AposentAI

Anita Monteiro (amspb)
Lucas Rodrigues (lrsj)
Lucas Monterazo (lsm6)

Recife, 10/08/2025

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

Sumário

1. Introdução 2
2. Contexto e Problema 2
3. Público-Alvo e Personas 2
4. Proposta de Valor e Solução 2
5. MVP e Funcionalidades 2
6. Modelagem e Design 3
7. Processo de Desenvolvimento 3
8. Garantia de Qualidade e Segurança 3
9. Arquitetura Técnica e Implementação 3
10. Implantação e DevOps 3
11. Colaboração e Versionamento 4
12. Lições Aprendidas 4
13. Conclusão 4
14. Anexos 4

1. Introdução

O processo de aposentadoria no Brasil, especialmente após a Reforma da Previdência (EC 103/2019), tornou-se um desafio tanto para segurados quanto para profissionais da área jurídica. A complexidade das novas regras de transição, aliada à multiplicidade de cálculos e cenários possíveis, exige análises personalizadas e dificulta a compreensão dos direitos e do momento mais vantajoso para se aposentar. Essa realidade afeta não apenas os cidadãos, mas também advogados previdenciaristas, que enfrentam grande volume de demandas, cálculos extensos e constantes mudanças legislativas.

Diante desse cenário, propõe-se o desenvolvimento de uma solução tecnológica com o objetivo principal de simplificar o entendimento das normas, automatizar cálculos e oferecer uma interface intuitiva e confiável, promovendo agilidade, padronização e segurança no processo. A escolha desse tema se justifica por sua relevância social e pelo alto potencial de impacto positivo, considerando o crescimento da população idosa e a demanda reprimida por planejamento previdenciário eficiente.

Por fim, para o desenvolvimento da solução, adotou-se uma metodologia híbrida, combinando Scrum (com reuniões semanais de alinhamento para planejamento, acompanhamento e revisão do projeto), além do Kanban, que foi utilizado para organizar as tarefas, visualizar o fluxo de trabalho e priorizar as entregas de forma dinâmica.

2. Contexto e Problema

O problema central abordado é a falta de clareza, previsibilidade e suporte ao longo da jornada de aposentadoria no Brasil, especialmente no entendimento das regras do INSS, nas simulações de tempo e valor de benefício, e na tomada de decisão sobre o momento e o regime mais vantajosos para se aposentar. A Reforma da Previdência (EC 103/2019) introduziu múltiplas regras de transição, ampliando a complexidade dos cálculos e tornando indispensável a análise comparativa personalizada para cada segurado.

A relevância e o impacto dessa questão são reforçados por dados obtidos em levantamento com mais de 40 entrevistados, dos quais a maioria relatou dificuldades significativas para compreender a legislação previdenciária e ausência de ferramentas confiáveis e intuitivas para simulações. Muitos conduziram o processo sozinhos, com insegurança, ou descobriram seus direitos tardiamente. Mesmo com a existência do site e aplicativo Meu INSS, usuários apontam limitações na clareza, na usabilidade e na abrangência das funcionalidades, o que compromete o planejamento previdenciário.

O problema também é evidente no cotidiano dos advogados previdenciaristas, que lidam com grande volume de clientes e precisam executar cálculos longos, suscetíveis a erros, além de se atualizar constantemente frente às mudanças legislativas. A falta de sistemas integrados obriga o uso de planilhas manuais e consultas fragmentadas, gerando perda de produtividade e maior risco de inconsistências.

O domínio de aplicação do projeto concentra-se no setor previdenciário, inicialmente voltado para escritórios de advocacia, que poderão utilizar a ferramenta para agilizar atendimentos, padronizar análises e oferecer um serviço mais preciso e seguro. Nesse contexto, uma plataforma com inteligência artificial, capaz de traduzir a legislação para linguagem acessível, realizar simulações personalizadas e oferecer suporte contínuo, apresenta alto potencial de impacto positivo para profissionais previdenciaristas.

3. Público-Alvo e Personas

O produto é voltado inicialmente para advogados previdenciaristas e profissionais que atuam diretamente com aposentadoria, contemplando tanto escritórios de advocacia especializados em previdência social quanto advogados autônomos que realizam requerimentos administrativos ou ações judiciais previdenciárias. Também fazem parte desse público profissionais de Recursos Humanos, consultores internos de órgãos públicos, contadores e representantes de sindicatos que prestam suporte a trabalhadores sobre aposentadoria.

Esses profissionais enfrentam dores recorrentes, como o tempo excessivo gasto com tarefas repetitivas — por exemplo, compilar documentos, montar pastas e preencher requerimentos no site do Meu INSS — e a falta de ferramentas confiáveis para realizar simulações precisas, considerando a legislação previdenciária atualizada.

Além disso, a demora e inconsistência das análises do INSS exigem preparação minuciosa e reforçam a necessidade de sistemas que reduzam erros e aumentem a produtividade.

Entre os atributos mais valorizados por esse público estão: a automatização da organização documental, checklists integrados, geração de relatórios simulados com base em múltiplas regras, integração com requerimentos do Meu INSS, economia de tempo sem abrir mão da segurança jurídica, e uma interface robusta, confiável e capaz de exportar relatórios e acompanhar processos.

A nossa persona é a Dra. Deolane Bezerra, uma advogada previdenciária sênior com 12 anos de experiência, que atua em um escritório de médio porte com presença nacional na Região Sudeste do Brasil. Ela lidera a área previdenciária do escritório, atendendo principalmente servidores públicos e autônomos. Sua rotina

diária envolve diversas atividades, como atendimento direto aos clientes, revisão e organização documental, elaboração de pareceres jurídicos e simulação, além do preenchimento de requerimentos no portal Meu INSS e acompanhamento dos processos judiciais. Também presta consultorias para empresas e sindicatos, o que exige atualização constante diante das mudanças legislativas.

Entre as principais dificuldades enfrentadas pela Dra. Deolane estão o excesso de tarefas manuais e repetitivas, como a montagem de pastas, análise detalhada de vínculos e a formatação dos requerimentos administrativos. Ela sente a falta de uma ferramenta unificada que possa integrar as simulações, a gestão documental e a submissão automática dos requerimentos ao INSS, o que tornaria seu trabalho mais ágil e menos suscetível a erros. Além disso, a complexidade das regras previdenciárias, que mudam frequentemente, gera incertezas e a necessidade de dividir casos mais técnicos com colegas ou parceiros especializados, aumentando a dependência e dificultando a padronização na geração de relatórios confiáveis para clientes e tribunais.

Motivada a melhorar sua produtividade e oferecer um serviço diferenciado, a Dra. Deolane busca uma plataforma que automatize a triagem documental, organize pastas automaticamente e permita realizar simulações múltiplas de aposentadoria para comparar regras por idade, tempo de contribuição, média salarial ou paridade. Ela deseja ainda que a ferramenta preencha os requerimentos administrativos no Meu INSS de forma automática, envie alertas sobre mudanças legais que afetem seus processos e possibilite o acompanhamento do andamento das aposentadorias. Para ela, a segurança dos dados, a interface clara com linguagem jurídica adequada, o suporte técnico constante e a possibilidade de exportar relatórios completos são requisitos essenciais. Como ela própria destaca, “casos simples eu mesma resolvo, mas nos complexos preciso de um sistema robusto. Se eu pudesse automatizar a montagem da pasta do cliente e preencher o requerimento com um clique, economizaria horas.

