FiscalizaDeputado

Equipe - 3

Mikaell Miguel da Silva (mms14)
Pablo Henrique (phfs2)
Sandrirames Albino Fausto(saf)

Recife, 13/08/2025

Prof. Vinicius Cardoso Garcia (vcg@cin.ufpe.br)

Sumário

1. Introdução 2
2. Contexto e Problema 2
3. Público-Alvo e Personas 3
    Persona 1: João, o Cidadão Consciente 3
    Persona 2: Ana, a Jornalista Investigativa 3
4. Proposta de Valor e Solução 3
5. MVP e Funcionalidades 4
6. Modelagem e Design 5
    6.1 Diagrama Lógico 5
    6.2 Modelo C4 - C1 6
    6.3 Modelo C4 - C2 6
    6.4 Modelo C4 - C3 7
7. Processo de Desenvolvimento 7
    7.1 Ferramentas Utilizadas 7
    7.2 Checklist de Planejamento vs. Execução 8
    7.3 Principais Sprints 9
8. Garantia de Qualidade e Segurança 9
9. Arquitetura Técnica e Implementação 10
10. Implantação e DevOps 10
11. Colaboração e Versionamento 12
12. Lições Aprendidas 12
13. Conclusão 13
14. Anexos 14
    14.1 Links úteis 14
    14.2 Quadros de Gestão - Github Project 15

1. Introdução

O projeto *FiscalizaDeputado* é uma aplicação web que tem como objetivo ampliar o acesso da população brasileira às informações sobre despesas declaradas por deputados federais. A partir dos dados abertos do portal da Câmara dos Deputados, o sistema permite que cidadãos acompanhem o uso dos recursos públicos, facilitando a fiscalização, a transparência e o controle social.

O principal objetivo do *FiscalizaDeputado* é oferecer uma plataforma intuitiva e acessível para consulta, análise e visualização dos gastos parlamentares, promovendo a participação cidadã e o fortalecimento da democracia.

A escolha deste tema se justifica pela crescente demanda da sociedade por transparência pública e pela necessidade de ferramentas que democratizam o acesso a informações oficiais, tornando o acompanhamento da atuação dos políticos mais simples e efetivo.

A metodologia adotada para o desenvolvimento do projeto combina práticas ágeis, utilizando o framework Scrum para garantir entregas incrementais e o alinhamento contínuo com as necessidades dos usuários e stakeholders.

2. Contexto e Problema

O problema central abordado pelo FiscalizaDeputado é a dificuldade enfrentada pela população em acessar, entender e fiscalizar os dados públicos referentes aos gastos realizados por deputados federais. Apesar da existência de bases de dados abertas, a grande quantidade e complexidade da informação dificultam o acompanhamento efetivo por parte de cidadãos comuns.

Essa limitação compromete a transparência, o controle social e a prestação de contas, abrindo espaço para o uso inadequado de recursos públicos. O DataSenado (2022) apontou que apenas 17% da população já acessou alguma vez o portal de dados da Câmara ou do Senado, evidenciando o baixo alcance das ferramentas oficiais.

O domínio de aplicação está na área de transparência pública e tecnologia para a participação social, motivado pelo impacto direto que o acesso facilitado a dados parlamentares pode trazer para o fortalecimento da democracia e a melhoria da gestão pública.

3. Público-Alvo e Personas

O público-alvo do *FiscalizaDeputado* inclui cidadãos interessados em transparência pública, jornalistas investigativos, pesquisadores acadêmicos, órgãos de controle e organizações da sociedade civil dedicadas à fiscalização dos recursos públicos.

Persona 1: João, o Cidadão Consciente

- **Idade:** 35 anos
- **Profissão:** Professor universitário
- **Dores:** Dificuldade em encontrar informações claras sobre os gastos dos deputados que representa sua cidade.
- **Motivações:** Quer acompanhar o uso do dinheiro público para exercer sua cidadania e orientar seus alunos sobre participação política.
- **Comportamentos:** Busca por fontes confiáveis de informação, utiliza a internet para pesquisa e participa de grupos de discussão sobre política.

Persona 2: Ana, a Jornalista Investigativa

- **Idade:** 28 anos
- **Profissão:** Repórter de política
- **Dores:** Gasta muito tempo compilando dados públicos para montar matérias de investigação.
- **Motivações:** Quer agilizar seu trabalho com ferramentas que facilitem a análise dos gastos parlamentares.
- **Comportamentos:** Utiliza ferramentas digitais avançadas, valoriza dados atualizados e precisos, e busca transparência nas fontes.

4. Proposta de Valor e Solução

O FiscalizaDeputado oferece uma solução integrada para consulta, análise e visualização dos gastos parlamentares, com diferenciais como:

- Interface amigável que facilita a compreensão dos dados mesmo para usuários sem conhecimento técnico.
- Visualizações interativas, incluindo gráficos e rankings.
- Sistema de acompanhamento personalizado, permitindo que usuários "sigam" parlamentares.

- Filtros dinâmicos que possibilitam análises por tipo de gasto, período e estado, dados como fornecedores com maior valor recebido ampliando o poder investigativo do cidadão.
- Disponibilização de dados abertos para uso jornalístico, acadêmico e social.

Esses diferenciais tornam o FiscalizaDeputado uma ferramenta única no contexto brasileiro, agregando valor ao fortalecer o controle social, estimular a participação cidadã e fomentar a cultura da transparência pública.

