modular. O Google Service Weaver permite escrever aplicações como monolitos modulares que podem ser implantados como processos únicos ou microsserviços distribuídos, sem alteração de código [32]. O Spring Modulith oferece suporte similar no ecossistema Java, com verificação automatizada de fronteiras entre módulos [33]. Embora o *Tô De Olho* não utilize esses frameworks específicos (dado o backend em Go), a arquitetura interna segue os mesmos princípios: módulos com interfaces explícitas, dependências unidirecionais e comunicação via contratos bem definidos.

Esta escolha não descarta a possibilidade de evolução futura. Caso o sistema alcance escala que justifique escalabilidade granular de componentes específicos, a estrutura modular facilitará a extração de microsserviços independentes — uma estratégia de "decomposição incremental" recomendada pela literatura como alternativa ao risco de migração completa [29].

2.3.4 Engenharia de Dados: APIs e Processos ETL

O padrão ETL (*Extract, Transform, Load*) constitui a espinha dorsal de sistemas de integração de dados desde os primórdios da computação analítica. Ralph Kimball, pioneiro em *data warehousing*, define ETL como o processo responsável por "extrair dados de sistemas fonte, aplicar transformações de qualidade e conformidade, e entregar os dados em formato dimensional" [34]. Vassiliadis, em revisão abrangente da literatura, estima que processos ETL consomem entre 60% e 80% do esforço de desenvolvimento de projetos de integração de dados [35].

A estratégia de ingestão do *Tô De Olho* segue as três fases clássicas do ETL:

1. **Extração (Extract)**: Coleta de dados brutos das fontes oficiais via APIs RESTful. O sistema consome três APIs distintas: a API Legislativa do Senado (matérias, votações nominais, comissões), a API Administrativa do Senado (CEAPS, remunerações de gabinete) e a API do Portal da Transparência (emendas parlamentares). Cada fonte possui características próprias de paginação, limites de requisição e formatos de resposta, exigindo adaptadores específicos;

2. **Transformação (Transform)**: Normalização, limpeza e enriquecimento dos dados extraídos. Esta fase inclui: padronização de nomes de senadores (tratamento de variações como "José da Silva" vs "JOSE DA SILVA"), conversão de formatos de data, cálculo de métricas derivadas (como percentuais de presença) e detecção de anomalias (valores negativos, datas futuras, registros duplicados);

3. **Carga (Load)**: Persistência dos dados transformados no banco PostgreSQL, com es-

tratégias de upsert para evitar duplicação e manutenção de histórico de alterações.

O processo combina duas estratégias complementares de carga:

- **Backfill (Carga Histórica)**: Execução única para ingestão do histórico completo desde um ano configurável (padrão: 2019). Esta flexibilidade, implementada via variável de ambiente, permite ajustar o escopo temporal conforme necessidades específicas — por exemplo, restringir a dados de 2023 em ambiente de desenvolvimento para acelerar testes, ou expandir para 2015 para análises de longo prazo;
- **Sincronização Incremental**: Tarefas agendadas (*CronJobs*) executadas diariamente às 03:00 BRT, capturando apenas atualizações desde a última sincronização. Esta abordagem otimiza o consumo de recursos e respeita os limites de requisição das APIs oficiais.

A escolha por APIs oficiais — em detrimento de técnicas de *web scraping* — fundamenta-se em critérios de confiabilidade e manutenibilidade. APIs possuem contratos documentados, formatos estruturados (JSON/XML) e versionamento explícito, enquanto o *scraping* de páginas HTML é intrinsecamente frágil a mudanças de layout. Kimball adverte que "a qualidade dos dados de entrada determina o teto de qualidade do sistema final" [36] — premissa que reforça a preferência por fontes oficiais com garantias de integridade.

Adicionalmente, as APIs oficiais brasileiras oferecem vantagens práticas: o Portal da Transparência disponibiliza dados de emendas com granularidade diária; a API Legislativa do Senado permite consultas por período específico; e a API Administrativa provê arquivos CSV consolidados para carga em massa. Essas características distintas exigem que o módulo ETL do *Tô De Olho* implemente adaptadores especializados para cada fonte, unificando os dados em um modelo dimensional coerente antes da carga.

3 Metodologia

O desenvolvimento do *Tô De Olho* enquadra-se no paradigma da *Design Science Research* (DSR), abordagem metodológica adequada para pesquisas que visam a construção e avaliação de artefatos tecnológicos destinados a resolver problemas organizacionais identificados [37]. O problema abordado — a fragmentação de dados públicos sobre a atuação de senadores federais em múltiplas APIs governamentais não integradas — demanda a construção de um artefato de software capaz de consolidar, processar e apresentar essas informações de forma acessível ao cidadão.

Esta seção detalha a abordagem de desenvolvimento iterativo adotada, as fontes de dados governamentais integradas, a arquitetura do sistema, a estratégia de ingestão via ETL, o algoritmo de ranking inspirado no *Legislative Effectiveness Score* e a infraestrutura de implantação.

3.1 Abordagem de Desenvolvimento

A natureza do projeto — integração de múltiplas APIs governamentais com estruturas de dados heterogêneas e documentação variável — demandou uma abordagem de desenvolvi-

mento capaz de acomodar descobertas incrementais e ajustes frequentes de escopo. Metodologias tradicionais de desenvolvimento em cascata, que pressupõem requisitos estáveis e bem definidos desde o início, mostraram-se inadequadas para este contexto de exploração de APIs públicas com comportamentos nem sempre previsíveis.

Optou-se, portanto, por uma abordagem **iterativa e incremental**, na qual o trabalho foi organizado em ciclos de desenvolvimento focados em entregas funcionais. Cada ciclo produzia um incremento utilizável do sistema, permitindo validação contínua das funcionalidades implementadas e ajustes baseados nos aprendizados obtidos durante a integração com cada API.

O desenvolvimento contou com apoio de ferramentas de inteligência artificial generativa como assistentes de codificação, seguindo a tendência contemporânea de *AI-assisted software development* [38]. Tais ferramentas, baseadas em modelos de linguagem de grande escala (LLMs), foram empregadas para aceleração de tarefas operacionais como geração de código *boilerplate*, refatoração e *debugging*. O uso de assistentes de IA em desenvolvimento de software representa uma evolução natural das ferramentas de produtividade, análoga à adoção de IDEs com *autocomplete* e analisadores estáticos de código.

É fundamental distinguir entre **assistência operacional** e **autoria intelectual**. As decisões estruturantes do projeto — arquitetura do sistema, design do algoritmo de ranking, seleção de fontes de dados, modelagem do domínio e interpretação de resultados — foram integralmente concebidas, avaliadas e validadas pelo desenvolvedor. A ferramenta de IA atuou como acelerador de implementação, não como substituto do julgamento técnico. A engenharia de contexto — técnica de estruturação de informações em arquivos de configuração que definem padrões de codificação, stack tecnológico e regras de negócio — foi aplicada para maximizar a aderência das sugestões aos padrões do projeto, mantendo consistência arquitetural ao longo do desenvolvimento.

A divisão do trabalho ocorreu em cinco fases principais:

1. **Fundação**: Estruturação do projeto em Golang, implementação do cliente para a API Legislativa do Senado, criação das *migrations* do banco de dados e configuração inicial do *frontend* em Next.js;

2. **Ingestão de Dados**: Implementação do cliente para a API Administrativa, configuração do *scheduler* para tarefas agendadas, coleta de votações nominais e carga de dados históricos;

3. **Ranking e API**: Desenvolvimento do serviço de cálculo de rankings, criação dos *endpoints* REST para consumo pelo *frontend*, configuração do cache Redis e implementação de testes automatizados;

4. **Frontend**: Desenvolvimento do *dashboard* principal, interface de ranking interativo e páginas de perfil dos senadores;

5. **Emendas e Polimento**: Integração com o Portal da Transparência para dados de emendas parlamentares, visualizações de dados e preparação para *deploy*.

3.2 Fontes de Dados

O sistema integra três fontes de dados governamentais oficiais, fundamentadas no arcabouço legal brasileiro de transparência pública. A Lei de Acesso à Informação (Lei n. 12.527/2011) estabelece como diretriz a "disponibilização de informações em formatos abertos, estruturados e legíveis por máquina" [11], princípio que as APIs governamentais operacionalizam. Conforme demonstrado pela Operação Serenata de Amor, tecnologias desenvolvidas sobre esses dados abertos podem gerar valor público ao facilitar o controle social do gasto parlamentar [14].

A seleção dos *endpoints* e campos consumidos seguiu três critérios: (i) **relevância para fiscalização cidadã** — priorizando dados de gastos, votações e atuação legislativa; (ii) **disponibilidade e confiabilidade** — selecionando fontes com documentação oficial e atualização regular; e (iii) **viabilidade técnica** — considerando formatos estruturados (JSON/XML), autenticação simples e limites de requisição adequados.

3.2.1 API Legislativa do Senado

Documentada em ⟨legis.senado.leg.br/dadosabertos⟩, esta API RESTful fornece dados do processo legislativo. A URL base para requisições é ⟨https://legis.senado.leg.br/dadosabertos⟩. Os dados são retornados em formato JSON, com suporte a paginação e limite de 10 requisições por segundo.

Endpoints de Senadores e Mandatos:

- `/dadosabertos/senador/lista/atual`: Lista de senadores em exercício, retornando código parlamentar, nome, foto, partido e UF;
- `/dadosabertos/senador/{codigo}`: Detalhes completos do senador, incluindo biografia, e-mail e telefone;
- `/dadosabertos/senador/{codigo}/mandatos`: Histórico de mandatos com legislatura, tipo e datas;
- `/dadosabertos/senador/{codigo}/licencas`: Licenças oficiais com motivo e período — essencial para ajuste dos cálculos de presença.

Endpoints de Votações:

- `/dadosabertos/votacao`: Votações nominais filtráveis por código do parlamentar, data e orientação de voto;
- `/dadosabertos/votacaoComissao/parlamentar/{codigo}`: Votações em comissões específicas;
- `/dadosabertos/plenario/votacao/orientacaoBancada/{data}`: Orientação partidária para cálculo de fidelidade.

Endpoints de Atuação Legislativa:

- `/dadosabertos/processo?codigoParlamentarAutor={codigo}`: Proposições de autoria do senador;
- `/dadosabertos/processo/relatoria?codigoParlamentar={codigo}`: Relatorias designadas;
- `/dadosabertos/senador/{codigo}/comissoes`: Participação em comissões;
- `/dadosabertos/senador/{codigo}/discursos`: Discursos proferidos.

3.2.2 API Administrativa do Senado

Documentada em ⟨adm.senado.gov.br/adm-dadosabertos/swagger-ui⟩, esta interface disponibiliza dados financeiros e administrativos. A URL base para requisições é ⟨https://adm.senado.gov.br/adm-dadosabertos⟩.

Cota Parlamentar (CEAPS):

- `/api/v1/senadores/despesas_ceaps/{ano}`: Despesas da Cota para o Exercício da Atividade Parlamentar dos Senadores, retornando código do senador, tipo de despesa, fornecedor, CPF/CNPJ, valor reembolsado e mês.

Estrutura e Benefícios:

- `/api/v1/senadores/auxilio-moradia`: Opção por auxílio-moradia;
- `/api/v1/senadores/escritorios`: Escritórios de apoio parlamentar.

Servidores de Gabinete:

- `/api/v1/servidores/servidores`: Lista de servidores com filtros por lotação;
- `/api/v1/servidores/remuneracoes/{ano}/{mes}`: Folha de pagamento mensal, retornando remuneração básica e líquida;
- `/api/v1/servidores/lotacoes`: Mapeamento entre senador e gabinete.

3.2.3 Portal da Transparência (CGU)

A API do Portal da Transparência (⟨api.portaldatransparencia.gov.br⟩) fornece dados de emendas parlamentares e transferências federais. O acesso requer autenticação via chave de API no *header* `chave-api-dados`.