4. Proposta de Valor e Solução

Nosso produto representa uma solução inovadora ao automatizar as rotinas técnicas e operacionais da advocacia previdenciária, com um foco inicial em profissionais que atuam na elaboração de requerimentos administrativos e judiciais de aposentadoria. A inovação está na integração de diversas funcionalidades que hoje se encontram dispersas em diferentes ferramentas (como simuladores, geradores de documentos em PDF e checklists de documentos) reunidas em uma única plataforma especializada para advogados previdenciários. Além disso, utilizamos inteligência artificial para analisar cenários complexos de aposentadoria, gerar recomendações personalizadas e preencher automaticamente os requerimentos com base no perfil do cliente, promovendo

um fluxo de trabalho guiado, seguro e padronizado. Isso transforma uma operação tradicionalmente manual, burocrática e sujeita a erros em um processo mais eficiente, com ganhos significativos em produtividade e redução de retrabalho.

Comparado a outras soluções disponíveis, nosso produto apresenta diferenciais claros. Enquanto o portal Meu INSS oferece apenas funcionalidades limitadas e simuladores jurídicos atuais fazem simulações básicas, nossa plataforma realiza simulações múltiplas com comparação detalhada de regras, organiza automaticamente a documentação dos clientes, gera relatórios completos e customizáveis em formatos como PDF e DOCX, preenche automaticamente requerimentos no sistema oficial e mantém foco total nas necessidades dos advogados, oferecendo parecer jurídico integrado e alertas legislativos personalizados. Esses recursos são fundamentais para suprir as lacunas existentes no mercado, tornando o trabalho dos profissionais mais ágil, preciso e seguro.

Os principais benefícios oferecidos pelo produto são claros e abrangem diferentes atores do ecossistema previdenciário. Para os advogados, proporciona economia de tempo, padronização dos processos, aumento da produtividade e um diferencial competitivo no mercado jurídico. Para os clientes finais, resulta em decisões mais informadas, acesso facilitado ao processo de aposentadoria e maior segurança jurídica. Já para o próprio ecossistema previdenciário, contribui para a desburocratização das interações com o INSS e eleva a qualidade dos requerimentos submetidos, reduzindo erros e retrabalho institucional. Dessa forma, a solução se posiciona como uma ferramenta estratégica e inovadora, capaz de transformar a forma como a advocacia previdenciária é conduzida no Brasil.

5. MVP e Funcionalidades

Nosso MVP (Minimum Viable Product) foi cuidadosamente delineado para validar a proposta de valor focada na rotina do advogado previdenciário, priorizando funcionalidades que automatizam tarefas críticas e agregam valor imediato ao escritório. Desde os estágios iniciais, passamos de uma ideia genérica sobre planejamento previdenciário com inteligência artificial para um produto com foco cirúrgico nas necessidades do profissional, sustentado por pesquisas quantitativas, entrevistas e validações reais com usuários.

As funcionalidades centrais implementadas na primeira versão do MVP incluem: simulação comparativa de regras de aposentadoria, upload e organização automática de documentos, geração de relatórios personalizados em PDF/DOCX, e preenchimento automático dos requerimentos do Meu INSS (em fase de desenvolvimento). Esses recursos foram priorizados por entregarem ganhos claros de produtividade, segurança jurídica e padronização, facilitando o fluxo de

trabalho do advogado desde a análise inicial até a entrega do parecer. Além disso, a gestão de múltiplos clientes e acesso rápido às suas pastas individuais compõe a base operacional para organização e acompanhamento eficiente dos processos.

Quanto ao escopo e prototipagem, o MVP está bem definido, com backlog estruturado em user stories claras, criteriosamente alinhadas ao modelo INVEST, garantindo funcionalidades independentes, testáveis e de alto valor para o usuário. A interface está sendo desenvolvida em tecnologias modernas (Vite.js e FastAPI) e já conta com protótipos navegáveis, facilitando iterações rápidas e adaptações conforme feedbacks. Funcionalidades adicionais, como notificações legislativas personalizadas, estão planejadas para versões futuras, assegurando um ciclo contínuo de evolução e escalabilidade do produto.

6. Modelagem e Design

O desenvolvimento do AposentAI demandou atenção especial à sua modelagem e ao seu design arquitetural, visando garantir robustez, escalabilidade e clareza estrutural. Para isso, optou-se por um conjunto de modelos e diagramas que, de forma integrada, descrevem o sistema sob diferentes perspectivas: desde a organização dos dados até a representação das suas funcionalidades em operação.

Esta seção é o nosso guia pelos bastidores da construção do AposentAI.

6.1 Modelagem de Dados: O Diagrama Entidade-Relacionamento (ER)

A estrutura de dados é o ponto de partida para a organização lógica do AposentAI. O Diagrama Entidade-Relacionamento (ER) foi elaborado para representar de forma clara as entidades centrais — como **Usuários**, **Casos** e **Simulações** — e as relações entre elas.

A escolha do **PostgreSQL** como banco de dados relacional ocorreu devido à sua capacidade de garantir segurança, consistência e integridade para o armazenamento de informações sensíveis.

O Diagrama ER orientou a implementação do banco de dados, permitindo:

- **Organização e integridade**: eliminação de redundâncias e prevenção de informações duplicadas.

- **Alinhamento com as regras de negócio**: por exemplo, definindo como um `Advogado` se associa a um `Caso` de um `Usuário Comum`.

- **Validação de relacionamentos**: assegurando que os vínculos entre tabelas sejam consistentes e adequados.

O modelo de dados completo encontra-se documentado em:

Diagrama modelo ER e Modelo C4

6.2 Arquitetura de Software: O Modelo C4

O AposentAI é um sistema complexo, cuja compreensão integral exige diferentes níveis de abstração. Para representar sua arquitetura, foi adotado o **Modelo C4**, que permite visualizar o sistema de forma hierárquica, passando do panorama geral até a descrição dos componentes internos.

Nível 1 – Contexto: visão macro do sistema, destacando suas interações com atores externos — **Usuário Comum, Advogado e Administrador** — e sistemas complementares, como gateways de pagamento, serviços de e-mail, a API do INSS e o motor de IA baseado em **LangChain**.

Nível 2 – Contêineres: detalhamento dos principais blocos do sistema, incluindo:

- **Frontend (SPA)**: interface web desenvolvida com **Vite.js**.

- **Backend (API)**: implementado em **FastAPI (Python)**, responsável pela lógica de negócio.

- **Banco de Dados**: instância **PostgreSQL** para armazenamento de dados.

- **Serviços de IA**: integração com a LangChain.

Nível 3 – Componentes: aprofundamento em cada contêiner, mapeando módulos internos, como:

- Autenticação

- Gestão de Usuários

- Simulação (conexão com LangChain)

- Pagamentos
- Repositórios de acesso a dados

6.3 Modelagem de Processos e Fluxos

Além da estrutura de dados e da arquitetura, foi necessário representar a dinâmica de funcionamento do sistema. Para isso, utilizaram-se diagramas adicionais que ilustram o comportamento e a sequência das operações.

- **Fluxo do Usuário (User Flow):** descreve a jornada de interação do usuário, como no caso do processo de simulação de aposentadoria — desde o login, passando pelo upload de documentos, até a visualização do resultado final.