Hipóteses iniciais incluem a aceitação da plataforma por parte do público-alvo e o aumento do engajamento cívico a partir do acesso facilitado aos dados. Validações parciais foram realizadas por meio de testes com usuários e feedbacks preliminares, que indicaram alta usabilidade e interesse na ferramenta.

5. MVP e Funcionalidades

O MVP (Produto Mínimo Viável) deste projeto foi concebido para entregar as funcionalidades essenciais que validam a proposta de valor da solução, focado em facilitar o acompanhamento e análise das despesas dos deputados federais. O objetivo é oferecer uma plataforma intuitiva, segura e eficiente para consulta, comparação e monitoramento de dados públicos, promovendo transparência e engajamento cidadão. As principais funcionalidades entregues no MVP incluem:

- **Consulta de Deputados:** Visualização de informações detalhadas dos deputados, incluindo dados pessoais, partido, estado e histórico de despesas.
- **Dashboard de Despesas:** Exibição de gráficos e tabelas interativas para análise das despesas por deputado, partido, estado, tipo de gasto e período.
- **Ranking de Gastos:** Listagem dos deputados e fornecedores com maiores despesas, permitindo filtros e ordenações.
- **Sistema de Autenticação:** Cadastro, login, verificação por e-mail e gerenciamento de sessão do usuário.
- **Favoritar Deputados:** Funcionalidade para usuários acompanharem deputados de interesse, recebendo atualizações.
- **Filtros Avançados:** Busca e filtragem por nome, partido, estado, tipo de despesa e período.
- **Página de Perfil:** Área do usuário para gerenciamento de dados pessoais e deputados favoritos.

Para garantir a entrega no tempo determinado para o MVP do projeto, algumas decisões de escopo foram tomadas:

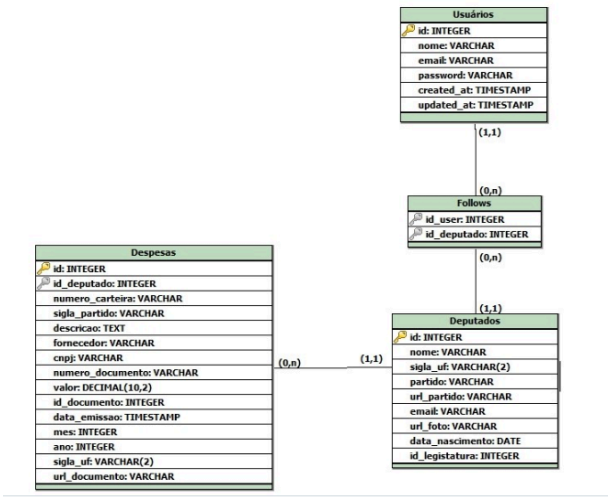
- **Foco nas funcionalidades de consulta e análise:** Recursos como notificações, integração com redes sociais/google maps e exportação de dados foram postergados para versões futuras.
- **Interface simplificada:** Priorização de usabilidade e clareza, com componentes reutilizáveis e navegação intuitiva.
- **Segurança básica:** Implementação de autenticação e verificação por e-mail, com planos para reforço de segurança em releases posteriores.
- **Dados públicos:** Utilização de dados abertos da Câmara dos Deputados, garantindo atualização e confiabilidade.

Além disso, foram desenvolvidos **wireframes no Figma** para representar a estrutura e o layout das principais telas da aplicação. Esses wireframes serviram como referência para a implementação do frontend, garantindo alinhamento entre a proposta visual e as funcionalidades planejadas Ademais destaca-se que as telas implementadas se encontram ao final da seção de Anexos.. Por fim, segue um link de um vídeo hospedado no youtube para ilustrar uma jornada típica de usuário da aplicação em funcionamento.