Endpoint Principal:

- `/api-de-dados/emendas`: Lista de emendas parlamentares com parâmetros de filtro: `ano` (ano fiscal), `nomeAutor` (nome do parlamentar), `tipoEmenda` (Individual, Bancada, Comissão, Relator ou **Transferência Especial**).

O filtro `tipoEmenda=Transferência Especial` permite identificar as chamadas “emendas PIX”, modalidade de repasse que dispensa convênio. Os campos retornados incluem código

da emenda, autor, localidade do gasto, função, subfunção e valores (empenhado, liquidado e pago).

3.2.4 Limitações e Cobertura Temporal

Cada fonte de dados apresenta limitações que impactam o escopo e a profundidade das análises possíveis:

- **API Legislativa**: Votações nominais disponíveis apenas a partir de 2019 (legislatura 56); dados anteriores requerem consulta a arquivos históricos em formato XML. Limite de 10 requisições por segundo;
- **API Administrativa**: Despesas CEAPS disponíveis desde 2008 em CSV; API REST cobre apenas o ano corrente. Não há *endpoint* para vincular servidores diretamente ao gabinete do senador — essa associação requer cruzamento com a tabela de lotações;
- **Portal da Transparência**: Emendas disponíveis a partir de 2015. Busca por autor utiliza correspondência textual (nome), não código parlamentar, exigindo normalização de grafia. Limite de 300 requisições por minuto com autenticação via chave de API.

Essas restrições foram consideradas no desenho do sistema: dados históricos são carregados via arquivos CSV (backfill), enquanto a sincronização contínua utiliza as APIs REST com tratamento adequado de limites de requisição.

3.3 Estratégia de Ingestão de Dados

A ingestão de dados representa um dos maiores desafios técnicos do projeto: consolidar informações dispersas em três APIs governamentais distintas, cada uma com suas idiossincrasias, limites de requisição e formatos de resposta. A estratégia adotada segue o paradigma *Extract, Transform, Load* (ETL) em uma abordagem híbrida que combina carga inicial massiva (*backfill*) com atualização incremental contínua.

Conforme fundamentado teoricamente na Seção 2.3.4, o processo ETL tipicamente consome entre 60% e 80% do esforço de desenvolvimento em projetos de *data warehousing* [35]. Essa proporção reflete a complexidade inerente à extração de dados de fontes heterogêneas e à necessidade de transformações para garantir consistência e qualidade. A presente seção detalha a implementação específica adotada no *Tô De Olho*, conforme ilustrado na Tabela 1.

Tabela 1: Estratégias de ingestão de dados: Backfill vs. Sincronização Contínua

Característica	Backfill (Carga Inicial)	Sincronização Contínua
Objetivo	Popular o banco com dados históricos	Manter dados atualizados
Execução	Uma única vez, no setup do sistema	Diariamente, via cron jobs
Volume de dados	Grande (anos de histórico)	Pequeno (delta diário)
Tempo de execução	Horas (processamento em lote)	Minutos ($<$5 min)
Fonte preferencial	CSVs para download em massa	APIs REST incrementais
Exemplo	CEAPS 2019–2025 via CSV	Votações dos últimos 7 dias

3.3.1 Backfill (Carga Histórica)

A carga inicial (*Backfill*) distingue-se fundamentalmente da sincronização diária pelo volume e pela estratégia de escrita. Conforme preceituado por Kimball e Ross [39], operações de carga histórica devem privilegiar o *throughput* (vazão) em detrimento da latência individual. Esta recomendação reflete décadas de experiência prática: tentar carregar milhões de registros através de inserções unitárias resulta em tempos de execução proibitivos e pressão desnecessária sobre as APIs de origem.

Para este projeto, optou-se pela estratégia de **Backfill em Lote Isolado**. Ao invés de reutilizar os *endpoints* REST — projetados para transações unitárias e limitados por *rate limiting* rigoroso — o sistema consome arquivos CSV consolidados disponibilizados pelas fontes primárias. Esta abordagem reduz o *overhead* de rede e permite o uso de `COPY` ou *bulk inserts* no banco de dados, operações que alcançam taxas de inserção na ordem de dezenas de milhares de registros por segundo, ordens de magnitude superiores a inserções linha a linha.

A distinção entre carga inicial e sincronização contínua não é meramente operacional, mas também arquitetural. O processo de *backfill* executa em ambiente isolado, podendo utilizar conexões de banco com configurações agressivas (*batch size* elevado, desativação temporária de índices não-essenciais) sem impactar a disponibilidade do sistema em produção. As fontes de dados para carga inicial incluem:

- **CEAPS**: Arquivos CSV anuais (2008–2025) disponíveis via *endpoint* de dados abertos estáticos do Senado, totalizando aproximadamente 500 mil registros de despesas;
- **Votações**: Iteração controlada na API Legislativa por ano/legislatura (2019–Presente), respeitando limites de 10 requisições por segundo;
- **Emendas**: *Dump* consolidado do Portal da Transparência, com dados desde 2015.

3.3.2 Sincronização Contínua (Change Data Capture)

Uma vez que os dados históricos foram carregados, o desafio seguinte consiste em mantê-los atualizados de forma eficiente. Idealmente, as APIs governamentais ofereceriam mecanismos de *push* — como *Webhooks* ou fluxos de eventos (*Streaming*) — permitindo

que o sistema fosse notificado proativamente sobre mudanças. Entretanto, nenhuma das três fontes de dados integradas disponibiliza tais recursos, uma limitação comum em APIs governamentais brasileiras.

Diante dessa restrição, adotou-se o padrão de **Polling Periódico Inteligente**, no qual o sistema atua como agente ativo que consulta periodicamente as fontes em busca de alterações (*Pull Model*). Embora menos eficiente que abordagens baseadas em eventos, o *polling* oferece vantagens significativas para integração com sistemas legados: simplicidade de implementação, ausência de dependências de infraestrutura complexa (como *message brokers*) e compatibilidade garantida com qualquer API que suporte requisições HTTP.

A periodicidade dos *cron jobs* foi calibrada com base na **Janela de Consistência Eventual** aceitável para cada domínio de dado. O conceito de consistência eventual, originário de bancos de dados distribuídos, reconhece que em sistemas descentralizados, os dados podem estar temporariamente desatualizados desde que eventualmente convirjam para o estado correto. Para um portal de fiscalização cidadã, a precisão ao segundo não é necessária; o que importa é que os dados reflitam a realidade em um horizonte razoável:

- **Alta Volatilidade (Diário)**: Status de senadores, presença e votações do dia anterior. A execução ocorre às 03:00 (horário de Brasília), período de baixa atividade nos servidores governamentais, capturando o fechamento completo do dia legislativo;
- **Média Volatilidade (Semanal)**: Remuneração de servidores e mapeamento de lotações. A folha de pagamento do Senado é atualizada mensalmente, mas a execução semanal (domingos às 04:00) garante margem de segurança para capturar eventuais correções;
- **Baixa Volatilidade (Mensal)**: Consolidação de emendas parlamentares. Os dados de execução orçamentária do Portal da Transparência são consolidados mensalmente, justificando a execução no dia 05 de cada mês, após o fechamento contábil público.

Esta estratégia de *polling* respeita os princípios de "Cidadania de API" (*API Good Citizenship*), concentrando requisições em horários de baixo tráfego para minimizar impacto nos servidores governamentais e maximizar a taxa de sucesso das chamadas.

3.3.3 Idempotência e Consistência Eventual

Em sistemas distribuídos sujeitos a falhas de rede, a garantia de entrega *exactly-once* é virtualmente impossível de alcançar na prática [40]. Qualquer comunicação entre componentes pode falhar após o processamento mas antes da confirmação, deixando o cliente em estado de incerteza. Portanto, o sistema foi projetado para operar sob o modelo *at-least-once*, no qual operações podem ser repetidas sem efeitos colaterais indesejados, apoiado pela propriedade de **Idempotência**.

Matematicamente, uma operação de ingestão $f(x)$ é idempotente se $f(f(x)) = f(x)$. Ou seja, aplicar a função múltiplas vezes produz o mesmo resultado que aplicá-la uma única vez. Esta propriedade, originária da álgebra abstrata, encontra aplicação direta em operações de banco de dados: uma inserção `INSERT OR UPDATE` (*upsert*) com chave primária definida é

idempotente, pois tentativas subsequentes simplesmente sobrescrevem o registro existente com valores idênticos.

A implementação de idempotência no *Tô De Olho* utiliza chaves naturais compostas (*Composite Natural Keys*) em vez de identificadores gerados. Esta escolha reflete uma decisão arquitetural deliberada: os sistemas de origem (APIs do Senado e Portal da Transparência) são a fonte de verdade para identificação de entidades. As operações de *upsert* garantem:

- **Despesas CEAPS**: Unicidade garantida pela tupla `(senador_id, fornecedor_cnpj, data_emissao, valor_centavos)`. Esta combinação identifica univocamente cada reembolso, considerando que o mesmo senador não pode receber dois reembolsos idênticos do mesmo fornecedor no mesmo dia;
- **Votações**: Unicidade via `(id_sessao, id_parlamentar)`. Cada senador registra no máximo um voto por sessão de votação;
- **Emendas**: Unicidade via `(codigo_emenda, autor, ano)`. O código de emenda é único por autor e ano fiscal.

Essa abordagem elimina a necessidade de gerenciamento de estado complexo no *pipeline* de ingestão: em caso de dúvida, falha de rede ou interrupção inesperada, o sistema pode simplesmente reprocessar o lote inteiro, garantindo convergência para o estado correto sem risco de duplicação de dados.

3.3.4 Resiliência e Tratamento de Erros

A integração com APIs governamentais apresenta desafios de estabilidade: diferentemente de serviços comerciais com *SLAs* bem definidos, as APIs públicas brasileiras operam sem garantias formais de disponibilidade. Para lidar com falhas transientes, o sistema implementa mecanismos simples e eficazes:

- **Retry com Backoff Exponencial**: Para erros de rede ou respostas 5xx, o sistema aguarda progressivamente mais tempo entre tentativas. A fórmula segue o padrão recomendado pela AWS [41]: $t_{wait} = base \times 2^{tentativa}$, com $base = 1s$ e máximo de 3 tentativas. Esta abordagem evita sobrecarregar APIs temporariamente indisponíveis;
- **Timeout Configurável**: Cada requisição possui timeout de 30 segundos, evitando que conexões travadas bloqueiem o processo de ingestão;
- **Falha Graciosa**: Quando uma fonte de dados está indisponível, o job registra o erro e prossegue com as demais fontes. Jobs que falham são marcados para reexecução manual.

Para cenários de produção com maior volume de requisições, padrões como *Circuit Breaker* [42] poderiam ser adicionados para isolar falhas em cascata. Entretanto, dado o escopo do MVP — com execuções diárias em horário de baixo tráfego — a estratégia de retry simples atende adequadamente às necessidades.

3.3.5 Logs e Monitoramento

O monitoramento do *pipeline* de ingestão adota uma abordagem pragmática, adequada ao escopo do projeto:

- **Logs Estruturados**: Todos os eventos de ingestão são emitidos em formato JSON, contendo: identificador do job, fonte de dados, duração, quantidade de registros processados e eventual mensagem de erro. Esta estruturação facilita a análise posterior e permite identificar rapidamente a causa de falhas;
- **Endpoint de Health**: O sistema expõe um *endpoint* `/health` que retorna o status da última sincronização de cada fonte, permitindo verificar se os dados estão atualizados.

Para evolução futura, a literatura recomenda a adoção dos três pilares da observabilidade — logs, métricas e traces [43]. Ferramentas como Prometheus (métricas) e Jaeger (tracing distribuído) poderiam ser integradas conforme a complexidade operacional aumentar.