- **Diagrama de Sequência:** demonstra as trocas de informações entre componentes durante a execução de uma função, como o processo de login, no qual o **Frontend**, o **Backend** e o **Banco de Dados** interagem para validar credenciais e emitir um token JWT.

7. Processo de Desenvolvimento

Para o desenvolvimento do nosso produto, adotamos uma metodologia híbrida que combina Scrum e Kanban, adequando-se às necessidades específicas da equipe e do projeto. Optamos pelo Scrum como base para o planejamento e acompanhamento do progresso, realizando reuniões semanais de alinhamento para definir prioridades, revisar entregas e ajustar o rumo conforme o feedback. Paralelamente, usamos Kanban para organizar as demandas diárias, monitorar as tarefas em andamento e garantir a transparência do fluxo de trabalho, facilitando a gestão visual das atividades.

Devido à composição enxuta da equipe, com apenas três integrantes, todos desempenharam múltiplas funções, acumulando papéis de desenvolvedores, testers e até gestores das demandas, sem a presença de um Product Owner (PO) exclusivo. Essa flexibilidade exigiu comunicação constante e colaboração para distribuir responsabilidades e manter o ritmo produtivo. As principais ferramentas adotadas para suportar o processo foram o Figma para prototipagem e design da interface, o GitHub para controle de versão e gestão do código, o Lovable para o desenvolvimento frontend, e o Discord para comunicação rápida e integração entre os membros.

O roadmap de desenvolvimento foi estruturado em fases bem definidas para garantir entregas incrementais e aprendizado contínuo. Na Fase 1 (0-3 meses), lançamos o MVP focado em funcionalidades centrais como upload e organização

automática de documentos, simulação comparativa de aposentadorias, geração de relatórios personalizados e preenchimento automatizado de requerimentos do Meu INSS, além do gerenciamento de múltiplos clientes. Durante essa etapa, foram realizadas implementações, testes com usuários reais e coleta de feedbacks para aprimorar a solução. A Fase 2 (4-6 meses) prevê a expansão do produto com notificações legislativas, integração com plataformas de terceiros e suporte ao cliente, seguindo um desenvolvimento iterativo apoiado por workshops e estratégias de marketing. As fases seguintes contemplam a melhoria contínua da experiência do usuário, a ampliação do público-alvo para pessoas físicas, e, finalmente, a escalabilidade e sustentabilidade do produto, incluindo análises avançadas e modelos de monetização. Esse planejamento permite uma evolução estruturada, alinhada às necessidades reais dos usuários e ao mercado.

8. Garantia de Qualidade e Segurança

A qualidade e a segurança são elementos centrais no desenvolvimento do AposentAI. Considerando que a aplicação manipula informações previdenciárias e dados pessoais sensíveis, foi definida uma estratégia abrangente de qualidade que se estende desde a concepção e escrita do código até a sua implantação em produção.

A abordagem adotada para testes segue a lógica da “Pirâmide de Testes”, priorizando uma base sólida de testes unitários, complementada por testes de integração e, no topo, por testes de sistema do tipo End-to-End (E2E). Essa estrutura garante rapidez no retorno de feedback aos desenvolvedores e eleva o grau de confiabilidade nas entregas. Todos os testes são automatizados e executados continuamente por meio do pipeline de Integração Contínua (CI/CD), configurado no GitHub Actions, acionado a cada envio de código (`push`) ou solicitação de integração (`Pull Request`). Alterações que resultem na quebra de testes existentes não são integradas à ramificação principal do projeto. Além disso, práticas como Desenvolvimento Guiado por Testes (TDD) e Desenvolvimento Guiado por Comportamento (BDD) são estimuladas, utilizando os critérios de aceitação definidos nas histórias de usuário como referência.

Para o backend, desenvolvido em Python com o framework FastAPI, a suíte de testes emprega o `Pytest` como ferramenta principal, o `HTTPX` para chamadas assíncronas aos endpoints durante testes de integração e o `pytest-cov` para mensuração da cobertura de código. No frontend, implementado em Vite.js e TypeScript, utilizam-se o `Vitest` para testes unitários e de componentes, o `Testing Library` para validação de comportamento e acessibilidade, e o `Playwright` para simulações completas de uso no navegador (E2E).

Quanto aos tipos de testes, destacam-se:

- **Testes unitários**, voltados à menor unidade de código isolada, como funções de cálculo de tempo de contribuição ou componentes de interface.

- **Testes de integração**, que validam a interação entre diferentes módulos, como a chamada a um endpoint de criação de usuário, a validação de dados com Pydantic, o processamento no serviço e o registro no banco de dados.

- **Testes de sistema (E2E)**, que percorrem fluxos completos de uso, por exemplo: cadastro, login, envio de documento CNIS e obtenção do resultado da simulação.

A garantia de qualidade também é reforçada pelo uso de ferramentas de análise estática, que atuam antes mesmo da execução dos testes. No backend, empregam-se `Black` e `isort` para formatação e organização de imports, `Ruff` como linter rápido e rigoroso, `MyPy` para verificação estática de tipos e `Bandit` para detecção de vulnerabilidades comuns. No frontend, o `ESLint` e o `Prettier` asseguram padronização de estilo e boas práticas no código JavaScript/TypeScript.

No que se refere à segurança, o AposentAI foi concebido desde o início em conformidade com a Lei Geral de Proteção de Dados (LGPD), garantindo consentimento explícito dos usuários e uso restrito das informações para as finalidades previstas. O controle de acesso baseia-se em autenticação e autorização robustas: as senhas são armazenadas apenas em formato de hash seguro utilizando o algoritmo Bcrypt, via biblioteca `passlib`; a autenticação de sessão é realizada com tokens JWT assinados, de curta duração; e a autorização é implementada por meio de RBAC (controle de acesso baseado em papéis), assegurando que funcionalidades restritas a advogados ou administradores não possam ser acessadas por usuários comuns.

Em termos de prevenção a vulnerabilidades, a aplicação segue as diretrizes do OWASP Top 10. Todas as entradas de dados são validadas rigorosamente pelos schemas do Pydantic, prevenindo ataques de injeção; chaves, tokens e URLs sensíveis são gerenciados por variáveis de ambiente, não sendo expostos no código-fonte; e toda a comunicação entre cliente e servidor é criptografada via HTTPS (TLS).

No nível de infraestrutura, documentos enviados pelos usuários são armazenados com criptografia em repouso e acessados exclusivamente por meio de URLs assinadas com prazo de expiração limitado, reforçando a proteção contra acessos indevidos.

9. Arquitetura Técnica e Implementação

Esta seção apresenta a descrição dos componentes técnicos, metodologias e decisões que compõem a base estrutural do projeto AposentAI. O objetivo foi selecionar tecnologias e padrões que garantissem não apenas a performance e a segurança da aplicação, mas também a produtividade da equipe e a facilidade de manutenção do código a longo prazo.

9.1. Linguagens, Frameworks e Bibliotecas Utilizadas

A definição da stack tecnológica priorizou a construção de um sistema moderno, reativo e escalável, apoiado por um ecossistema sólido e com ampla documentação.