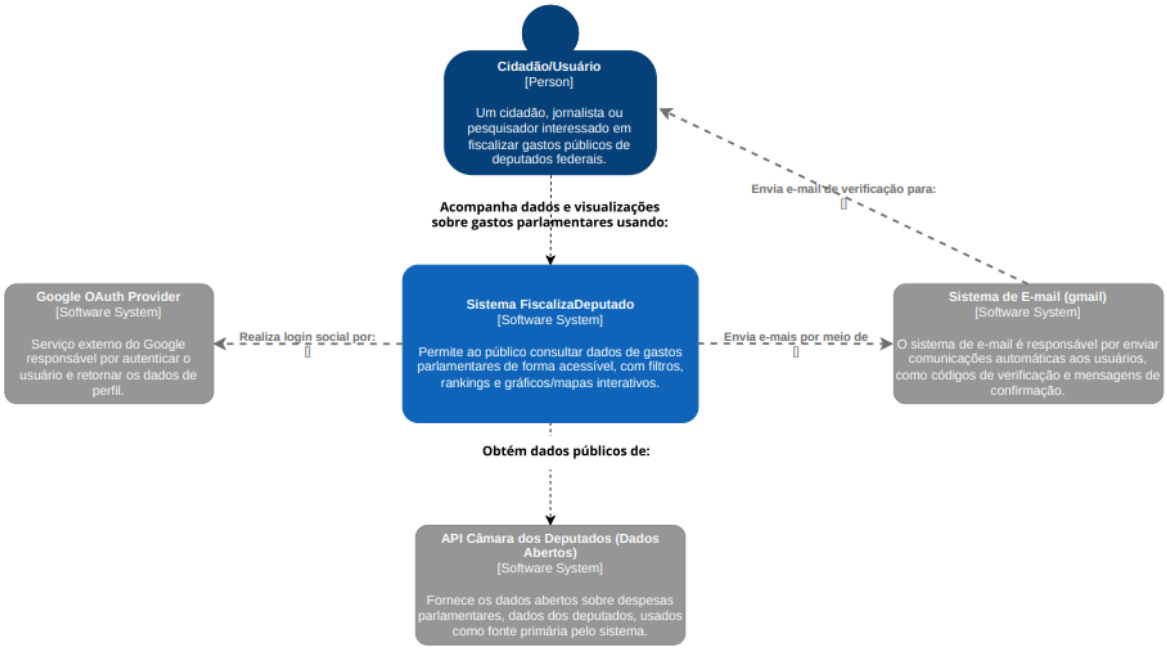
6. Modelagem e Design

6.1 Diagrama Lógico

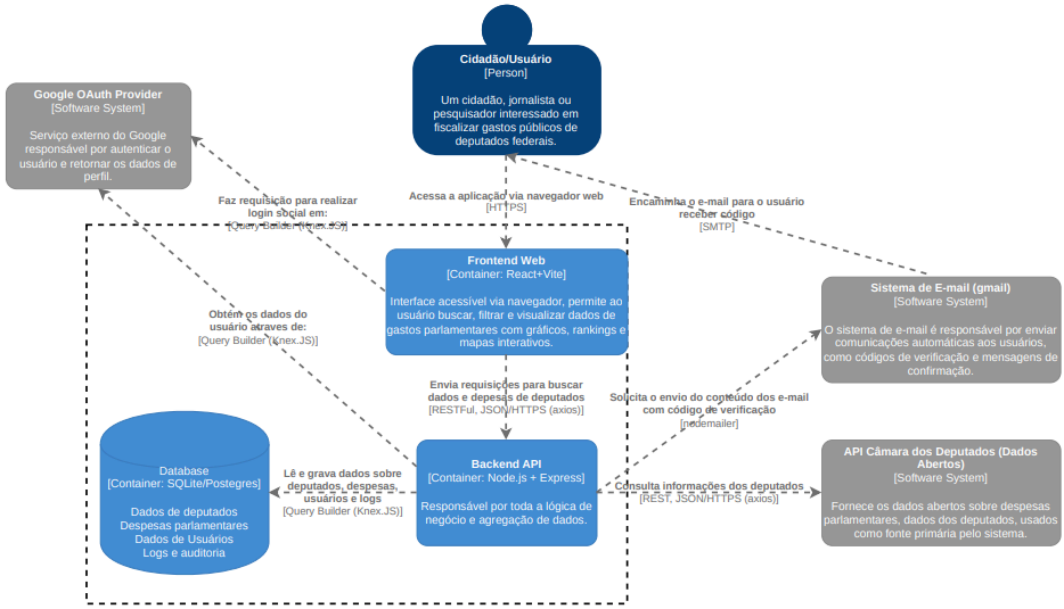
Representa as entidades principais do sistema FiscalizaDeputado, como **Usuário**, **Deputado**, **Despesa**, **Seguidor** e demais relacionamentos entre elas. Esse modelo define como os dados são estruturados e relacionados no banco, servindo de base para consultas e integridade das informações.



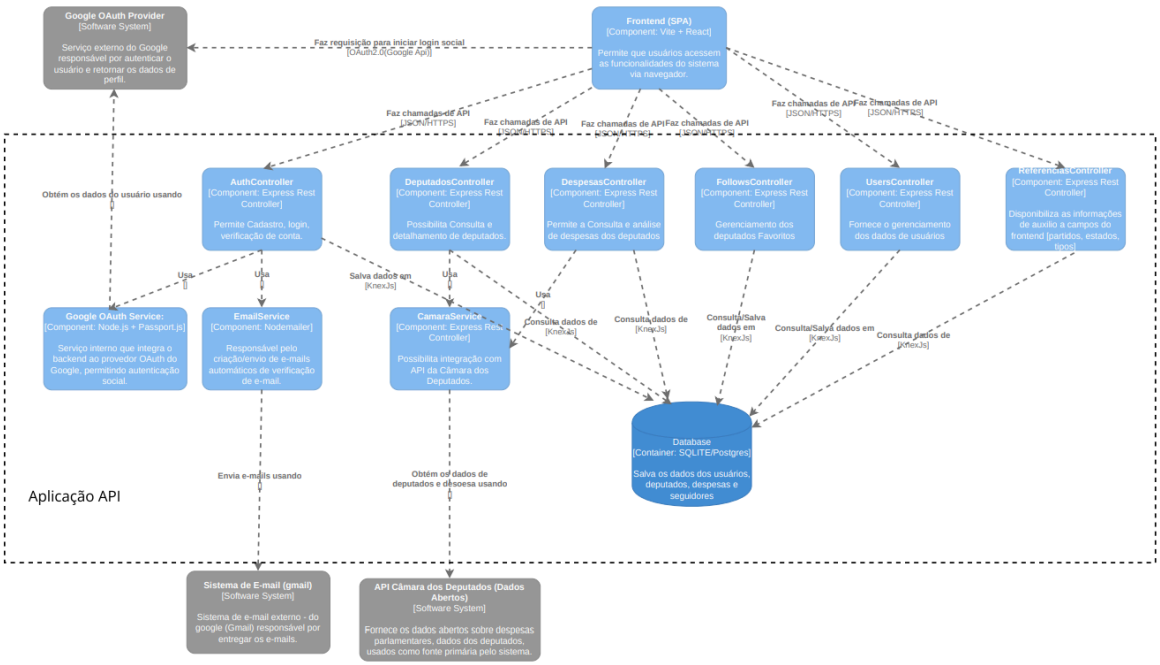
6.2 Modelo C4 - C1



6.3 Modelo C4 - C2



6.4 Modelo C4 - C3



É válido ressaltar que as imagens acima foram retiradas diretamente do repositorio podendo ser encontradas em melhor definição em:
https://github.com/mikaellmiguel/IF977-2025.1-FRONTEND
https://github.com/mikaellmiguel/IF977-2025.1-BACKEND