3.4 Arquitetura do Sistema

A definição da arquitetura de software constitui uma das decisões mais consequentes no ciclo de desenvolvimento. Uma escolha inadequada impõe custos que se acumulam ao longo do tempo: dívida técnica, dificuldade de evolução e, em casos extremos, a necessidade de reescrever sistemas inteiros. Conforme fundamentado teoricamente na Seção 2.3.3, o padrão de **monolito modular** emergiu na última década como alternativa pragmática que busca equilibrar a simplicidade operacional dos monolitos tradicionais com a organização interna característica de microsserviços [30].

O *Tô De Olho* adota esta arquitetura não por inércia, mas por escolha deliberada após avaliação de alternativas. As subseções seguintes detalham a estrutura organizacional, os padrões de comunicação e o fluxo de dados que materializam essa decisão.

3.4.1 Justificativa Arquitetural

A decisão pelo monolito modular considerou quatro critérios interrelacionados:

- **Simplicidade Operacional**: Um único contêiner Docker simplifica a infraestrutura de implantação. Em plataformas *serverless* como Google Cloud Run, essa característica traduz-se em escala automática (inclusive a zero) sem a complexidade de orquestração de múltiplos serviços — vantagem particularmente relevante para projetos com orçamento limitado e equipe reduzida;
- **Latência Previsível**: A comunicação entre módulos ocorre via chamadas de função em memória, eliminando a variabilidade de latência inerente a chamadas de rede. Em microsserviços, cada interação adiciona *overhead* de serialização, transmissão e desserialização — custos que, embora individualmente pequenos, acumulam-se em operações que atravessam múltiplos serviços;
- **Consistência Transacional**: Operações que envolvem múltiplos módulos compartilham

uma única transação de banco de dados, garantindo atomicidade sem a complexidade de *sagas* ou *two-phase commit*. Por exemplo, a atualização do ranking de um senador após nova ingestão de votações executa atomicamente: ou todos os dados são persistidos, ou nenhum;

- **Estratégia MonolithFirst**: Martin Fowler argumenta que "quase todos os casos de sucesso de microsserviços começaram com um monolito que cresceu grande demais e foi decomposto" [44]. Iniciar com fronteiras modulares bem definidas permite compreender os limites naturais do domínio antes de incorrer na complexidade operacional de sistemas distribuídos. A estrutura atual facilita extração futura de microsserviços apenas onde houver necessidade comprovada (por exemplo, se o módulo de ranking exigir escalabilidade independente).

3.4.2 Organização em Módulos

A estrutura interna segue os princípios de *Bounded Contexts* do Domain-Driven Design [31], onde cada módulo representa um contexto de domínio com responsabilidades claramente delimitadas. O diretório `internal/` do backend Go organiza-se em cinco módulos principais:

- `internal/senador`: Gerencia o ciclo de vida de dados cadastrais dos parlamentares — nome, partido, UF, foto, biografia — e seus mandatos históricos. Este módulo atua como "fonte de verdade" para identificação de senadores, fornecendo identificadores únicos que os demais módulos referenciam;

- `internal/ceaps`: Processa e totaliza despesas da Cota para o Exercício da Atividade Parlamentar dos Senadores. Responsável pela ingestão de arquivos CSV históricos e sincronização incremental via API Administrativa, este módulo implementa detecção de anomalias (valores negativos, datas futuras) e agregações por categoria de despesa;

- `internal/votacao`: Coleta e armazena votações nominais do plenário e comissões. Calcula métricas derivadas como percentual de presença (ajustado por licenças oficiais), fidelidade partidária e padrões de votação;

- `internal/emenda`: Integra dados de emendas parlamentares do Portal da Transparência, com tratamento especial para Transferências Especiais ("emendas PIX"). Normaliza a correspondência textual de autores — desafio não trivial dado que o Portal usa nomes, não códigos parlamentares;

- `internal/ranking`: Orquestra o cálculo de *scores* compostos conforme metodologia detalhada na Seção 3.6. Opera como consumidor dos demais módulos, agregando métricas de produtividade, presença, economia e participação em comissões.

Adicionalmente, módulos transversais fornecem funcionalidades compartilhadas: `internal/api` expõe *endpoints* REST consumidos pelo *frontend*; `internal/scheduler` gerencia tarefas agendadas de sincronização; e `internal/cache` abstrai operações de leitura e escrita no Redis.

3.4.3 Padrões de Comunicação

Os módulos comunicam-se exclusivamente através de **interfaces Go** (contratos), nunca acessando estruturas internas de outros módulos diretamente. Esta restrição, inspirada no princípio de inversão de dependência, permite substituir implementações sem afetar consumidores — por exemplo, trocar o adaptador de cache de Redis para memória local em testes unitários.

Para operações de ingestão que envolvem centenas de requisições HTTP, o sistema emprega **pools de workers** com *goroutines*. O modelo de concorrência de Go, baseado em CSP (*Communicating Sequential Processes*), permite processamento paralelo com footprint de memória reduzido: cada *goroutine* consome aproximadamente 2KB iniciais [45], viabilizando milhares de operações simultâneas em hardware modesto.

O padrão adotado segue a estrutura de "fan-out/fan-in": uma *goroutine* coordenadora distribui tarefas para *N* workers (fan-out), que processam independentemente e enviam resultados para um canal agregador (fan-in). Esta abordagem maximiza utilização de CPU em operações *I/O-bound* como chamadas HTTP, mantendo o código legível e testável.

3.4.4 Fluxo de Dados

O ciclo completo de dados no sistema segue quatro estágios bem definidos:

1. **Ingestão**: Os módulos de domínio (`senador`, `ceaps`, `votacao`, `emenda`) extraem dados das APIs governamentais conforme estratégia detalhada na Seção 3.3. Cada módulo implementa seu próprio adaptador HTTP, respeitando as idiossincrasias de cada fonte (paginação, limites de requisição, formatos de resposta);

2. **Transformação**: Dados brutos são normalizados, validados e enriquecidos antes da persistência. Esta fase inclui padronização de nomes (tratamento de acentos, maiúsculas/minúsculas), conversão de formatos de data e cálculo de métricas derivadas. O GORM, ORM utilizado, abstrai as operações de *upsert* que garantem idempotência;

3. **Agregação**: O módulo de `ranking` consulta os demais módulos para computar *scores* compostos. Resultados são materializados no Redis como estruturas pré-computadas, evitando recálculos a cada requisição. A invalidação de cache ocorre após cada sincronização bem-sucedida;

4. **Apresentação**: A API REST expõe os dados agregados ao *frontend* Next.js. Endpoints seguem convenções RESTful, com respostas paginadas e suporte a filtros por partido, UF e período temporal.

Esta separação em estágios facilita a depuração (cada estágio pode ser testado isoladamente), a evolução (novos módulos integram-se sem alterar existentes) e a observabilidade (logs estruturados identificam claramente em qual estágio ocorreu uma falha).

3.5 Stack Tecnológico

A escolha das tecnologias foi orientada por critérios de desempenho, manutenibilidade e adequação ao domínio do problema. Para cada camada do sistema, foram avaliadas alternativas antes da decisão final, conforme detalhado a seguir.

3.5.1 Backend — Golang

A linguagem Go foi selecionada para o *backend* após avaliação comparativa de três alternativas:

- **Node.js**: Embora ofereça vasto ecossistema e familiaridade com JavaScript, o modelo de concorrência baseado em *event loop* único apresenta limitações para operações CPU-bound. Além disso, a gestão de dependências via npm frequentemente resulta em árvores de módulos complexas;
- **Python (FastAPI/Django)**: Excelente para prototipagem rápida e possui bibliotecas maduras para análise de dados. Entretanto, o *Global Interpreter Lock* (GIL) limita o paralelismo real, e o desempenho em operações I/O-bound é inferior ao de linguagens compiladas;
- **Go**: Binários compilados estaticamente, modelo de concorrência nativo e tipagem estática. O fator decisivo foi o modelo de **goroutines**: threads leves gerenciadas pelo runtime Go, consumindo aproximadamente 2KB de memória inicial — em contraste com threads do sistema operacional que utilizam cerca de 1MB cada [45].

Essa eficiência permite processar centenas de milhares de conexões simultâneas, característica essencial para a ingestão paralela de dados de três APIs distintas com limites de requisição distintos.

O framework **Gin** foi escolhido por seu roteamento HTTP baseado em *radix tree*, reportando desempenho até 40 vezes superior a frameworks anteriores [46]. Alternativas como *Echo* e *Fiber* foram consideradas, mas Gin apresenta comunidade mais ativa e documentação mais abrangente. O ORM **GORM** oferece mapeamento objeto-relacional com suporte a migrações automáticas e operações de *upsert* para garantia de idempotência.

Para ambientes de produção, o sistema implementa **connection pooling** — técnica que mantém um cache de conexões reutilizáveis, evitando o custo de estabelecer novas conexões a cada requisição. Estudos demonstram que esta abordagem pode reduzir a utilização de CPU do banco de dados em até 30% e aumentar significativamente o *throughput* em cenários de alta concorrência. A configuração define limites para conexões ociosas (`MaxIdleConns`), conexões abertas (`MaxOpenConns`) e tempo máximo de vida (`ConnMaxLifetime`), prevenindo acúmulo de conexões obsoletas. Adicionalmente, a opção `PrepareStmt` habilita cache de *prepared statements*, otimizando queries repetidas ao reutilizar planos de execução.

3.5.2 Banco de Dados — PostgreSQL e Redis

Para a camada de persistência, foram avaliadas três opções:

- **MySQL**: Popular e bem documentado, mas com suporte limitado a tipos de dados avançados e funcionalidades analíticas;
- **MongoDB**: Modelo de documentos oferece flexibilidade de schema, porém a natureza relacional dos dados parlamentares (senador $\rightarrow$ mandatos $\rightarrow$ votações) favorece bancos relacionais;
- **PostgreSQL**: Robustez em consultas analíticas, suporte a índices compostos e CTEs (*Common Table Expressions*), essenciais para agregações por senador, período e tipo de despesa.

PostgreSQL foi selecionado. Originado em 1986 na Universidade da Califórnia, Berkeley [47], sua conformidade ACID garante integridade nas operações de ingestão. Funcionalidades como `JSONB` permitem armazenar respostas brutas das APIs para auditoria, combinando benefícios de bancos relacionais e documentais.

Redis atua como camada de cache para rankings pré-computados e totalizadores de gastos. Por armazenar dados inteiramente em memória RAM, Redis alcança latências típicas entre 100 e 500 microssegundos [48], eliminando a necessidade de recalcular métricas a cada requisição. Alternativas como Memcached foram descartadas pela ausência de estruturas de dados avançadas (sorted sets, hashes) que Redis oferece nativamente.

3.5.3 Frontend — Next.js 15

Para a interface web, foram consideradas:

- **React puro (SPA)**: Flexibilidade máxima, porém com SEO comprometido e necessidade de configuração manual de roteamento, SSR e otimizações;
- **Vue.js/Nuxt**: Curva de aprendizado suave, mas ecossistema menor para visualização de dados;
- **Next.js**: Framework React com SSR/SSG nativos, roteamento baseado em sistema de arquivos e otimizações automáticas de imagem e fonte.

A escolha de **Next.js** justifica-se pela necessidade de **SEO** e performance inicial [49]. Plataformas de fiscalização cidadã dependem de indexação por mecanismos de busca para alcançar seu público-alvo. Next.js resolve o problema de indexação com *Server-Side Rendering* (SSR) e *Static Site Generation* (SSG), entregando HTML pré-renderizado aos *crawlers*.

A biblioteca **Recharts** foi selecionada para visualização de dados por sua integração nativa com React [50]. Alternativas como D3.js oferecem maior controle, mas exigem implementação manual de componentes; Chart.js tem API imperativa menos compatível com o paradigma declarativo do React. **Tailwind CSS** permite estilização eficiente com classes utilitárias; a partir da versão 4, o modo *Just-In-Time* (JIT) gera apenas o CSS utilizado [51].

3.6 Modelo de Dados

O modelo relacional foi projetado para garantir integridade referencial e eficiência em consultas analíticas. A Figura 1 apresenta o diagrama entidade-relacionamento do sistema, incluindo as sete entidades principais e seus relacionamentos.