Backend

- **Linguagem:** Python 3.11+
- **Framework Web:** FastAPI – possibilita a criação de APIs de alta performance, com suporte assíncrono nativo.
- **ORM e Migrations:** SQLAlchemy e Alembic – permitem o mapeamento objeto-relacional e o controle de versionamento do esquema de banco de dados.
- **Validação de Dados:** Pydantic – integrado ao FastAPI para validação, serialização e documentação automática de modelos de dados.
- **Autenticação:** Passlib com Bcrypt – para hashing seguro de senhas.
- **Inteligência Artificial:** LangChain – responsável por orquestrar interações com modelos de linguagem e compor cadeias de simulação.

Frontend

- **Linguagem:** JavaScript (TypeScript)
- **Framework/Build Tool:** Vite.js – garante um ambiente de desenvolvimento rápido e otimizado.

- **Framework de UI:** React ou Vue.js – para construção de Single-Page Applications (SPA) reativas e componentizadas.

- **Estilização:** Windy CSS ou Tailwind CSS – abordagem *utility-first* para criação ágil de interfaces customizadas.

Banco de Dados

- **SGBD:** PostgreSQL – sistema de banco de dados relacional robusto, confiável e com suporte a tipos de dados avançados, como `JSONB`.

Ferramentas e DevOps

- **Controle de Versão:** Git e GitHub.

- **Containerização:** Docker e Docker Compose – criam um ambiente de desenvolvimento consistente e facilitam o processo de *deploy*.

9.2. Estrutura Geral do Projeto

O projeto foi estruturado em um mono repositório, permitindo gerenciar o código do backend e do frontend de forma unificada.

```
/aposentai
├── backend/
│   ├── app/
│   │   ├── models.py      # Definições das tabelas (SQLAlchemy)
│   │   ├── schemas.py     # Modelos de dados da API (Pydantic)
│   │   ├── repositories.py # Camada de acesso aos dados
│   │   ├── services/      # Lógica de negócio
│   │   └── routers/       # Endpoints da API
│   ├── alembic/           # Migrations do banco de dados
├── frontend/
│   ├── src/
│   │   ├── components/    # Componentes de UI reutilizáveis
│   │   ├── views/         # Páginas da aplicação
│   │   ├── router/        # Configuração das rotas
└── docs/                  # Documentação do projeto
```

Backend:

- `models` – estrutura do banco de dados.
- `schemas` – contratos de dados da API.
- `repositories` – camada de acesso ao banco.
- `services` – lógica de negócio.
- `routers` – endpoints HTTP.

Frontend:

- `views` – páginas completas.
- `components` – blocos reutilizáveis de interface.

9.3. Estilos e Padrões Arquiteturais

Estilo Arquitetural – Monolito Modular
Adotou-se a abordagem de monolito modular, combinando a simplicidade de desenvolvimento e *deploy* de um monolito com a organização interna modular. Essa estrutura reduz a complexidade operacional e mantém baixo acoplamento entre componentes, permitindo migração futura para microsserviços, se necessário.

Influência da Arquitetura Hexagonal (Portas e Adaptadores)
A lógica de negócio, concentrada nos `services`, é isolada de preocupações externas, como persistência de dados e integrações externas. Essa separação é implementada por meio de interfaces (*portas*) e classes concretas (*adaptadores*), aumentando a testabilidade e a flexibilidade.

Padrões de Design Utilizados:

- **Repository Pattern** – abstrai o acesso a dados e desacopla a lógica de negócio do ORM.
- **Injeção de Dependência** – utilizada pelo FastAPI para fornecer dependências, como sessão de banco de dados e repositórios, de forma

desacoplada.

- **Strategy Pattern** – planejado para o módulo de autenticação, permitindo alternar entre métodos de login (local ou via terceiros).

9.4. Justificativas das Decisões Técnicas

- **Adoção do FastAPI no Backend:** escolhido por sua alta performance, suporte assíncrono, produtividade e integração com bibliotecas de IA.
- **Uso do PostgreSQL:** necessário pela consistência, suporte a transações ACID e capacidade de armazenar dados semi-estruturados via `JSONB`.
- **Arquitetura Monolito Modular:** reduz complexidade inicial, mas mantém flexibilidade para expansão.

9.5. Repositório de Código-Fonte

O código-fonte completo do projeto, incluindo módulos, documentação e testes, encontra-se no repositório oficial:
https://github.com/Monterazo/AposentAI

10. Implantação e DevOps

A estratégia de Implantação e DevOps para o AposentAI foi projetada com foco em automação, consistência e confiabilidade. O objetivo é permitir entregas contínuas, seguras e previsíveis, minimizando o trabalho manual e garantindo que o ambiente de produção seja estável.

10.1. Estratégia de Deploy e Execução

A aplicação é dividida em três componentes principais (Frontend, Backend, Banco de Dados), cada um com uma estratégia de deploy específica para otimizar performance, segurança e custo.

- **Banco de Dados (PostgreSQL):** Para ambientes de produção, o banco de dados será hospedado em um serviço gerenciado de nuvem (Managed Database). Esta abordagem terceiriza a complexidade de manutenção, backups, escalabilidade e segurança.
    - **Opções de Provedores:** Amazon RDS for PostgreSQL, Google Cloud SQL, ou serviços mais simples como Railway ou Supabase.

    - **Execução:** A aplicação backend se conectará ao banco de dados através de uma URL de conexão segura, gerenciada por variáveis de ambiente.
- **Backend (API FastAPI):** O backend será empacotado em um container Docker e implantado em uma plataforma como serviço (PaaS) ou um serviço de orquestração de containers. A aplicação será executada como um processo stateless, o que facilita a escalabilidade horizontal.
    - **Opções de Provedores:** Render, Heroku, ou serviços mais robustos como AWS Elastic Container Service (ECS) ou Google Cloud Run.
    - **Execução:** O provedor de nuvem será responsável por baixar a imagem Docker, executar o container e expor a API para a internet, gerenciando também o escalonamento e o monitoramento da saúde da aplicação.
- **Frontend (Aplicação Vite.js):** Após o processo de build (`npm run build`), o frontend se torna um conjunto de arquivos estáticos (HTML, CSS, JS). A melhor estratégia é implantá-lo em uma plataforma de hospedagem de sites estáticos com CDN (Content Delivery Network) integrada.
    - **Opções de Provedores:** Vercel ou Netlify.
    - **Execução:** Estas plataformas otimizam a entrega dos arquivos para o usuário final, garantindo baixa latência globalmente, e se integram perfeitamente com o repositório no GitHub para deploys automáticos a cada `push`.

10.2. Uso de Containers (se aplicável)

O uso de containers com **Docker** é fundamental na nossa estratégia.

- **Por que Docker?**
    - **Consistência de Ambiente:** Garante que a aplicação se comporte da mesma forma no ambiente de desenvolvimento, testes e produção, eliminando o clássico problema de "na minha máquina funciona".
    - **Isolamento e Portabilidade:** Empacota a aplicação e todas as suas dependências (versão do Python, bibliotecas, etc.) em uma imagem auto contida que pode ser executada em qualquer lugar onde o Docker esteja instalado.
- **Como Usamos no AposentAI:**
    - **`Dockerfile` para o Backend:** Um `Dockerfile` define passo a passo como construir a imagem da nossa API FastAPI, desde a instalação do Python até a execução do servidor Uvicorn.
    - **`docker-compose.yml` para Desenvolvimento:** Utilizamos o Docker Compose para orquestrar o ambiente de desenvolvimento local. Com um único comando (`docker-compose up`), subimos o container da API e o container do banco de dados PostgreSQL, já conectados e prontos para uso.