7. Processo de Desenvolvimento

Inicialmente, a equipe do projeto FiscalizaDeputado adotou a **metodologia Scrum**, com papéis definidos de forma clara: Product Owner, Scrum Master e desenvolvedores com funções específicas de frontend e backend. No entanto, devido à indisponibilidade de alguns membros em determinados momentos, tornou-se inviável manter os papéis fixos, levando a uma abordagem mais **multidisciplinar**, na qual todos colaboraram em diferentes áreas conforme a necessidade. Essa adaptação permitiu maior flexibilidade, mas também exigiu maior comunicação e coordenação entre os integrantes.

7.1 Ferramentas Utilizadas

- **GitHub Projects:** Utilizado para o gerenciamento do backlog, acompanhamento de issues e organização das sprints, centralizando a gestão do projeto e facilitando a comunicação entre os membros da equipe.

- **Figma:** Ferramenta escolhida para a criação dos wireframes, que serviram como referência visual para o desenvolvimento das interfaces do frontend.
- **Postman:** Utilizada para testar e validar os endpoints da API durante o desenvolvimento, garantindo que o backend estivesse funcionando corretamente antes da integração com o frontend.

Essas ferramentas foram selecionadas por sua facilidade de uso, integração com o fluxo de trabalho da equipe e por permitirem uma gestão eficiente do projeto e validação técnica dos requisitos.

7.2 Checklist de Planejamento vs. Execução

No planejamento inicial, o roadmap previa a entrega de funcionalidades essenciais para o MVP, como consulta e análise de despesas de deputados, autenticação de usuários, dashboards interativos e sistema de favoritos. Algumas ideias, como integração com APIs externas adicionais (ex: consulta de CNPJ ou Google Street View) e notificações, foram consideradas, mas priorizou-se a implementação das funcionalidades centrais devido ao tempo disponível e à complexidade técnica.

Durante a execução, o foco foi garantir a entrega de um sistema funcional e estável, priorizando as demandas do MVP. Ajustes no escopo foram realizados conforme surgiram desafios técnicos e limitações de tempo, mantendo a qualidade e a usabilidade do produto.

- ☑ ~~Consulta de despesas de deputados~~
- ☑ ~~Análise de despesas (dashboards e gráficos)~~
- ☑ ~~Autenticação de usuários~~
- ☑ ~~Sistema de favoritos (acompanhamento de deputados)~~
- ☑ ~~Integração entre frontend e backend~~
- ☑ ~~Testes automatizados das principais funcionalidades~~
- ☑ ~~Wireframes e protótipos visuais no Figma~~
- ☐ Integração com APIs externas adicionais (ex: consulta de CNPJ, Google Street View)
- ☐ Sistema de notificações
- ☐ Responsividade
- ☐ Testes de Integração
- ☐ Comparações entre deputados

7.3 Principais Sprints

- **Sprint 1 (06/07-20/07):** Estruturação inicial do backend, criação dos principais endpoints para deputados e despesas, e implementação dos primeiros testes unitários.
- **Sprint 2 (21/07-04/08):** Desenvolvimento das funcionalidades básicas do frontend, integração inicial com o backend, implementação de autenticação e visualização de despesas.
- **Sprint 3 (04/08-13/08):** Finalização das funcionalidades, integração completa entre backend e frontend, implementação dos dashboards, gráficos de análise, ranking de gastos e sistema de favoritos e deploy.

8. Garantia de Qualidade e Segurança

A garantia de qualidade e segurança foi um aspecto priorizado desde o início do desenvolvimento do FiscalizaDeputado. Desde as primeiras etapas, buscou-se adotar práticas que prevenissem erros, asseguram o correto funcionamento do sistema e protegessem a aplicação contra ataques comuns, mesmo considerando que o projeto ainda se encontra em sua fase inicial.

No que diz respeito à qualidade, foram aplicados testes unitários automatizados desde o começo do ciclo de desenvolvimento. Esses testes tiveram como objetivo verificar o funcionamento isolado de funções e módulos críticos, como controladores e serviços, garantindo que as regras de negócio fossem validadas de forma independente. A execução regular desses testes permitiu identificar e corrigir regressões rapidamente, mantendo a confiabilidade do código. Nesta primeira fase, optou-se por não aplicar testes de integração, sistema ou aceitação, priorizando os testes unitários em função do escopo e do tempo disponível para o projeto.

Quanto à análise estática de código, não foram utilizadas ferramentas específicas para essa finalidade. Ainda assim, a organização do código e as revisões manuais realizadas durante o desenvolvimento asseguraram a conformidade com boas práticas e contribuíram para a legibilidade e manutenção do sistema.

Em relação à segurança, embora não tenham sido aplicados testes formais específicos para detecção de vulnerabilidades, foram adotadas medidas fundamentais para mitigar riscos. A validação das entradas fornecidas pelos usuários foi implementada de forma a prevenir ataques como SQL Injection, garantindo que dados externos fossem processados com segurança. O uso do Knex.js, um query builder que abstrai a execução direta de comandos SQL, também contribuiu para prevenir grande parte das tentativas de injeção maliciosa.