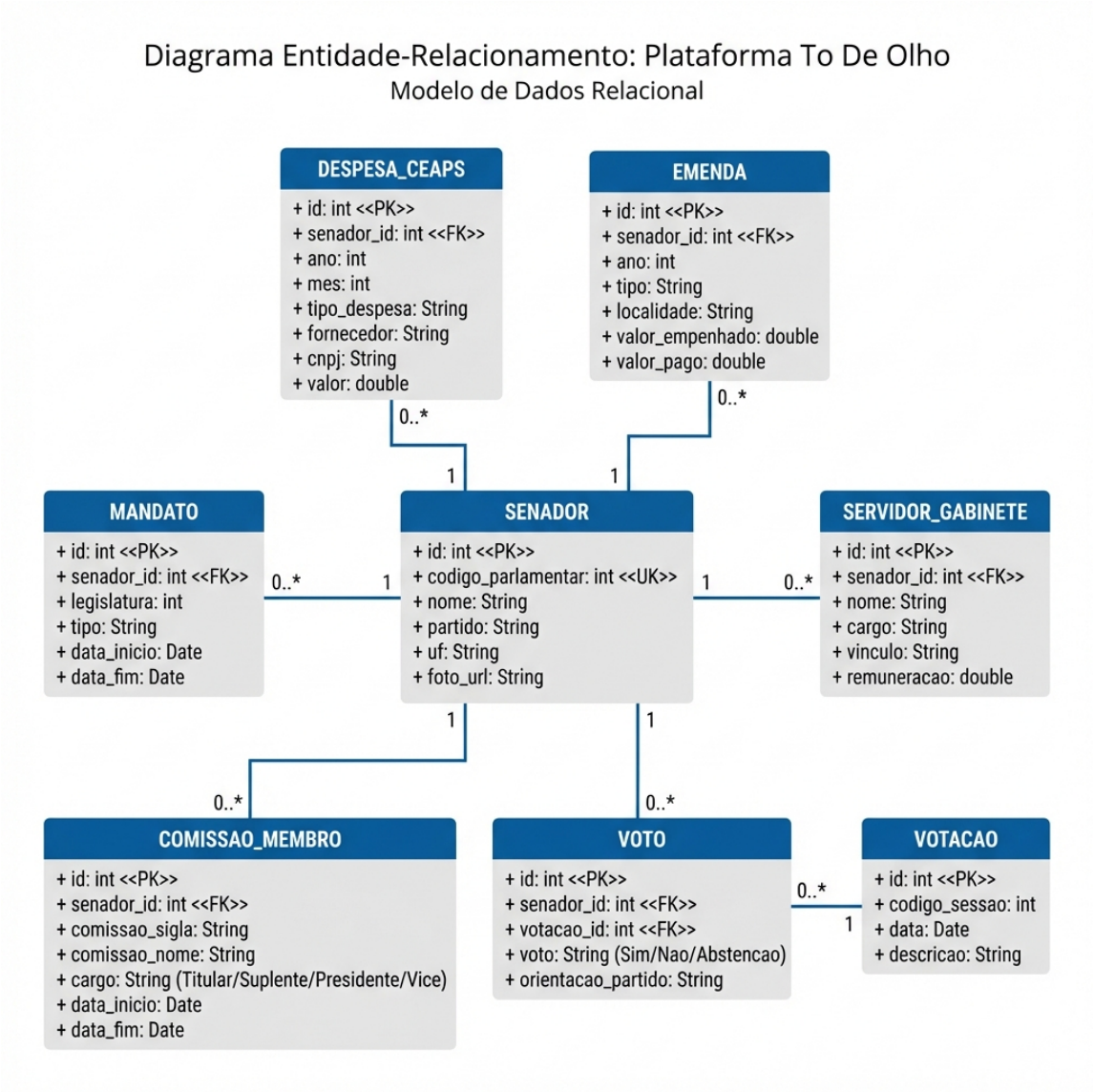


Figura 1: Diagrama entidade-relacionamento do sistema Tô De Olho
Fonte: Autoria Própria

As entidades principais e seus relacionamentos são:

- **Senador**: Entidade central com código parlamentar único (`codigo_parlamentar`), nome, partido, UF e URL da foto. O código parlamentar, fornecido pela API Legislativa, atua como chave natural para integração entre sistemas;
- **Mandato**: Relacionamento 1:N com Senador, registrando legislatura, tipo (titular/suplente) e período. Um senador pode ter múltiplos mandatos em diferentes legislaturas;

- **Votação**: Implementado como tabela associativa entre Senador e Sessão, armazenando voto individual (Sim, Não, Abstenção, Obstrução) e orientação partidária. A combinação `(sessao_id, senador_id)` forma chave única;
- **Despesa CEAPS**: Relacionamento 1:N com Senador, contendo ano, mês, tipo de despesa, fornecedor, CPF/CNPJ e valor. A chave composta `(senador_id, fornecedor_cnpj, data_emissao, valor_centavos)` garante idempotência na ingestão;
- **Servidor de Gabinete**: Relacionamento 1:N com Senador, registrando nome, cargo, vínculo funcional e remuneração bruta/líquida;
- **Emenda**: Relacionamento 1:N com Senador, contendo ano, tipo (Individual, Bancada, Transferência Especial), localidade do gasto e valores (empenhado, liquidado e pago);
- **Comissão Membro**: Relacionamento 1:N com Senador, registrando participação em comissões permanentes e temporárias. Armazena sigla e nome da comissão, cargo exercido (Titular, Suplente, Presidente, Vice-Presidente) e período de atuação. Esta entidade é essencial para o cálculo do critério "Participação em Comissões" (20%) do algoritmo de ranking.

Estratégia de Indexação: Índices compostos foram criados nas colunas de consulta frequente para otimizar operações de agregação:

- `idx_despesa_senador_ano`: `(senador_id, ano)` para totalização de gastos por período;
- `idx_votacao_sessao_senador`: `(sessao_id, senador_id)` para consultas de presença;
- `idx_emenda_senador_tipo`: `(senador_id, tipo, ano)` para filtros por modalidade de emenda;
- `idx_comissao_senador`: `(senador_id, comissao_sigla)` para consultas de participação em comissões.

Esta modelagem permite consultas analíticas eficientes, como ranking de senadores por economia na cota, distribuição geográfica de emendas ou pontuação por participação em comissões estratégicas, sem necessidade de joins custosos em tempo de execução.

3.7 Algoritmo de Ranking

A avaliação objetiva do desempenho parlamentar constitui elemento central para a participação cidadã informada. Conforme a escada de participação de Arnstein [9], o acesso a informações claras e comparáveis é pré-requisito para que cidadãos avancem de níveis meramente consultivos para formas efetivas de controle social. Entretanto, a construção de rankings requer transparência metodológica: critérios e pesos devem ser explícitos para que o público possa avaliar criticamente os resultados apresentados [8].

O cálculo do *score* de cada senador é inspirado no *Legislative Effectiveness Score* (LES), metodologia desenvolvida por Volden e Wiseman para avaliar a efetividade legislativa de parlamentares americanos [7]. O LES define efetividade legislativa como "a capacidade com-

provada de avançar itens da agenda de um parlamentar através do processo legislativo até sua transformação em lei" [27]. A metodologia original utiliza uma matriz de 5 estágios de tramitação por 3 tipos de proposição, resultando em 15 indicadores ponderados.

Para a adaptação brasileira, manteve-se a filosofia de valorizar o avanço de proposições através do processo legislativo, incorporando critérios adicionais relevantes ao contexto de fiscalização cidadã. O índice proposto compõe-se de quatro dimensões objetivamente mensuráveis: **produtividade legislativa** (capacidade de aprovar proposições), **presença em votações** (compromisso com o mandato), **economia na cota parlamentar** (responsabilidade fiscal) e **participação em comissões** (trabalho técnico especializado). A adaptação considerou as especificidades do Senado Federal e a disponibilidade de dados via API.

3.7.1 Critérios e Pesos

O índice compõe-se de quatro critérios objetivos, cujos pesos refletem sua importância relativa para a avaliação de desempenho parlamentar:

Tabela 2: Critérios do algoritmo de ranking

Critério	**Peso**	**Justificativa**
Produtividade Legislativa	35%	Core do LES: capacidade de aprovar proposições
Presença em Votações	25%	Compromisso efetivo com o mandato
Economia na Cota (CEAPS)	20%	Responsabilidade fiscal
Participação em Comissões	20%	Trabalho técnico especializado

3.7.2 Produtividade Legislativa

Este critério avalia o avanço de proposições de autoria do senador através do processo legislativo, atribuindo pontuação crescente por estágio alcançado:

Tabela 3: Pontuação por estágio de tramitação

Estágio	**Pontos**	**Mult. PEC**	**Mult. PLP**
Apresentado	1	$\times$3.0	$\times$2.0
Em Comissão	2	$\times$3.0	$\times$2.0
Aprovado em Comissão	4	$\times$3.0	$\times$2.0
Aprovado em Plenário	8	$\times$3.0	$\times$2.0
Transformado em Lei	16	$\times$3.0	$\times$2.0

Os multiplicadores refletem a maior complexidade de Propostas de Emenda Constitucional (PEC), que exigem quórum qualificado de 3/5 em duas votações, e Projetos de Lei Complementar (PLP), que requerem maioria absoluta. Projetos de Lei Ordinária (PL) utilizam o multiplicador base ($\times$1.0). Requerimentos e moções, por seu menor impacto legislativo, recebem multiplicador reduzido ($\times$0.5).

Bônus de Relatoria: Senadores que atuam como relatores recebem pontuação adicional: +4 pontos para relatoria de PEC, +2 pontos para PLP ou PL, e +1 ponto para relatoria em

comissão permanente.

3.7.3 Presença em Votações

Calculada como a razão entre votações participadas e votações disponíveis:

$$\text{Presença} = \frac{\text{Votações Participadas}}{\text{Votações Disponíveis}} \times 100 \tag{1}$$

As votações disponíveis excluem períodos de licença oficial (médica ou para cargo executivo), garantindo que afastamentos justificados não penalizem o senador. Obstrução **não** conta como presença.

3.7.4 Economia na Cota Parlamentar

Calculada como a proporção não utilizada do teto da CEAPS:

$$\text{Economia} = \left(1 - \frac{\text{Gasto Senador}}{\text{Teto CEAPS}}\right) \times 100 \tag{2}$$

O teto da CEAPS varia por UF (R$ 26.000 para DF até R$ 44.000+ para estados da região Norte). Senadores com gasto acima de 120% do teto recebem pontuação zero neste critério.

3.7.5 Participação em Comissões

Pontuação atribuída conforme o nível de engajamento:

- Membro titular de comissão permanente: +2 pontos por comissão;
- Membro suplente: +1 ponto por comissão;
- Vice-presidente de comissão: +3 pontos;
- Presidente de comissão: +5 pontos;
- Membro de comissão temporária ou CPI: +1 ponto.

Comissões estratégicas recebem multiplicador adicional: CAE (Assuntos Econômicos) e CCJ (Constituição e Justiça) aplicam $\times 1.5$, enquanto CAS (Assuntos Sociais) e CI (Infraestrutura) aplicam $\times 1.2$.

3.7.6 Fórmula Final

Cada métrica é normalizada de 0 a 100 antes da ponderação. O *score* final é calculado por:

$$Score = (\text{Produtividade} \times 0.35) + (\text{Presença} \times 0.25) + (\text{Economia} \times 0.20) + (\text{Comissões} \times 0.20) \quad (3)$$

3.7.7 Tratamento de Casos Especiais

- **Senadores novos (suplentes)**: Período mínimo de 30 dias para entrar no ranking, com *badge* "Novo" por 6 meses;
- **Licença curta ($<$30 dias)**: Mantido no ranking com ajuste nos denominadores;
- **Licença longa ($>$30 dias)**: Exibido com *badge* "Licenciado" e *score* congelado;
- **Dados indisponíveis**: Critério afetado recebe peso zero; demais critérios são reponderados proporcionalmente.