10.3. Pipeline de CI/CD (Descrição e Link)

Planejamos um pipeline de Integração Contínua e Entrega Contínua (CI/CD) utilizando **GitHub Actions** para automatizar todo o processo de teste e implantação.

- **Descrição do Pipeline:**
  1. **Gatilho (Trigger):** O pipeline é acionado a cada `push` na branch `main` ou na abertura de um `Pull Request`.
  2. **Integração Contínua (CI):**
     - **Linting & Testes:** O código é verificado contra regras de estilo (lint) e a suíte de testes automatizados (`pytest`) é executada. Se qualquer teste falhar, o pipeline é interrompido.
     - **Build:** Se os testes passarem, o pipeline constrói os artefatos: a imagem Docker do backend e os arquivos estáticos do frontend.
  3. **Entrega Contínua (CD):**
     - **Push da Imagem:** A imagem Docker do backend é enviada para um registro de containers (ex: GitHub Container Registry).
     - **Deploy:** O pipeline aciona os webhooks de deploy das plataformas de nuvem. O Render/Heroku baixa a nova imagem e atualiza a API, enquanto o Vercel/Netlify publica os novos arquivos do frontend.

10.4. Documentação de Build e Execução (Resumo do `README.md`)

Para executar o projeto localmente, um desenvolvedor deve seguir os seguintes passos:

- **Pré-requisitos:**
  1. Git
  2. Docker e Docker Compose
  3. Python 3.11+
  4. Node.js e npm
- **Passos para Execução Local:**
  1. Clonar o repositório: `git clone Monterazo/AposentAI`
  2. Configurar variáveis de ambiente: Copiar `.env.example` para `.env` e preencher os valores.
  3. Iniciar o ambiente Docker: `docker-compose up -d` (isso iniciará o banco de dados PostgreSQL).
  4. Preparar o Backend:
     - Navegar para a pasta `backend/`.

    - Instalar dependências: `pip install -r requirements.txt`.
    - Aplicar as migrações do banco: `alembic upgrade head`.
    - Iniciar o servidor: `uvicorn app.main:app --reload`.
5. Preparar o Frontend:
    - Navegar para a pasta `frontend/`.
    - Instalar dependências: `npm install`.
    - Iniciar o servidor de desenvolvimento: `npm run dev`.

11. Colaboração e Versionamento

A colaboração eficaz e o versionamento disciplinado foram pilares centrais para a entrega de um produto coeso. Utilizando Git e GitHub como base, a equipe adotou um fluxo de trabalho que combinava rastreabilidade, transparência e integração contínua.

O código foi organizado em um **monorepositório**, reunindo backend, frontend e documentação. Essa abordagem garantiu evolução sincronizada das diferentes camadas do sistema, com diretórios bem definidos (`/backend`, `/frontend`, `/docs` e `/.github`) e versionamento conjunto da documentação.

A divisão de responsabilidades foi clara:

- **Anita Monteiro (Full Stack Developer)** desenvolveu o núcleo da gestão de usuários, segurança (hashing, JWT) e modelos de dados, além de integrar frontend e backend.

- **Lucas Rodrigues (UX/UI Designer & Frontend Developer)** criou a identidade visual e traduziu designs em componentes responsivos, focando na simplificação da experiência do usuário.

- **Lucas Monterazo (Machine Learning Engineer & Project Lead)** estruturou a pipeline de IA e liderou a equipe, cuidando de extração de dados, engenharia de prompts e infraestrutura.

Para evitar instabilidade, utilizou-se um **Git Flow simplificado**, com `main` como branch estável, desenvolvimento em branches de feature e integração via Pull Requests obrigatoriamente revisados e validados por CI. Essa rotina garantiu que todo código fosse analisado, testado e aprovado antes de chegar à produção.

12. Lições Aprendidas

Ao término do ciclo de desenvolvimento do AposentAI, foi conduzida uma retrospectiva abrangente, com o objetivo de registrar os principais aprendizados, dificuldades enfrentadas e sucessos obtidos ao longo do projeto. Esse exercício de análise pós-implementação constitui um ativo importante para futuros trabalhos, pois fornece uma base de referências práticas para evitar erros repetidos e replicar soluções bem-sucedidas.

Desafios enfrentados

O obstáculo técnico mais significativo esteve na extração de dados dos arquivos PDF do Cadastro Nacional de Informações Sociais (CNIS). A ausência de um layout padronizado comprometeu a eficiência da extração direta de texto, resultando em inconsistências que afetaram a precisão das simulações previdenciárias. Outro desafio relevante foi a obtenção de respostas consistentes e estruturadas da inteligência artificial, que, nas primeiras iterações, retornava formatos incorretos ou informações inventadas. No frontend, o crescimento da aplicação evidenciou a complexidade do gerenciamento de estado global, exigindo soluções mais robustas para manter coerência nos dados e responsividade na interface.

Ajustes implementados

Para contornar as dificuldades com PDFs, a equipe substituiu a abordagem inicial por uma solução híbrida, combinando OCR (PyTesseract) e expressões regulares para detecção e extração de padrões. No backend, criou-se uma camada intermediária para validação rigorosa das respostas do modelo de linguagem, garantindo que estivessem em formato JSON válido e que respeitassem os tipos de dados esperados. Além disso, a gestão de estado no frontend foi centralizada por meio de uma biblioteca dedicada, simplificando o fluxo de dados e reduzindo a duplicação de lógica entre componentes.

Aspectos que funcionaram bem

A elaboração prévia da arquitetura (C4 Model), do modelo de dados (ERD) e do planejamento de sprints demonstrou-se um investimento de alto retorno, assegurando clareza de direção técnica. A escolha da pilha tecnológica — FastAPI, Python e LangChain no backend, e Vite.js no frontend — revelou-se acertada, permitindo alto desempenho e produtividade. Outro ponto positivo foi a adoção de um pipeline de Integração e Entrega Contínua (CI/CD) desde o início, automatizando testes e deploys, o que reduziu riscos e aumentou a confiança nas entregas. Por fim, a sinergia entre os membros da equipe, com papéis bem definidos e comunicação constante, favoreceu a execução coordenada.

Oportunidades de melhoria

A mitigação de riscos técnicos poderia ter sido abordada mais cedo, especialmente no que se refere à extração de dados do PDF, com a realização de uma Prova de Conceito aprofundada antes do início das sprints regulares. Além

disso, embora os testes unitários e de integração tenham recebido forte atenção, uma cobertura mais abrangente de testes End-to-End desde as primeiras fases poderia ter identificado falhas de fluxo de usuário de forma antecipada.

Depoimentos individuais

- *Anita Monteiro (Full Stack Developer)* destacou a importância de um contrato de API bem definido para integrar de forma segura e fluida o frontend aos múltiplos serviços do backend, ressaltando o papel do FastAPI e do Pydantic para alcançar esse objetivo.

- *Lucas Rodrigues (UX/UI Designer & Frontend Developer)* apontou como maior aprendizado o desafio de transformar informações previdenciárias complexas em uma interface limpa e intuitiva, reforçando a importância do feedback visual em todas as etapas.