Além disso, o sistema de autenticação foi projetado para ser robusto, incluindo a verificação de e-mail por meio de código de confirmação, o que impede a criação de contas com endereços de terceiros. Outro ponto de destaque é a utilização obrigatória de HTTPS para todas as comunicações entre cliente e servidor, assegurando que os dados transmitidos sejam criptografados e não possam ser interceptados ou alterados por terceiros mal-intencionados.

Essas práticas, mesmo sem o emprego de mecanismos avançados de segurança nesta fase, proporcionam uma base sólida de proteção e qualidade, alinhada às necessidades e limitações do estágio atual do projeto.

9. Arquitetura Técnica e Implementação

O projeto foi desenvolvido utilizando uma arquitetura em camadas, onde cada parte do sistema possui responsabilidades bem definidas, promovendo organização, escalabilidade e facilidade de manutenção. No frontend, foi utilizada a linguagem JavaScript com o framework React, além do Vite para otimizar o processo de desenvolvimento e build. O backend foi implementado em Node.js, utilizando o framework Express para a construção das APIs REST, Knex.js para abstração e manipulação do banco de dados PostgreSQL, e ferramentas como Jest para testes automatizados.

A estrutura dos repositórios reflete essa separação: o frontend está organizado em módulos de componentes, páginas, hooks e serviços, enquanto o backend possui camadas para controllers, serviços, middlewares, rotas e integração com o banco de dados. O padrão adotado não é estritamente MVC, mas sim uma arquitetura em camadas, garantindo que a lógica de apresentação, negócio e persistência estejam desacopladas.

As decisões técnicas priorizaram tecnologias modernas, com grande comunidade e documentação, além da facilidade de uso e conhecimento prévio da equipe. O uso de ferramentas como ESLint, Figma (para prototipação visual das interfaces), Postman (para testes de API) e GitHub Projects (para gestão ágil do projeto) contribuíram para a qualidade e organização do desenvolvimento.

FRONTED: https://github.com/mikaellmiguel/IF977-2025.1-BACKEND
BACKEND: https://github.com/mikaellmiguel/IF977-2025.1-FRONTEND

10. Implantação e DevOps

A estratégia de deploy adotada para o projeto buscou garantir escalabilidade, disponibilidade e facilidade de manutenção. O backend e o

frontend foram implantados em ambientes distintos, cada um utilizando tecnologias adequadas ao seu contexto.

- **Banco de Dados:** O banco de dados foi provisionado no Azure Database for PostgreSQL, garantindo alta disponibilidade, backups automáticos e segurança dos dados. Inicialmente, pensou-se em realizar o deploy pelo SupaDatabase por ser gratuito até certo ponto. No entanto, a quantidade de registros de despesas, levou a uma solução paga, apesar de acessível.
- **Backend (API)**: O deploy da API foi realizado utilizando o serviço App Web for Containers do Azure, em conjunto com **Docker.** Isso permitiu empacotar toda a aplicação e suas dependências em containers, facilitando o gerenciamento, atualização e escalabilidade do serviço.
- **Frontend:** O deploy do frontend foi feito na Vercel, plataforma especializada em aplicações React, proporcionando integração contínua, deploy automático e CDN para entrega rápida dos arquivos estáticos totalmente gratuitos.

Além disso, no **backend (link do workflow)**, o pipeline executa testes unitários automatizados para validar o funcionamento das funcionalidades antes de qualquer integração. No **frontend (link do workflow)**, o pipeline também realiza testes automatizados para garantir que o código se comporte conforme esperado.

Esses pipelines asseguram que o código integrado seja testado continuamente, reduzindo riscos e aumentando a confiabilidade do sistema, mesmo que outros processos como build e deploy automatizado ainda não estejam configurados.

A documentação de build e execução do projeto foi elaborada para simplificar o processo de instalação, configuração e inicialização tanto do backend quanto do frontend. No backend, após instalar as dependências com npm install e configurar as variáveis de ambiente, é possível executar os testes automatizados e iniciar a aplicação localmente ou construir a imagem Docker com um único comando, facilitando o deploy em ambientes como o Azure App Web for Containers. Esse processo garante escalabilidade, portabilidade e segurança para o serviço.

No frontend, o fluxo também começa com a instalação das dependências via npm install e configuração das variáveis de ambiente para integração com a API. O projeto pode ser executado em modo de desenvolvimento com hot reload ou preparado para produção com o comando de build. O deploy é automatizado na plataforma Vercel, permitindo integração contínua e entrega rápida dos arquivos estáticos. Toda a documentação orienta de forma clara e objetiva,

possibilitando que novos desenvolvedores executem e contribuam com o projeto de maneira eficiente e segura.

11. Colaboração e Versionamento

O projeto está organizado em dois repositórios distintos, refletindo a separação entre as camadas de frontend e backend:

- **Frontend:** Responsável pela interface do usuário, desenvolvido em React, com estrutura modular para componentes, páginas, hooks e serviços.
- **Backend:** Responsável pela API e lógica de negócio, desenvolvido em Node.js/Express, com módulos para controladores, rotas, serviços, banco de dados e testes.

Essa separação facilita o desenvolvimento independente de cada camada, permitindo maior flexibilidade e escalabilidade.