3.8 Infraestrutura e Implantação

A infraestrutura do *Tô De Olho* foi projetada seguindo os princípios da metodologia *Twelve-Factor App* [52], conjunto de boas práticas para construção de aplicações cloud-native que priorizam portabilidade, escalabilidade e manutenibilidade. A adoção de containers como unidade de implantação representa uma evolução natural desses princípios, encapsulando a aplicação e suas dependências em artefatos imutáveis e reproduzíveis.

Esta seção detalha as decisões de containerização, o pipeline de integração e entrega contínuas, e os padrões operacionais que garantem resiliência em ambientes serverless.

3.8.1 Containerização com Docker

Todos os componentes do sistema são containerizados com Docker, utilizando ***multi-stage builds*** para otimização das imagens finais. Esta técnica, recomendada pelas melhores práticas de containerização do Google Cloud [53], separa o ambiente de compilação do ambiente de execução, resultando em imagens significativamente menores e mais seguras.

O processo de build ocorre em dois estágios distintos:

1. **Estágio de Compilação**: Utiliza a imagem oficial `golang:1.21-alpine` contendo todas as ferramentas de desenvolvimento (compilador, gerenciador de dependências, ferramentas de análise). Neste estágio, o código-fonte é compilado em um binário estático;
2. **Estágio de Execução**: Parte da imagem mínima `gcr.io/distroless/static-debian12`, que contém apenas o runtime essencial. O binário compilado é copiado do estágio anterior, descartando todas as ferramentas de build.

Esta abordagem oferece benefícios significativos:

- **Redução de Tamanho**: A imagem final contém apenas o binário Go e bibliotecas de sistema essenciais, resultando em tamanhos típicos de 15–25MB (versus centenas de MB

em imagens não otimizadas). Estudos demonstram reduções de até 97% no tamanho de imagens com esta técnica [53];

- **Superfície de Ataque Reduzida**: Ao eliminar ferramentas de desenvolvimento, shells interativos e pacotes desnecessários, a imagem distroless minimiza vetores de vulnerabilidade — característica crítica para aplicações que processam dados sensíveis;
- **Tempo de Deploy Acelerado**: Imagens menores são transferidas mais rapidamente entre registries e nós de execução, reduzindo o tempo total de implantação.

3.8.2 Pipeline de CI/CD

O pipeline de Integração Contínua e Entrega Contínua (CI/CD) foi implementado com **GitHub Actions**, plataforma que permite definição de workflows como código diretamente no repositório. Estudos empíricos analisando mais de 10.000 repositórios demonstram que a adoção de GitHub Actions para automação de workflows reduz significativamente o tempo de feedback e aumenta a frequência de deploys [54].

O pipeline automatiza quatro etapas sequenciais, cada uma condicionada ao sucesso da anterior:

1. **Build**: Compilação dos binários Go com flags de otimização (`-ldflags=-w -s"` para redução de tamanho) e verificação estática de erros via `go vet`. Esta etapa falha imediatamente se houver erros de compilação, provendo feedback rápido ao desenvolvedor;
2. **Test**: Execução de testes unitários e de integração. Testes de integração utilizam *testcontainers* para provisionar instâncias efêmeras de PostgreSQL e Redis, garantindo reprodutibilidade sem dependência de serviços externos. A cobertura mínima de 80% é verificada automaticamente;
3. **Publish**: Construção da imagem Docker multi-stage e envio para o Google Container Registry (GCR). A imagem é tagged tanto com o SHA do commit quanto com a tag `latest`, permitindo rollback preciso e deploy de versão mais recente;
4. **Deploy**: Atualização automática do serviço no Cloud Run via comando `gcloud run deploy`. O deploy utiliza estratégia de *rolling update*, onde novas instâncias são provisionadas e verificadas antes de receber tráfego.

Para segurança, credenciais sensíveis (chaves de API, tokens de acesso) são armazenadas como **GitHub Secrets** e injetadas como variáveis de ambiente apenas durante a execução do workflow, nunca aparecendo em logs ou artefatos.

3.8.3 Plataforma de Implantação — Google Cloud Run

A implantação ocorre via **Google Cloud Run** [55], plataforma serverless para containers que oferece escala automática (inclusive a zero) e modelo de cobrança por uso. Diferentemente de plataformas Function-as-a-Service (FaaS) que impõem restrições de runtime, Cloud Run executa containers Docker padrão, oferecendo flexibilidade para aplicações que não se

adequam ao modelo de funções efêmeras.

As características que justificaram a escolha incluem:

- **Escala Automática**: O número de instâncias ajusta-se dinamicamente baseado na demanda, de zero a centenas de containers. Para um projeto acadêmico com tráfego irregular, esta característica traduz-se em custo operacional próximo de zero em períodos de baixa atividade;
- **Deploy Simplificado**: A implantação resume-se a um único comando (`gcloud run deploy`), abstraindo complexidades de infraestrutura como balanceamento de carga, certificados TLS e health checks;
- **Integração Nativa**: Cloud Run integra-se nativamente com Cloud SQL (PostgreSQL gerenciado), Memorystore (Redis gerenciado) e Cloud Logging, reduzindo a necessidade de configuração manual de conectividade.

3.8.4 Graceful Shutdown

Em ambientes containerizados, o Cloud Run envia o sinal `SIGTERM` antes de encerrar um container — seja por escalonamento, atualização de versão ou inatividade. O tratamento adequado deste sinal é fundamental para garantir *zero-downtime deployments* e preservar a integridade de operações em andamento.

Falhar em tratar `SIGTERM` adequadamente pode causar consequências graves: perda de requisições parcialmente processadas, corrupção de transações de banco de dados interrompidas, e experiência degradada para usuários cujas requisições são abruptamente encerradas. A literatura sobre resiliência de sistemas distribuídos enfatiza que o encerramento ordenado é tão importante quanto a inicialização correta [42].

O *Tô De Olho* implementa o padrão de **graceful shutdown** através de uma sequência ordenada de ações ao receber `SIGTERM`:

1. **Parada de Aceitação**: O servidor HTTP para de aceitar novas conexões, retornando código 503 (Service Unavailable) para requisições subsequentes. Esta sinalização permite que balanceadores de carga redirecionem tráfego para instâncias saudáveis;
2. **Drenagem de Requisições**: O sistema aguarda a conclusão de requisições em andamento, com timeout configurável de 30 segundos. Este período deve ser calibrado baseado na latência máxima esperada das operações;
3. **Fechamento de Conexões**: Conexões com PostgreSQL e Redis são fechadas de forma ordenada, garantindo que transações pendentes sejam commitadas ou rolled back apropriadamente;
4. **Encerramento do Processo**: Apenas após a conclusão das etapas anteriores, o processo encerra com código de saída zero, sinalizando término bem-sucedido ao orquestrador.

Esta abordagem garante que nenhuma operação de ingestão ou consulta seja interrompida abruptamente, preservando a consistência dos dados e a experiência do usuário.

3.8.5 Mitigação de Cold Start

Plataformas serverless como Cloud Run podem escalar a zero instâncias quando não há tráfego, eliminando custos durante períodos de inatividade. Entretanto, a primeira requisição após um período ocioso incorre em latência adicional conhecida como ***cold start*** — o tempo necessário para provisionar uma nova instância, carregar o container e inicializar a aplicação.

Pesquisas sistemáticas sobre desafios de computação serverless identificam o cold start como um dos principais fatores que afetam a experiência do usuário, com latências que podem variar de centenas de milissegundos a vários segundos dependendo do tamanho da imagem, linguagem de programação e complexidade de inicialização [56].

Para mitigar este efeito, o sistema adota múltiplas estratégias complementares:

- **Imagens Mínimas**: Conforme descrito anteriormente, o uso de imagens distroless reduz drasticamente o tempo de download e extração do container. Imagens menores significam cold starts mais rápidos;
- **Inicialização Preguiçosa**: Conexões com banco de dados e Redis são estabelecidas apenas no primeiro uso (*lazy initialization*), não durante a inicialização do container. Esta técnica adia operações custosas (handshake TCP, autenticação, pool warmup) até que sejam efetivamente necessárias, reduzindo o tempo até a primeira resposta;
- **Binários Estáticos**: A compilação Go com ligação estática elimina dependências de bibliotecas dinâmicas, acelerando o carregamento do executável. O runtime Go, por ser compilado, já oferece vantagem significativa sobre linguagens interpretadas (Python, Node.js) em cenários de cold start;
- **Health Checks Leves**: O endpoint `/health` retorna imediatamente sem verificar dependências externas, permitindo que o Cloud Run marque a instância como saudável rapidamente. Verificações mais profundas ocorrem em endpoints separados (`/ready`).

Para cenários de produção com requisitos de latência mais rigorosos, Cloud Run oferece a configuração `--min-instances` que mantém um número mínimo de instâncias sempre aquecidas, eliminando completamente o cold start às custas de custo fixo. Esta opção não foi habilitada no escopo do MVP, mas representa uma evolução natural conforme o tráfego justifique o investimento.

4 Requisitos

A elicitação de requisitos seguiu as diretrizes da norma ISO/IEC/IEEE 29148 [57], que estabelece boas práticas para especificação de requisitos em projetos de software. Os requisitos foram organizados em duas categorias: funcionais (RF), que descrevem as funcionalidades do sistema, e não-funcionais (RNF), que definem atributos de qualidade.

Para a categorização dos requisitos não-funcionais, adotou-se o modelo de qualidade da ISO/IEC 25010 [58], que define oito características de qualidade de software: funcionalidade, eficiência de desempenho, compatibilidade, usabilidade, confiabilidade, segurança, manutenibilidade e portabilidade. Essa estrutura permite uma cobertura sistemática dos atributos de qualidade esperados para a plataforma.

4.1 Requisitos Funcionais

Conforme destacado na análise de trabalhos relacionados (Seção 1.6), o *Tô De Olho* sintetiza funcionalidades de plataformas consolidadas como *De Olho em Você* e *De Olho no Congresso*, adaptando-as ao contexto do Senado Federal. Os requisitos funcionais foram organizados em sete módulos:

Módulo de Senadores:

- **[RF01]** O sistema deve apresentar a lista atualizada dos 81 senadores com foto, partido e estado.
- **[RF02]** O sistema deve permitir a busca de senadores por nome, sigla partidária ou UF.
- **[RF03]** O sistema deve exibir o perfil completo do senador com abas organizadas (Visão Geral, Gastos, Gabinete, Votações, Emendas).

Módulo de Transparência Financeira (CEAPS):

- **[RF04]** O sistema deve importar os lançamentos da Cota Parlamentar (CEAPS) através das APIs de Dados Abertos do Senado.
- **[RF05]** O sistema deve permitir visualizar o gasto acumulado por tipo de despesa (passagens, correios, consultorias, combustível).
- **[RF06]** O sistema deve exibir os fornecedores que mais receberam recursos de um determinado senador.
- **[RF07]** O sistema deve gerar alertas automáticos para despesas atípicas: valores acima da média, pagamentos em finais de semana, intervalos menores que 3 dias entre pagamentos ao mesmo fornecedor.

Módulo de Emendas e Orçamento:

- **[RF08]** O sistema deve integrar com o Portal da Transparência para buscar emendas de autoria do senador.
- **[RF09]** O sistema deve destacar valores destinados via “Transferências Especiais” (emendas PIX).
- **[RF10]** O sistema deve exibir mapas interativos de distribuição de emendas por município, permitindo identificar concentração geográfica de recursos.