- *Lucas Monterazo (Machine Learning Engineer & Líder do Projeto)* enfatizou o caráter iterativo da engenharia de prompts e a necessidade de equilibrar responsabilidades técnicas com a gestão da equipe, valorizando o resultado obtido ao lidar com dados desestruturados e tecnologias não determinísticas.

13. Conclusão

Ao final do projeto, uma retrospectiva permitiu registrar os principais aprendizados e melhorias para iniciativas futuras.

Desafios enfrentados incluíram a extração de dados de PDFs do CNIS, prejudicada pela falta de padronização, e a dificuldade inicial em obter respostas consistentes da IA. No frontend, o aumento da complexidade expôs a necessidade de melhor gerenciamento de estado.

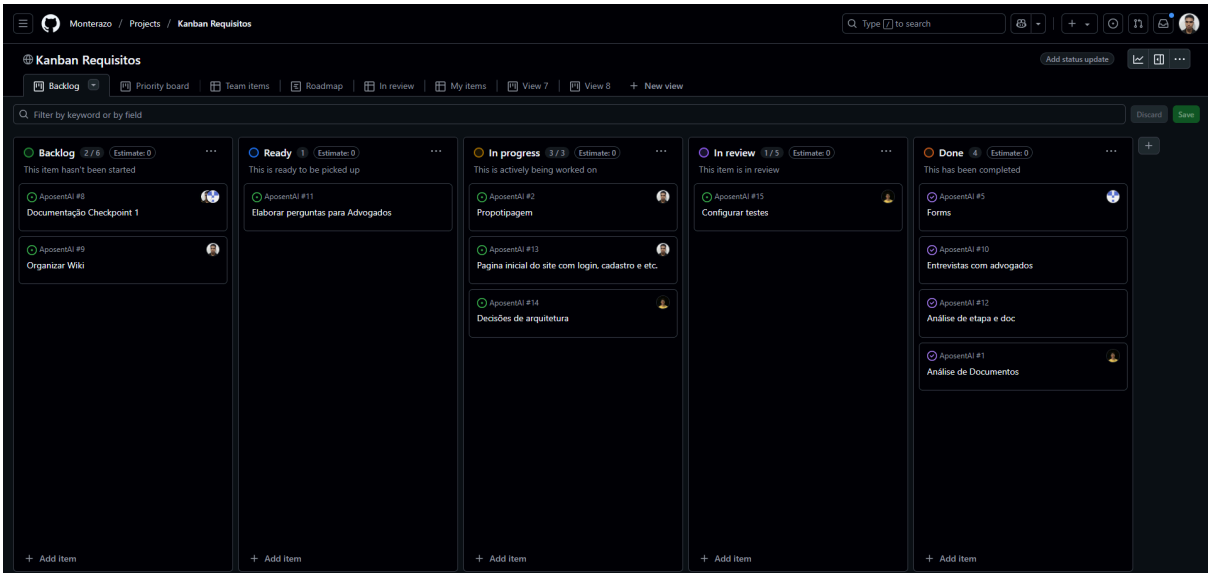
Ajustes implementados envolveram uso combinado de OCR e expressões regulares para leitura de PDFs, camada intermediária para validar respostas da IA e adoção de biblioteca dedicada para centralizar o estado no frontend.

Aspectos que funcionaram bem incluíram a definição prévia da arquitetura e modelo de dados, escolha acertada da pilha tecnológica, adoção de CI/CD desde o início e colaboração fluida entre os membros.

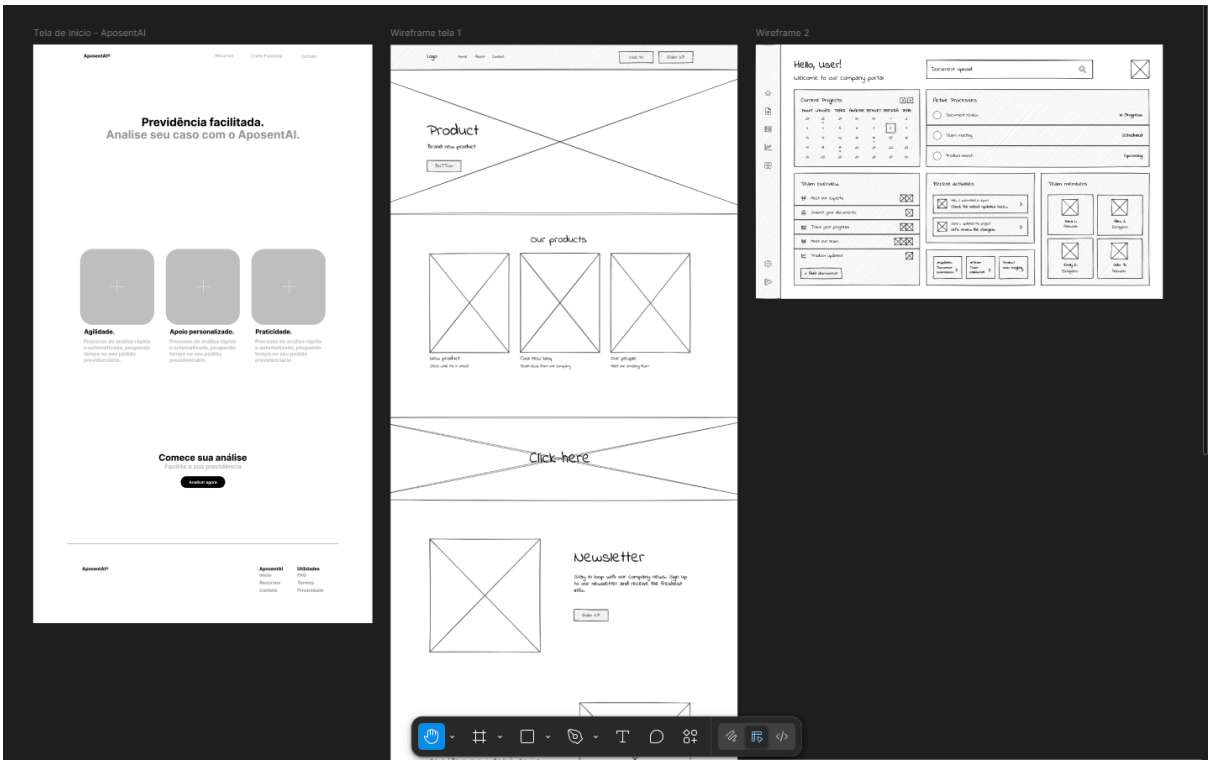
Oportunidades de melhoria apontam para a importância de provas de conceito antecipadas para mitigar riscos técnicos e de ampliar a cobertura de testes End-to-End nas fases iniciais.

Os **depoimentos individuais** reforçaram a relevância de contratos claros de API, interfaces intuitivas e um processo iterativo de engenharia de prompts, destacando também o equilíbrio entre liderança técnica e gestão de equipe.