O histórico de contribuições pode ser acompanhado diretamente pelos insights do GitHub, que mostram o envolvimento de cada integrante, frequência de commits, pull requests e revisões:

- **Insights Frontend**
- **Insights Backend**

Como estratégia de integração adotamos o GitHub Flow para organizar a integração do código. Cada funcionalidade ou correção é feita em uma branch separada, seguida pela abertura de uma pull request para revisão e evitar commits direto na main. Essa estratégia de Branching ajuda a manter a qualidade e o padrão do código, principalmente, com a implementação de um pipeline de CI.

Também usamos GitHub Actions para rodar testes automaticamente a cada alteração, garantindo que o código integrado esteja sempre funcional e sem erros. Essa abordagem torna o desenvolvimento mais ágil, seguro e colaborativo.

11.1 Participação Individual

Mikaell Miguel: Desenvolvimento do frontend e backend, implementação das APIs, testes automatizados, integração com banco de dados e configuração do pipeline de CI.

Sandrirames Albino Fausto: Auxílio no levantamento de requisitos, documentação do projeto e parte do deploy.

Pablo Henrique: Contribuição na pesquisa de tecnologias, design da interface e validação do wireframes no Figma.

12. Lições Aprendidas

O desenvolvimento do *FiscalizaDeputado* proporcionou diversos aprendizados técnicos e organizacionais, além de expor desafios que influenciaram diretamente a forma como o projeto foi conduzido.

Entre os principais desafios enfrentados, destacam-se questões relacionadas ao **deploy** da aplicação (dado a alta quantidade de registros da tabela despesas) e à utilização de uma **API de CNPJ**. Grande parte das APIs disponíveis para consulta de CNPJ são pagas ou possuem limitações severas de uso, o que inviabilizou sua integração na primeira versão do MVP. O **tempo reduzido** para execução do projeto também foi um fator limitante, afetando principalmente o desenvolvimento do frontend. A elaboração dessa camada apresentou dificuldades devido ao nível de conhecimento que a equipe possuía inicialmente, agravado pela disponibilidade limitada de alguns membros ao longo do processo.

Durante o desenvolvimento, ajustes significativos foram realizados para otimizar o fluxo de trabalho. Inicialmente, a API seria documentada no **Postman**, ferramenta já utilizada pela equipe para validar endpoints. Contudo, optou-se por migrar para o **Swagger**, pois, além de permitir a visualização interativa da documentação, ele se integra diretamente ao código, garantindo que a documentação esteja sempre atualizada e acessível a qualquer pessoa que utilize a API. Outro ajuste importante foi a alteração da forma de integração entre backend e frontend. No início, desenvolvia-se grande parte do backend antes de iniciar o frontend, o que resultava em mudanças significativas no backend para atender às necessidades descobertas posteriormente. Para evitar esse retrabalho, passou-se a adotar uma abordagem mais interativa: a cada funcionalidade implementada no backend, a correspondente no frontend era desenvolvida logo em seguida, permitindo identificar e corrigir ajustes de forma mais ágil.

Entre os aspectos que funcionaram bem, destacam-se os **testes unitários**, que ajudaram na detecção precoce de bugs, possibilitando correções rápidas e evitando que problemas se acumulassem nas etapas posteriores. Essa prática aumentou a confiabilidade do código e contribuiu para a estabilidade do sistema.

Por outro lado, alguns pontos poderiam ser melhorados. A **organização da equipe** e o **planejamento no GitHub Projects** necessitam de mais atenção, especialmente para garantir que tarefas sejam melhor distribuídas e priorizadas. Os **testes de integração** também deveriam ter sido considerados desde o início do projeto, o que permitiria validar a comunicação entre diferentes módulos de forma mais eficaz. Além disso, a **comunicação interna** apresentou falhas em alguns momentos, gerando desalinhamentos que impactaram prazos e execução de tarefas.

12.1 Relato Individual

Mikaell Miguel: Responsável por conduzir todo o desenvolvimento do frontend em React e do backend em Node.js/Express. Durante o projeto, aprendeu a implementar CI com GitHub Actions, garantindo execução automática de testes unitários, além de criar documentação interativa com Swagger, integrando-a diretamente aos endpoints da API. Também adquiriu experiência com deploy no Azure, lidando com configuração de container, otimização de performance e resolução de problemas relacionados a grande volume de dados. Aprimorou habilidades em planejamento de sprints, revisão de código e organização de branches usando GitHub Flow.

Pablo Henrique: Aprimorou habilidades em documentação e organização de artefatos do projeto, incluindo o registro de requisitos e elaboração de diagramas. Aprendeu sobre a importância de manter informações atualizadas e estruturadas para facilitar o desenvolvimento e a comunicação da equipe.

Sandrirames Fausto: Contribuiu com a criação de wireframes no Figma, estruturando layouts, fluxos de navegação e organização das interfaces. Desenvolveu capacidade de alinhar visualmente ideias com os requisitos do projeto, compreendendo a importância do design e da prototipagem para a experiência do usuário e para a comunicação com a equipe de desenvolvimento.

13. Conclusão

A solução entregue pelo FiscalizaDeputado se mostra alinhada ao problema inicialmente proposto, atendendo grande parte de seus objetivos centrais: facilitar o acesso aos dados de despesas parlamentares, promover a transparência e estimular o controle social. A aplicação cumpre o papel de oferecer uma plataforma acessível, com funcionalidades de visualização e filtragem, permitindo que qualquer cidadão acompanhe a utilização de recursos públicos. Apesar disso, reconhece-se que há espaço para evolução, especialmente na ampliação de funcionalidades e na integração com novos serviços.