Módulo de Atividade Legislativa:

- **[RF11]** O sistema deve listar as votações nominais recentes e o voto de cada senador (Sim/Não/Abstenção).
- **[RF12]** O sistema deve exibir a participação do senador em comissões permanentes e especiais, incluindo cargo ocupado (titular, suplente, presidente, relator).
- **[RF13]** O sistema deve listar proposições de autoria do senador, com indicação do tipo (PEC, PLP, PL) e estágio de tramitação.
- **[RF14]** O sistema deve exibir discursos proferidos pelo senador em plenário, com data e tema.
- **[RF15]** O sistema deve exibir a agenda pública de reuniões de comissões, permitindo visualizar pautas futuras.
- **[RF16]** O sistema deve apresentar links para os perfis oficiais do senador em redes sociais (Twitter/X, Instagram, Facebook).

Módulo de Gabinete:

- **[RF17]** O sistema deve exibir a lista de servidores do gabinete do senador, incluindo cargo e vínculo.
- **[RF18]** O sistema deve apresentar a folha de pagamento do gabinete, com remuneração mensal por servidor.

Módulo de Comparação e Análise:

- **[RF19]** O sistema deve permitir comparar de 2 a 5 senadores lado a lado em múltiplas dimensões: despesas, emendas, votações e fornecedores em comum.
- **[RF20]** O sistema deve exibir ranking de fornecedores com cruzamento de sanções administrativas, identificando empresas com situação cadastral irregular.
- **[RF21]** O sistema deve mostrar indicadores de confiança: data da última sincronização, completude dos dados e fonte de cada informação.

Módulo de Ranking e Score:

- **[RF22]** O sistema deve calcular e exibir o *Score* de efetividade legislativa de cada senador, baseado no algoritmo de ranking do projeto.
- **[RF23]** O sistema deve apresentar gráfico radar com as quatro dimensões do *Score*: Produtividade Legislativa, Presença em Votações, Economia na Cota e Participação em Comissões.
- **[RF24]** O sistema deve permitir ordenar e filtrar senadores por cada critério individual do ranking.

4.2 Requisitos Não-Funcionais

Desempenho:

- **[RNF01]** O sistema deve responder a requisições de consulta em até 2 segundos sob condições normais de uso.
- **[RNF02]** A arquitetura deve suportar escalabilidade horizontal para lidar com picos de acesso em períodos eleitorais.

Usabilidade e Acessibilidade:

- **[RNF03]** O sistema deve ser acessível via navegadores *web* em dispositivos *desktop* e *mobile*.
- **[RNF04]** A interface deve seguir o padrão *mobile-first* para garantir boa experiência em dispositivos móveis.
- **[RNF05]** O sistema deve seguir as diretrizes de acessibilidade WCAG 2.1 nível AA.

Confiabilidade:

- **[RNF06]** Os dados devem ser sincronizados diariamente com as APIs oficiais do Senado Federal e Portal da Transparência.
- **[RNF07]** O sistema deve manter disponibilidade mínima de 99% durante o período eleitoral.

Segurança e Privacidade:

- **[RNF08]** As comunicações devem ser criptografadas utilizando HTTPS/TLS.
- **[RNF09]** O sistema deve estar em conformidade com a LGPD (Lei Geral de Proteção de Dados).

Manutenibilidade:

- **[RNF10]** A arquitetura modular deve permitir atualizações isoladas de cada módulo sem impacto em outros componentes.
- **[RNF11]** O código deve seguir padrões de desenvolvimento (*linting*, formatação) e estar documentado.
- **[RNF12]** O sistema deve possuir *pipelines* de CI/CD para integração e deploy contínuos.

5 Design

5.1 Projeto UML

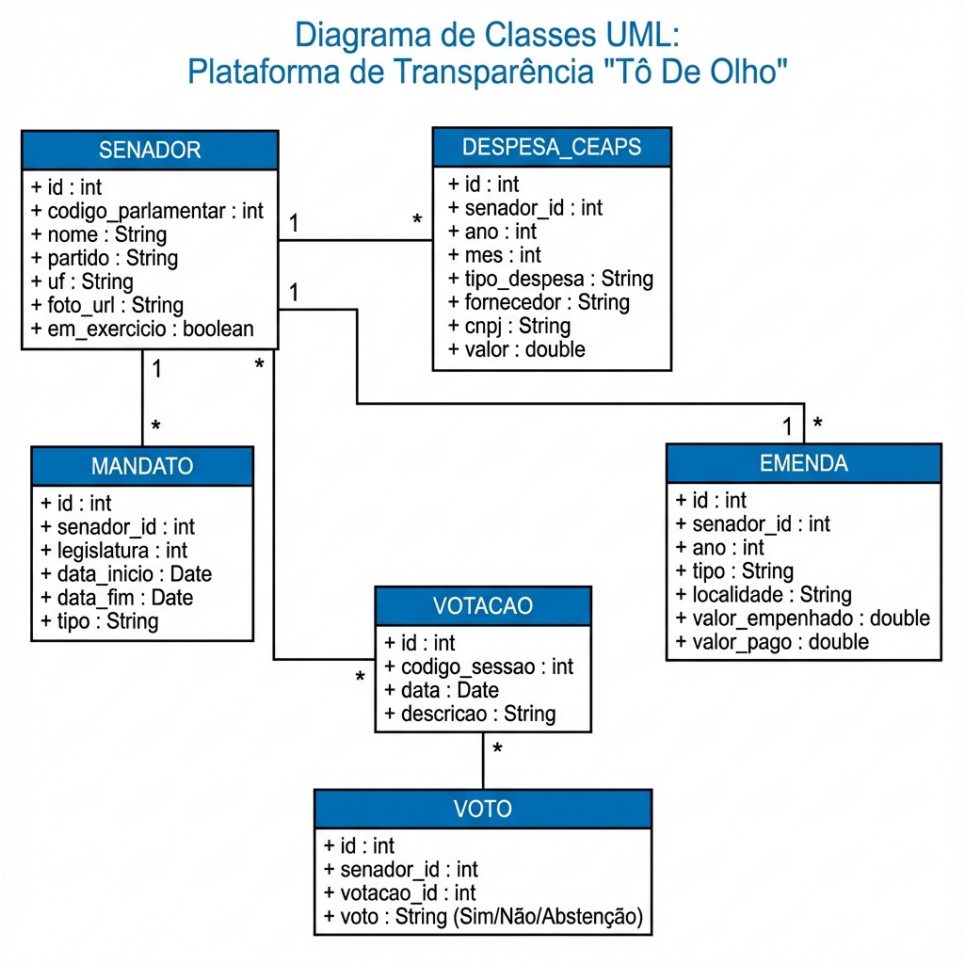
O projeto utiliza a Unified Modeling Language (UML) para documentar visualmente a estrutura e o comportamento do sistema. Abaixo são apresentados os diagramas essenciais para o entendimento da arquitetura proposta.

5.1.1 Diagrama de Classes

O Diagrama de Classes modela a estrutura estática do domínio. As principais entidades identificadas são:

- **Senador**: Representa o parlamentar, contendo atributos como Identificador, Nome, Partido e Estado.
- **Despesa**: Representa um lançamento na CEAPS, associada a um Senador e um Fornecedor.
- **Votacao**: Representa uma sessão deliberativa no Plenário, composta por múltiplos Votos.
- **Emenda**: Representa verba orçamentária destinada pelo senador, contendo Valor, Ano e Beneficiário.

Figura 2: Diagrama de Classes Simplificado (Conceitual)

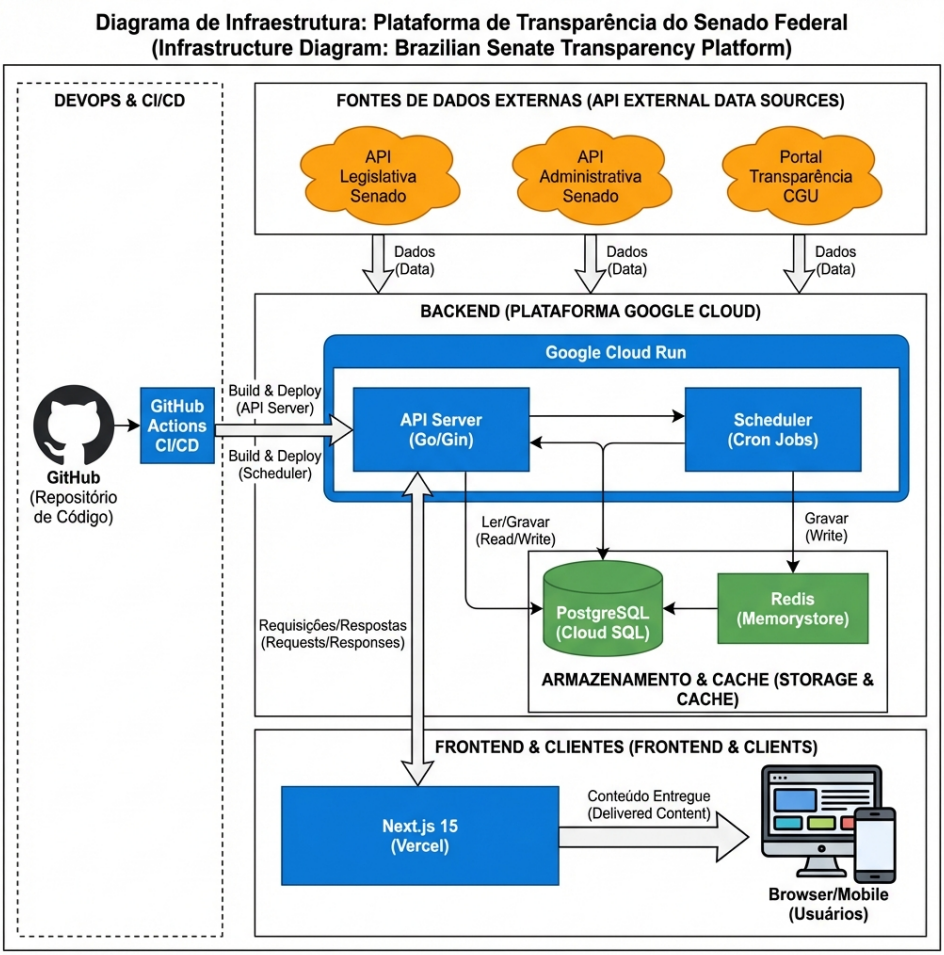


Fonte: Autoria Própria

5.1.2 Diagrama de Implantação

O sistema é implantado em nuvem (GCP), utilizando orquestração de containers.

Figura 3: Diagrama de Implantação e Infraestrutura



Fonte: Autoria Própria

5.2 Visão arquitetural

O *Tô De Olho* adota uma arquitetura de **monolito modular**, onde os componentes são organizados internamente em módulos bem definidos, mas compartilham um único processo de deploy. Essa escolha arquitetural foi motivada pelos seguintes fatores:

- **Simplicidade de Deploy**: um único container simplifica a infraestrutura e reduz custos operacionais;
- **Manutenibilidade**: a organização em módulos internos permite atualizações isoladas sem a complexidade de orquestração distribuída;
- **Evolução Futura**: a estrutura modular permite migração gradual para microsserviços, se necessário.

Módulos Internos:

- `internal/senador`: Gerencia dados cadastrais dos parlamentares e mandatos;
- `internal/ceaps`: Processa e totaliza despesas da Cota Parlamentar;

- `internal/votacao`: Armazena e consulta votações nominais em plenário;
- `internal/emenda`: Integração com Portal da Transparência para emendas parlamentares;
- `internal/ranking`: Cálculo e agregação de scores de desempenho.

Estratégia de Ingestão de Dados:

A plataforma utiliza uma estratégia híbrida de ingestão:

1. **Backfill**: carga inicial histórica através das APIs do Senado, processando dados desde 2019;
2. **Sincronização Contínua**: *CronJobs* diários às 03:00 BRT que consomem as APIs oficiais para manter os dados atualizados.

Infraestrutura:

O sistema é containerizado com Docker e implantado no Google Cloud Run, que oferece:

- **Escala a Zero**: sem custos quando não há tráfego;
- **Deploy Simplificado**: um único Dockerfile para toda a aplicação;
- **Escalabilidade Automática**: instâncias são criadas sob demanda durante picos de acesso.