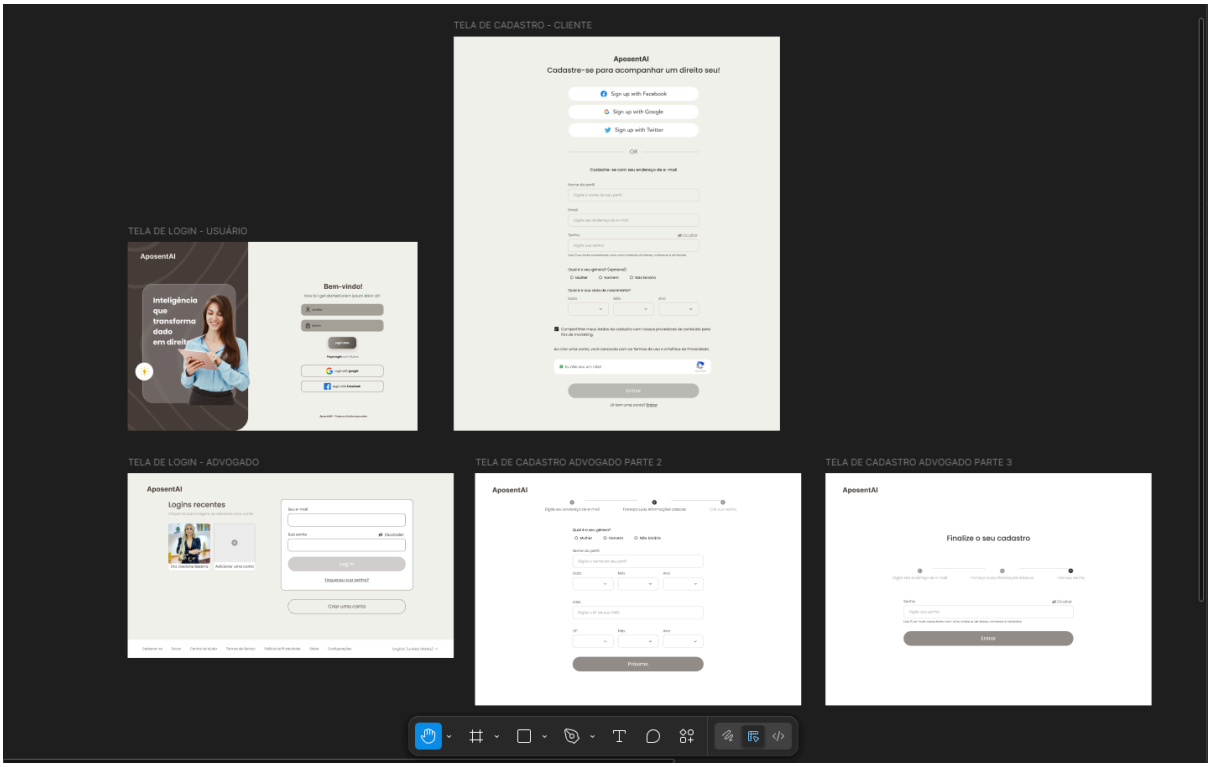
14. Anexos



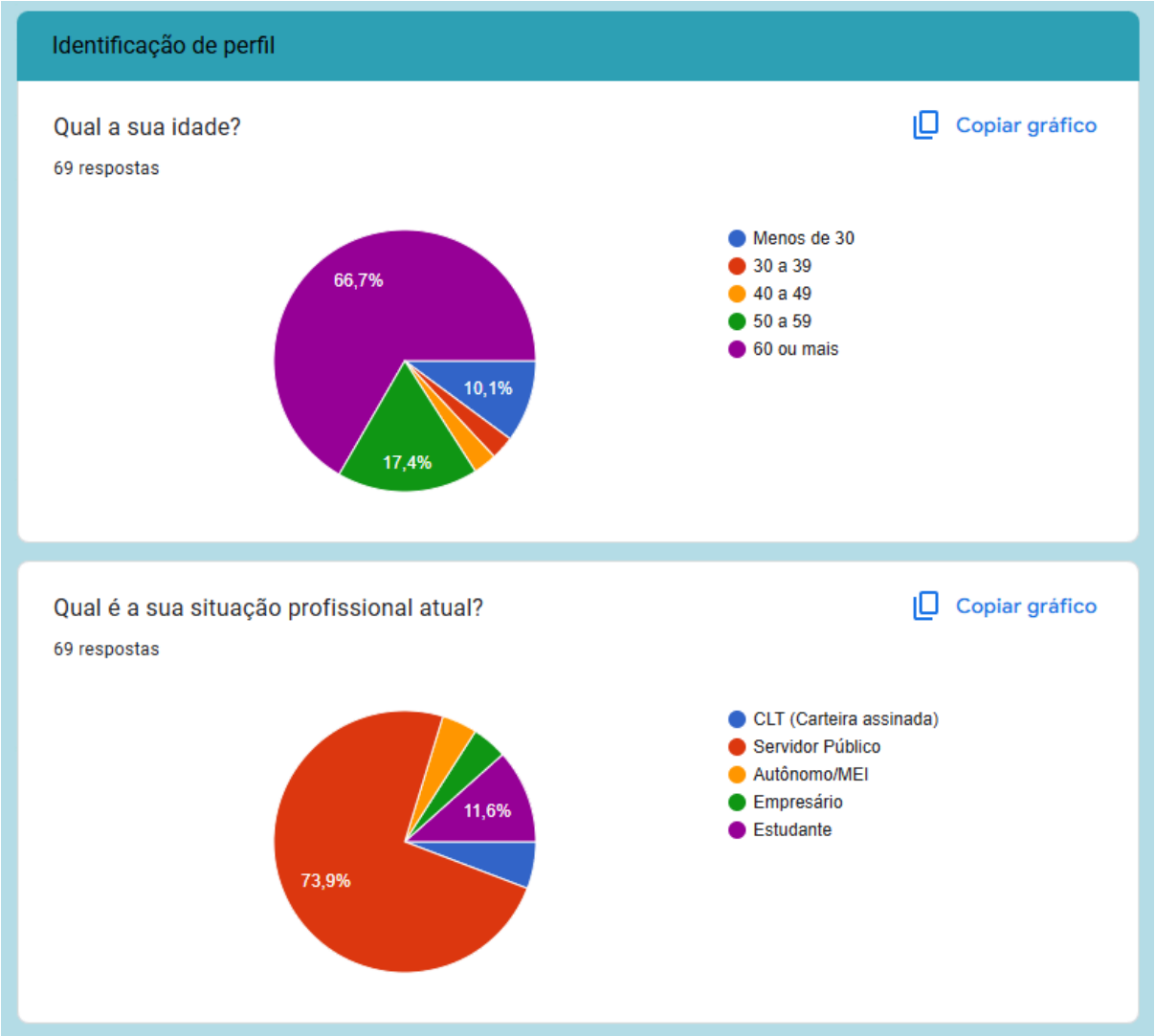
Anexo 1 - Kanban de Requisitos do Projeto