Para o futuro, a visão do projeto inclui a implementação de uma consulta de CNPJ integrada ao Google Street View, permitindo que o usuário visualize não apenas informações sobre empresas relacionadas às despesas parlamentares, mas também sua localização física, agregando contexto geográfico e visual às análises. Outro ponto previsto é a adoção de práticas de observabilidade e um desenvolvimento orientado a dados, ainda que dentro das limitações de infraestrutura disponíveis, de forma a melhorar o monitoramento e a análise de uso da aplicação.

Além disso, há a necessidade de garantir a **responsividade** da aplicação, assegurando que a interface funcione adequadamente em diferentes dispositivos,

como smartphones, tablets e desktops, para proporcionar uma experiência de usuário fluida e acessível a todos.

Também está no planejamento a implementação de testes de integração, fundamentais para garantir a robustez da comunicação entre módulos do sistema, e a exploração da possibilidade de criar um chat com inteligência artificial capaz de responder perguntas com base nos dados dos deputados e nas despesas declaradas, tornando a interação com o sistema ainda mais intuitiva e acessível.

Do ponto de vista de experiência, o desenvolvimento em equipe foi marcado por aprendizados técnicos e de organização, exigindo colaboração constante e adaptação a imprevistos. Embora desafios como comunicação, gestão de tarefas e retrabalho tenham surgido, a capacidade de ajustar o processo e buscar soluções conjuntas demonstrou maturidade e comprometimento do grupo. O FiscalizaDeputado, mesmo em sua versão inicial, representa um passo concreto na direção de tornar a fiscalização política mais acessível, e seu potencial de impacto aumenta consideravelmente com as melhorias planejadas para as próximas versões.

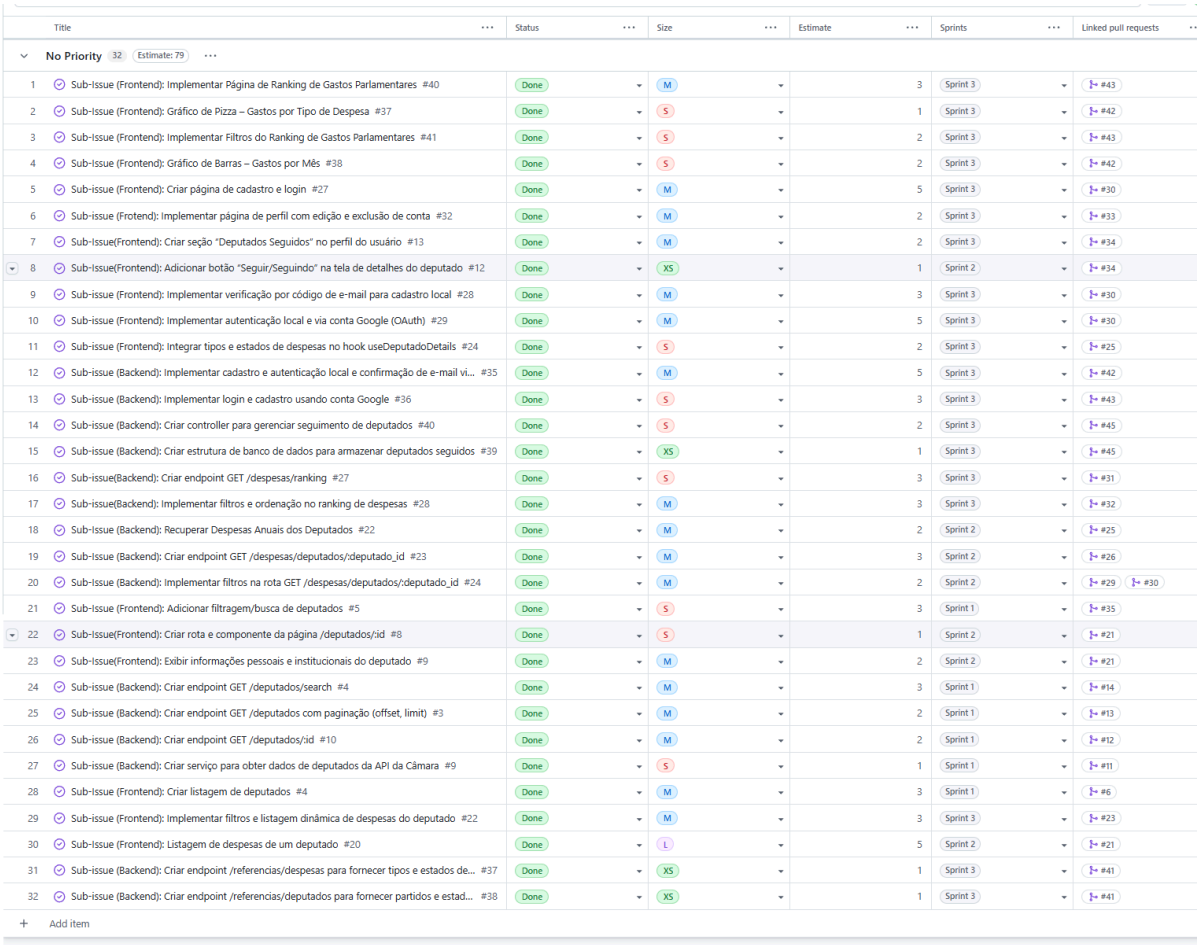
14. Anexos

14.1 Links úteis

- Jornada Típica do Usuário na aplicação - https://youtu.be/VFbpWc2Z9EY
- Deploy Frontend -https://fiscalizadeputado.vercel.app/
- Figma - Wireframes https://www.figma.com/design/uYwHi7ngPUnYge16BvIlr7/Projeto-ES2025.1
- Deploy Backend
    - Documentação: https://fiscalizadeputado.azurewebsites.net/api-docs/
    - Endpoint: https://fiscalizadeputado.azurewebsites.net/