5.3 Modelo de Banco de Dados

O modelo de dados relacional foi projetado para garantir integridade e eficiência nas consultas analíticas. As tabelas principais são:

- **TB_SENADORES**: Tabela mestre. PK: `codigo_senado`. Colunas: nome, partido, uf, url_foto.
- **TB_MANDATOS**: Histórico de legislaturas. FK para `TB_SENADORES`.
- **TB_DESPESAS_CEAPS**: Armazena cada nota fiscal reembolsada. FK para `TB_SENADORES`. Colunas: valor, data, tipo_despesa, cnpj_fornecedor, url_documento. Indexada por senador e ano para performance em buscas.
- **TB_VOTACOES**: Cabeçalho das votações. PK: `codigo_sessao`.
- **TB_VOTOS**: Tabela de junção (*many-to-many*) entre Senadores e Votações, registrando o voto individual (Sim/Não).
- **TB_EMENDAS**: Registra valores destinados. Colunas: valor_empenhado, valor_pago, beneficiario, modalidade (ex: Pix).

6 Testes de Software

6.1 Projeto de Testes

A estratégia de qualidade do *Tô De Olho* combina testes em diferentes níveis, aproveitando o ferramental nativo da linguagem Go:

- **Testes Unitários**: Validam regras de negócio isoladas, como o cálculo do "Score" do senador e parsers de CSV. Implementados com o pacote padrão `testing` do Go, utilizando a técnica de *Table-Driven Tests* para cobrir múltiplos cenários de borda.
- **Testes de Integração**: Validam a comunicação entre os componentes e o banco de dados. Utiliza-se a biblioteca `testcontainers-go` para subir instâncias efêmeras do PostgreSQL e Redis durante a execução da pipeline, garantindo que as queries e a persistência funcionem como esperado num ambiente controlado.
- **Testes de Contrato (API)**: Asseguram que as respostas dos serviços externos (Senado/Transparência) continuam respeitando os formatos esperados, alertando sobre "Quebras de API" em dependências externas.

7 Implantação

7.1 Projeto de Implantação

A implantação do sistema segue as práticas de *GitOps* e Infraestrutura como Código.

Ambiente de Execução: O sistema é empacotado em uma imagem Docker otimizada (*multi-stage build*) e implantado no Google Cloud Run. Esta plataforma serverless oferece escala automática baseada na demanda, incluindo escala a zero quando não há tráfego, otimizando custos operacionais.

Pipeline de CI/CD: Utiliza-se GitHub Actions para automação. A cada *push* na branch principal:

1. *Build*: Compilação dos binários Go e checagem de erros;
2. *Test*: Execução dos testes unitários e de integração;
3. *Publish*: Criação da imagem Docker e envio para o Google Container Registry;
4. *Deploy*: Atualização automática do serviço no Cloud Run.

8 Manual do Usuário Simplificado

A plataforma é pública e não requer cadastro para consulta, maximizando a transparência.

1. **Acesso**: Navegue para ⟨https://todeolho.org.br⟩.

2. **Ranking**: Na página inicial, visualize os “Top 3” senadores em Economia e Presença.

3. **Busca**: Utilize a barra superior para digitar o nome de um senador.

4. **Detalhes**: Ao clicar em um senador, navegue pelas abas “Gastos” (para ver notas fiscais detalhadas) e “Emendas” (para ver destino das verbas).

5. **Fiscalização**: Use o botão “Compartilhar” para enviar ficha do senador nas redes sociais.

9 Considerações Finais

Este trabalho apresentou o *Tô De Olho*, uma plataforma *web* que centraliza e democratiza o acesso aos dados do Senado Federal. A arquitetura de **monolito modular** em Go, combinada com a ingestão de dados via APIs oficiais, mostrou-se adequada para consolidar informações dispersas em três fontes distintas. O *front-end* em Next.js oferece ao cidadão uma interface acessível para fiscalizar despesas da CEAPS, acompanhar votações nominais e avaliar o desempenho dos 81 senadores por meio de um ranking com metodologia transparente.

Uma limitação relevante deste trabalho é a ausência do módulo de fórum para debate cívico, inicialmente planejado mas não implementado devido a restrições de prazo. Conforme aponta Costa [59], a gestão pública digital efetiva requer não apenas transparência, mas também canais de participação direta. O *Tô De Olho* avança significativamente na dimensão da *accountability*, porém ainda não contempla espaços deliberativos.

Para trabalhos futuros, sugere-se: (1) a implementação do Fórum de Cidadania para debate qualificado; (2) a expansão do escopo para incluir a Câmara dos Deputados, tornando a plataforma bicameral; e (3) a aplicação de técnicas de aprendizado de máquina para detecção de padrões anômalos em despesas parlamentares, seguindo o exemplo da Operação Serenata de Amor. Essas evoluções fortaleceriam o controle social e aproximariam a ferramenta dos degraus superiores da Escada de Arnstein.

Agradecimentos

Agradeço ao meu orientador, Prof. Pablo Vieira Florentino, pela orientação e apoio durante o desenvolvimento deste trabalho. Aos colegas de curso que contribuíram com discussões e sugestões. À minha família pelo apoio incondicional. Por fim, às comunidades de código aberto que mantiveram as ferramentas e documentações utilizadas neste projeto.

Referências

1 GOMES, W. Democracia digital: Que democracia? **Compolítica**, 2010. Disponível em: ⟨http://www.compolitica.org/home/wp-content/uploads/2011/01/gt_ip-wilson.pdf⟩.

2 AVELINO, D. P. de; POMPEU, J. C. B.; FONSECA, I. F. da. Democracia digital: mapeamento de experiências em dados abertos, governo digital e ouvidorias públicas. **Instituto de Pesquisa Econômica Aplicada (Ipea)**, 2021.

3 PATEMAN, C. **Participation and Democratic Theory**. [S.l.]: Cambridge University Press, 1970.

4 GOMES, W. **A democracia no mundo digital**. [S.l.]: Sesc, 2019.

5 ALENCAR, H. N. **O problema da falta de transparência das "emendas PIX" no orçamento constitucional brasileiro**. Tese (Tese de Doutorado em Direito Constitucional) — Instituto Brasileiro de Ensino, Desenvolvimento e Pesquisa (IDP), Brasília, 2024. Orientação: Profa. Dra. Mariana Barbosa Cirne.

6 Ação Educativa; Instituto Paulo Montenegro. **Indicador de Alfabetismo Funcional (INAF) 2024**. São Paulo, 2024. 29% da população brasileira entre 15 e 64 anos é funcionalmente analfabeta.

7 VOLDEN, C.; WISEMAN, A. E. Legislative effectiveness in the american states. **American Political Science Review**, Cambridge University Press, v. 112, n. 4, p. 1–17, 2018. Metodologia do State Legislative Effectiveness Score (SLES).

8 HULLMAN, J.; DIAKOPOULOS, N. Visualization rhetoric: Framing effects in narrative visualization. **IEEE Transactions on Visualization and Computer Graphics**, IEEE, v. 17, n. 12, p. 2231–2240, 2011.

9 ARNSTEIN, S. R. A ladder of citizen participation. **Journal of the American Institute of Planners**, v. 35, n. 4, p. 216–224, 1969.

10 Controladoria-Geral da União. **Portal da Transparência do Governo Federal**. 2024. ⟨https://portaldatransparencia.gov.br⟩. Lançado em novembro de 2004, reformulado em 2018, registra 1,3 a 1,5 milhão de usuários mensais.

11 Brasil. **Lei n. 12.527, de 18 de novembro de 2011 (Lei de Acesso à Informação)**. 2011. ⟨https://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12527.htm⟩. Regula o acesso a informações públicas previsto na Constituição Federal.

12 Senado Federal. **Portal de Dados Abertos do Senado Federal**. 2024. ⟨https://dadosabertos.senado.leg.br⟩. Institucionalizado pelo Ato da Comissão Diretora n. 14 de 2013.

13 Câmara dos Deputados. **Portal de Dados Abertos da Câmara dos Deputados**. 2024. ⟨https://dadosabertos.camara.leg.br⟩. APIs disponíveis desde 2006 (SIT Câmara), modernizado em 2017 com API RESTful v2.

14 ALBUQUERQUE, O. J. d.; ALMEIDA, B. d.; COSTA, L. V. Valor público por meio de tecnologias desenvolvidas com dados governamentais abertos: o caso da operação serenata de amor. **Revista de Administração Pública**, SciELO Brasil, v. 52, n. 4, p. 610–629, 2018.

15 De Olho no Congresso. **De Olho no Congresso: Fiscalize os Gastos dos Deputados Federais**. 2024. ⟨https://deolhonocongresso.com.br⟩. Plataforma de transparência focada em gastos parlamentares da Câmara.

16 De Olho em Você. **De Olho em Você: Transparência que Dá para Entender**. 2024. ⟨https://deolhoemvoce.com.br⟩. Plataforma de transparência focada em Deputados Federais,

com destaque para Emendas PIX e mapas de distribuição.

17 mySociety. **mySociety: Digital Tools for Democracy**. 2024. ⟨https://www.mysociety.org⟩. Organização britânica pioneira em civic tech, responsável pelo TheyWorkForYou.

18 ____. **TheyWorkForYou: Parliamentary Monitoring**. 2024. ⟨https://www.theyworkforyou.com⟩. Plataforma de monitoramento parlamentar do Reino Unido, ativa desde 2004.

19 OpenSecrets. **OpenSecrets: Following the Money in Politics**. 2024. ⟨https://www.opensecrets.org⟩. Organização não-partidária que rastreia financiamento de campanhas nos EUA.

20 MICHENER, G.; CONTRERAS, E.; NISKIER, I. From opacity to transparency? evaluating access to information in brazil five years later. **Revista de Administração Pública**, SciELO Brasil, v. 52, n. 4, p. 610–629, 2018. Avaliação da implementação da Lei de Acesso à Informação (LAI) no Brasil.

21 O'REILLY, T. Government as a platform. In: LATHROP, D.; RUMA, L. (Ed.). **Open Government: Collaboration, Transparency, and Participation in Practice**. Sebastopol, CA: O'Reilly Media, 2010. p. 11–40. ISBN 978-0596804350. Ensaio seminal que cunhou o termo "Government as a Platform" (GaaP).

22 BERNERS-LEE, T. **Linked Data - Design Issues**. 2010. ⟨https://www.w3.org/DesignIssues/LinkedData.html⟩. Princípios fundamentais de Linked Data e esquema de 5 estrelas para dados abertos.

23 Open Government Partnership. **Brazil: Open Government Partnership Member Page**. 2024. ⟨https://www.opengovpartnership.org/members/brazil/⟩. Brasil membro desde 2011, com 6 planos de ação e 130 reformas implementadas.

24 MOE, T. M. The new economics of organization. **American Journal of Political Science**, v. 28, n. 4, p. 739–777, 1984. Artigo seminal sobre teoria de agência aplicada a organizações públicas.

25 LATHROP, D.; RUMA, L. (Ed.). **Open Government: Collaboration, Transparency, and Participation in Practice**. Sebastopol, CA: O'Reilly Media, 2010. 402 p. Coleção de ensaios sobre governo aberto, incluindo contribuições de Tim O'Reilly, Beth Noveck e Aaron Swartz. ISBN 978-0596804350.

26 Transparência Brasil. **Transparência Brasil**. 2024. ⟨https://www.transparencia.org.br⟩. Organização fundada em 2000, dedicada à promoção da integridade no setor público.

27 VOLDEN, C.; WISEMAN, A. E. **Legislative Effectiveness in the United States Congress: The Lawmakers**. [S.l.]: Cambridge University Press, 2014. Obra fundamental sobre metodologia do Legislative Effectiveness Score.

28 Center for Effective Lawmaking. **The Center for Effective Lawmaking**. 2024. ⟨https://thelawmakers.org⟩. Centro de pesquisa co-dirigido por Volden e Wiseman, mantendo dados de efetividade legislativa.