- Repositório Backend:
    - https://github.com/mikaellmiguel/IF977-2025.1-BACKEND
- Repositório Frontend:
    - https://github.com/mikaellmiguel/IF977-2025.1-FRONTEND
- GithubProjects:
    - https://github.com/users/mikaellmiguel/projects/4

14.2 Quadros de Gestão - Github Project



	Title	Status	Size	Estimate	Sprints	Linked pull requests
	No Priority 32 Estimate: 79					
1	Sub-Issue (Frontend): Implementar Página de Ranking de Gastos Parlamentares #40	Done	M	3	Sprint 3	#43
2	Sub-Issue (Frontend): Gráfico de Pizza – Gastos por Tipo de Despesa #37	Done	S	1	Sprint 3	#42
3	Sub-Issue (Frontend): Implementar Filtros do Ranking de Gastos Parlamentares #41	Done	S	2	Sprint 3	#43
4	Sub-Issue (Frontend): Gráfico de Barras – Gastos por Mês #38	Done	S	2	Sprint 3	#42
5	Sub-issue (Frontend): Criar página de cadastro e login #27	Done	M	5	Sprint 3	#30
6	Sub-issue (Frotend): Implementar página de perfil com edição e exclusão de conta #32	Done	M	2	Sprint 3	#33
7	Sub-Issue(Frontend): Criar seção "Deputados Seguidos" no perfil do usuário #13	Done	M	2	Sprint 3	#34
8	Sub-Issue(Frontend): Adicionar botão "Seguir/Seguindo" na tela de detalhes do deputado #12	Done	XS	1	Sprint 2	#34
9	Sub-issue (Frontend): Implementar verificação por código de e-mail para cadastro local #28	Done	M	3	Sprint 3	#30
10	Sub-issue (Frontend): Implementar autenticação local e via conta Google (OAuth) #29	Done	M	5	Sprint 3	#30
11	Sub-issue (Frontend): Integrar tipos e estados de despesas no hook useDeputadoDetails #24	Done	S	2	Sprint 3	#25
12	Sub-issue (Backend): Implementar cadastro e autenticação local e confirmação de e-mail vi... #35	Done	M	5	Sprint 3	#42
13	Sub-issue (Backend): Implementar login e cadastro usando conta Google #36	Done	S	3	Sprint 3	#43
14	Sub-issue (Backend): Criar controller para gerenciar seguimento de deputados #40	Done	S	2	Sprint 3	#45
15	Sub-issue (Backend): Criar estrutura de banco de dados para armazenar deputados seguidos #39	Done	XS	1	Sprint 3	#45
16	Sub-issue(Backend): Criar endpoint GET /despesas/ranking #27	Done	S	3	Sprint 3	#31
17	Sub-issue(Backend): Implementar filtros e ordenação no ranking de despesas #28	Done	M	3	Sprint 3	#32
18	Sub-Issue (Backend): Recuperar Despesas Anuais dos Deputados #22	Done	M	2	Sprint 2	#25
19	Sub-Issue (Backend): Criar endpoint GET /despesas/deputados/:deputado_id #23	Done	M	3	Sprint 2	#26
20	Sub-Issue (Backend): Implementar filtros na rota GET /despesas/deputados/:deputado_id #24	Done	M	2	Sprint 2	#29 #30
21	Sub-Issue (Frontend): Adicionar filtragem/busca de deputados #5	Done	S	3	Sprint 1	#35
22	Sub-Issue(Frontend): Criar rota e componente da página /deputados/:id #8	Done	S	1	Sprint 2	#21
23	Sub-Issue(Frontend): Exibir informações pessoais e institucionais do deputado #9	Done	M	2	Sprint 2	#21
24	Sub-issue (Backend): Criar endpoint GET /deputados/search #4	Done	M	3	Sprint 1	#14
25	Sub-issue (Backend): Criar endpoint GET /deputados com paginação (offset, limit) #3	Done	M	2	Sprint 1	#13
26	Sub-issue (Backend): Criar endpoint GET /deputados/:id #10	Done	M	2	Sprint 1	#12
27	Sub-issue (Backend): Criar serviço para obter dados de deputados da API da Câmara #9	Done	S	1	Sprint 1	#11
28	Sub-Issue (Frontend): Criar listagem de deputados #4	Done	M	3	Sprint 1	#6
29	Sub-issue (Frontend): Implementar filtros e listagem dinâmica de despesas do deputado #22	Done	M	3	Sprint 3	#23
30	Sub-Issue (Frontend): Listagem de despesas de um deputado #20	Done	L	5	Sprint 2	#21
31	Sub-issue (Backend): Criar endpoint /referencias/despesas para fornecer tipos e estados de... #37	Done	XS	1	Sprint 3	#41
32	Sub-issue (Backend): Criar endpoint /referencias/deputados para fornecer partidos e estad... #38	Done	XS	1	Sprint 3	#41
	+ Add item					

Ressalta-se que há outros quadros, no entanto, não ficam com uma qualidade aceitável no documento dando a sua altura.