29 DRAGONI, N. et al. Microservices: Yesterday, today, and tomorrow. **Present and Ulterior Software Engineering**, Springer, p. 195–216, 2017. Discussão de trade-offs entre microsserviços e monólitos.

30 LAIGNER, R.; KALINOWSKI, M. et al. A systematic literature review on modular monolith architecture. **Applied Sciences**, MDPI, v. 14, n. 8, p. 3234, 2024. Revisão sistemática sobre arquitetura de monólito modular.

31 EVANS, E. **Domain-Driven Design: Tackling Complexity in the Heart of Software**. [S.l.]: Addison-Wesley, 2003. Obra fundamental sobre design orientado a domínio e bounded contexts. ISBN 978-0321125217.

32 Google. **Service Weaver: A Programming Framework for Writing Distributed Applications**. 2024. ⟨https://serviceweaver.dev⟩. Framework que suporta desenvolvimento modular com deploy flexível.

33 VMware. **Spring Modulith: Building Well-Structured Spring Boot Applications**. 2024. ⟨https://spring.io/projects/spring-modulith⟩. Framework para construção de aplicações modulares em Java.

34 KIMBALL, R.; CASERTA, J. **The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data**. [S.l.]: Wiley, 2004. Obra de referência para processos ETL em data warehousing. ISBN 978-0764567575.

35 VASSILIADIS, P. A survey of extract–transform–load technology. **International Journal of Data Warehousing and Mining**, IGI Global, v. 5, n. 3, p. 1–27, 2009. Revisão abrangente de tecnologias e técnicas ETL.

36 KIMBALL, R.; ROSS, M. **The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling**. 3rd. ed. [S.l.]: Wiley, 2013. Guia definitivo de modelagem dimensional e arquitetura bottom-up. ISBN 978-1118530801.

37 HEVNER, A. R. et al. Design science in information systems research. **MIS Quarterly**, Management Information Systems Research Center, v. 28, n. 1, p. 75–105, 2004. Artigo seminal sobre Design Science Research em sistemas de informacao.

38 PENG, S. et al. The impact of ai on developer productivity: Evidence from github copilot. **arXiv preprint arXiv:2302.06590**, 2023. Estudo controlado randomizado com 95 desenvolvedores mostrando 55.8% aumento na velocidade de conclusao de tarefas.

39 KIMBALL, R.; ROSS, M. **The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling**. 3rd. ed. [S.l.]: John Wiley & Sons, 2013. Fundamentação teórica sobre estratégias de carga de dados (ETL). ISBN 978-1118530801.

40 HUMMER, W. et al. A model-driven approach for document-orientated restful apis. In: IEEE. **2013 IEEE 6th International Conference on Service-Oriented Computing and Applications**. [S.l.], 2013. p. 247–254. Discute idempotência como propriedade essencial para consistência em APIs REST.

41 Amazon Web Services. **Exponential Backoff And Jitter**. 2015. ⟨https://aws.amazon.com/blogs/architecture/exponential-backoff-and-jitter/⟩. Best practice da AWS para retry com backoff exponencial e jitter.

42 NYGARD, M. T. **Release It!: Design and Deploy Production-Ready Software**. 2nd. ed. [S.l.]: Pragmatic Bookshelf, 2018. Obra de referência sobre padrões de estabilidade e resiliência (Circuit Breaker, Bulkheads). ISBN 978-1680502398.

43 SRIDHARAN, C. Distributed systems observability: A guide to building robust systems. O'Reilly Media, 2018. Formalização dos três pilares: logs, métricas e traces.

44 FOWLER, M. Monolithfirst. **martinfowler.com**, 2015. Artigo seminal defendendo a estratégia de iniciar projetos como monolitos antes de microserviços. Disponível em: ⟨https://martinfowler.com/bliki/MonolithFirst.html⟩.

45 NANZ, S.; FURIA, C. A. A comparative study of programming languages in rosetta code. In: IEEE. **2015 IEEE/ACM 37th IEEE International Conference on Software Engineering**. [S.l.], 2015. p. 778–788. Estudo comparativo de linguagens incluindo Go e Python.

46 ALFIAN, M. et al. Analyzing the performance of golang web frameworks utilizing gorm in the oil and gas industry. In: IEEE. **2024 IEEE 9th International Conference on Information Technology and Digital Applications (ICITDA)**. [S.l.], 2024. Avaliação de performance de frameworks Go incluindo Gin.

47 STONEBRAKER, M.; ROWE, L. A. The design of postgres. **ACM SIGMOD Record**, ACM, v. 15, n. 2, p. 340–355, 1986. Paper fundacional do PostgreSQL, apresentado na UC Berkeley.

48 Redis Ltd. **Redis: The Real-time Data Platform**. 2024. ⟨https://redis.io⟩. Banco de dados em memória com latências em microssegundos.

49 SALIM, M. et al. Evaluating the efficacy of next.js: A comparative analysis with react.js on performance, seo, and global network equity. **arXiv preprint arXiv:2403.13350**, 2024. Comparação de performance e SEO entre Next.js e React.

50 Recharts Community. **Recharts: A Composable Charting Library Built on React Components**. 2024. ⟨https://recharts.org⟩. Biblioteca de visualização de dados para React.

51 Tailwind Labs. **Tailwind CSS: A Utility-First CSS Framework**. 2024. ⟨https://tailwindcss.com⟩. Framework CSS com abordagem utility-first e purge automatico.

52 WIGGINS, A. **The Twelve-Factor App**. 2017. ⟨https://12factor.net⟩. Metodologia para construção de aplicações cloud-native, incluindo logs como streams e processos stateless.

53 Google Cloud. **Best Practices for Building Containers**. 2024. ⟨https://cloud.google.com/architecture/best-practices-for-building-containers⟩. Guia oficial do Google sobre otimização de imagens Docker.

54 KINSMAN, T. et al. How do software developers use github actions to automate their workflows? In: **2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)**. [S.l.]: IEEE, 2021. p. 420–431. Análise empírica de workflows GitHub Actions em mais de 10.000 repositórios.

55 Google Cloud. **Cloud Run: Serverless Container Platform**. 2024. ⟨https://cloud.google.com/run⟩. Plataforma serverless para containers com escala automática e cobrança por uso.

56 SILVA, P.; FIREMAN, D.; PEREIRA, T. E. A survey on serverless computing: Challenges and research opportunities. **Future Generation Computer Systems**, Elsevier, v. 115, p. 213–236, 2020. Revisão sistemática de desafios serverless, incluindo estratégias de mitigação de cold start.

57 ISO/IEC/IEEE. **ISO/IEC/IEEE 29148:2018 - Systems and software engineering – Life cycle processes – Requirements engineering**. 2018. ⟨https://www.iso.org/standard/72089.html⟩. Padrão internacional que substitui IEEE 830-1998 para especificação de requisitos de software.

58 ISO/IEC. **ISO/IEC 25010:2011 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models**. 2011. ⟨https://www.iso.org/standard/35733.html⟩. Define oito características de qualidade de produto de software: funcionalidade, eficiência, compatibilidade, usabilidade, confiabilidade, segurança, manutenibilidade e portabilidade.

59 COSTA, E. A. D. **Gestão Pública Digital: o poder das TIC na democracia brasileira**. Dissertação (Dissertação de Mestrado em Gestão Pública e Sociedade) — Universidade Federal de Alfenas (UNIFAL-MG), Varginha, MG, 2019. Disponível no Repositório Institucional UNIFAL-MG.

Apêndices

Glossário, Siglas e Abreviações

ACID *Atomicity, Consistency, Isolation, Durability* — propriedades que garantem integridade em transações de banco de dados.

ADPF Arguição de Descumprimento de Preceito Fundamental — ação judicial perante o STF para questionar violações à Constituição.

API *Application Programming Interface* — interface de programação que permite a comunicação entre sistemas de software.

CEAP Cota para o Exercício da Atividade Parlamentar — verba destinada aos Deputados Federais (equivalente à CEAPS para senadores).

CEAPS Cota para o Exercício da Atividade Parlamentar dos Senadores — verba destinada ao custeio de despesas relacionadas ao exercício do mandato parlamentar.

CGU Controladoria-Geral da União — órgão responsável pelo controle interno do Poder Executivo Federal e pela transparência pública.

CI/CD *Continuous Integration / Continuous Deployment* — práticas de integração e implantação contínuas de software.

CSR *Client-Side Rendering* — renderização de páginas no navegador do usuário.

CSV *Comma-Separated Values* — formato de arquivo de texto para dados tabulares.

DVL *Data Visualization Literacy* — literacia em visualização de dados; capacidade de interpretar representações visuais de informações.

ETL *Extract, Transform, Load* — processo de extração, transformação e carga de dados.

FCP *First Contentful Paint* — métrica de performance web que mede o tempo até o primeiro conteúdo visível.

GCP *Google Cloud Platform* — plataforma de serviços em nuvem do Google.

INAF Indicador de Alfabetismo Funcional — pesquisa brasileira que avalia níveis de alfabetização da população.

JIT *Just-In-Time* — compilação sob demanda; no contexto de CSS, geração de estilos apenas quando utilizados.

JSON *JavaScript Object Notation* — formato leve de intercâmbio de dados.

LAI Lei de Acesso à Informação (Lei nº 12.527/2011) — legislação que garante o direito de acesso a informações públicas.

LCP *Largest Contentful Paint* — métrica de performance web que mede o tempo até o maior elemento visível.

LES *Legislative Effectiveness Score* — metodologia para avaliação de efetividade legislativa desenvolvida por Volden e Wiseman.

LGPD Lei Geral de Proteção de Dados (Lei nº 13.709/2018) — legislação brasileira sobre privacidade e proteção de dados pessoais.

MVP *Minimum Viable Product* — produto mínimo viável, versão inicial com funcionalidades essenciais.

OGP *Open Government Partnership* — Parceria para Governo Aberto; iniciativa multilateral para promoção da transparência.

ORM *Object-Relational Mapping* — técnica de mapeamento objeto-relacional para persistência de dados.

PAC *Political Action Committee* — comitê de ação política nos EUA para arrecadação de fundos eleitorais.

PEC Proposta de Emenda à Constituição — proposição legislativa que visa alterar a Constituição Federal; exige quórum qualificado de 3/5 em duas votações.

PIX Sistema de pagamentos instantâneos do Banco Central; no contexto parlamentar, refere-se às "Emendas PIX" (Transferências Especiais) — modalidade de repasse que dispensa convênio.

PLP Projeto de Lei Complementar — proposição que regulamenta matérias específicas previstas na Constituição; exige maioria absoluta para aprovação.

RDF *Resource Description Framework* — modelo de dados para representação de informações na web semântica.

REST *Representational State Transfer* — estilo arquitetural para APIs *web*.

SEO *Search Engine Optimization* — otimização para motores de busca.

SIAFI Sistema Integrado de Administração Financeira do Governo Federal — sistema de contabilidade pública do governo brasileiro.

SLES *State Legislative Effectiveness Score* — versão estadual do LES, adaptada para legislaturas subnacionais.

SSG *Static Site Generation* — geração estática de páginas web em tempo de build.

SSR *Server-Side Rendering* — renderização de páginas no servidor.

STF Supremo Tribunal Federal — órgão máximo do Poder Judiciário brasileiro.

TIC Tecnologias da Informação e Comunicação.

TLS *Transport Layer Security* — protocolo de segurança para comunicações criptografadas na internet.

TTL *Time To Live* — tempo de vida de dados em cache.

UF Unidade Federativa — estado brasileiro.

URI *Uniform Resource Identifier* — identificador único de recursos na web.

W3C *World Wide Web Consortium* — organização internacional que desenvolve padrões para a web.

WCAG *Web Content Accessibility Guidelines* — diretrizes de acessibilidade para conteúdo *web*.

XML *eXtensible Markup Language* — linguagem de marcação para estruturação de dados.