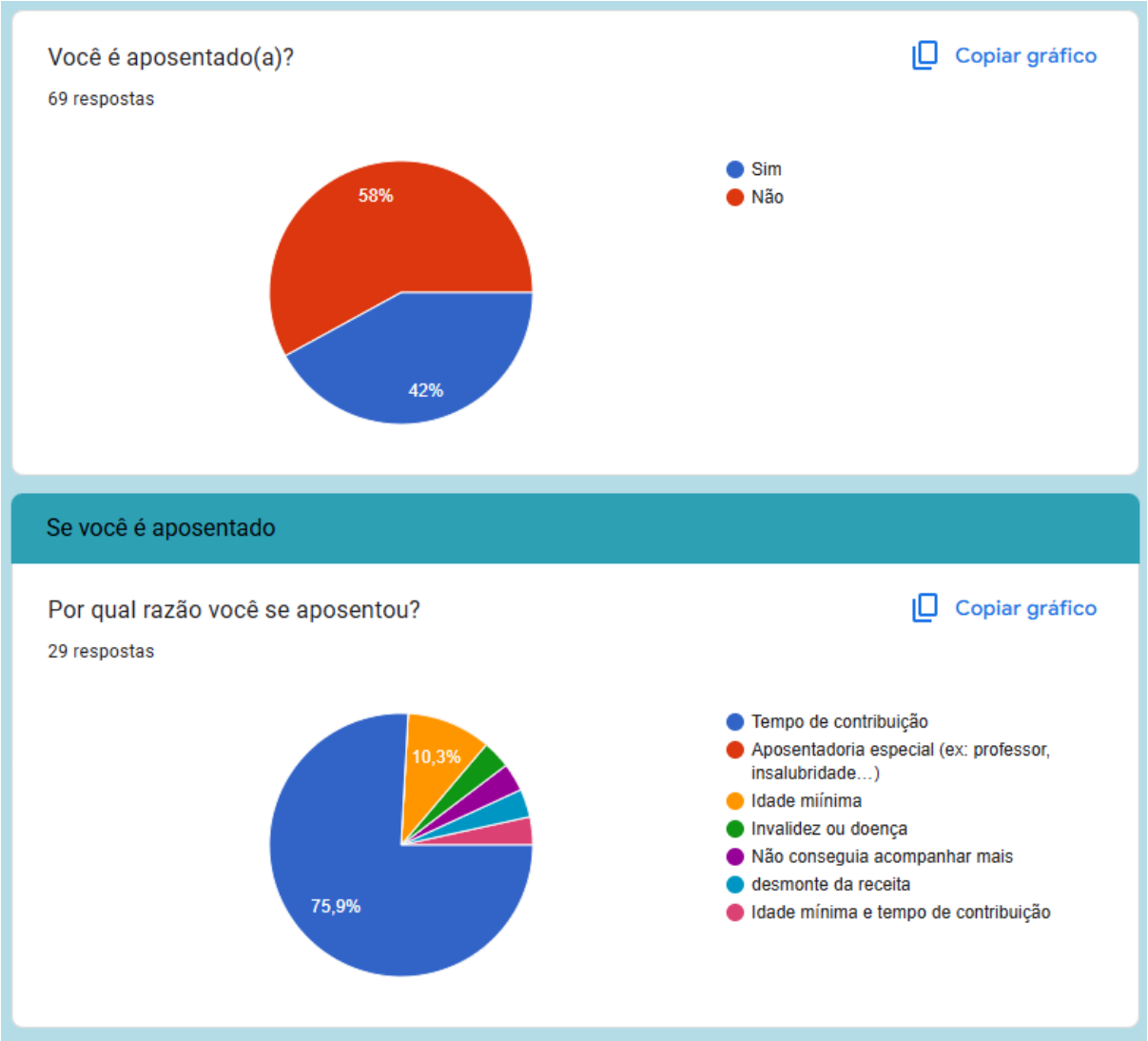
Anexo 2 – Figma em projeto da criação do site



Anexo 3 – Criação de tela de login e registro de usuário no figma



Anexo 4 – Evidências de entrevistas para levantamento de requisitos e criação de personas.

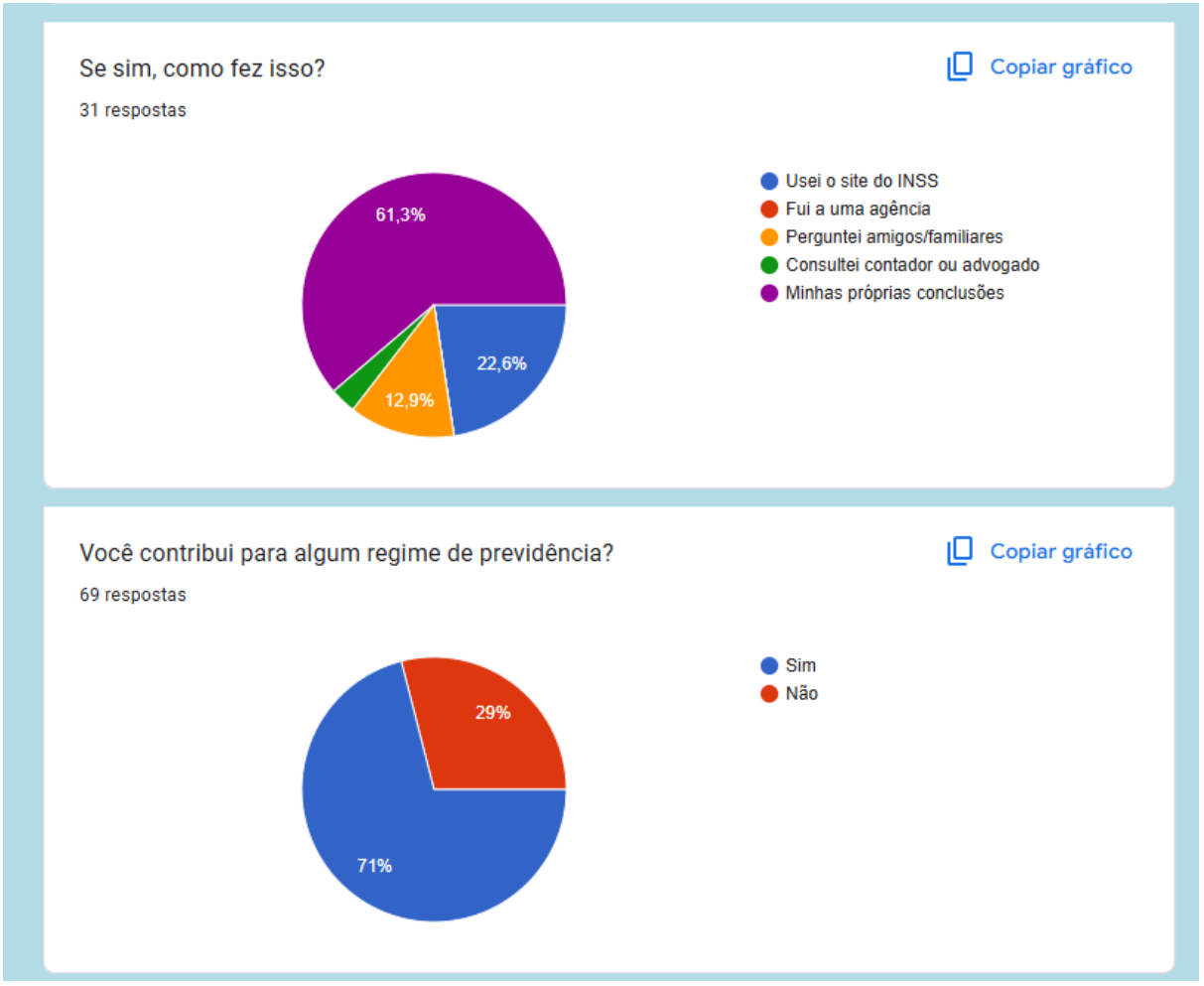


Anexo 5 – Evidências de entrevistas para levantamento de requisitos e criação de personas.

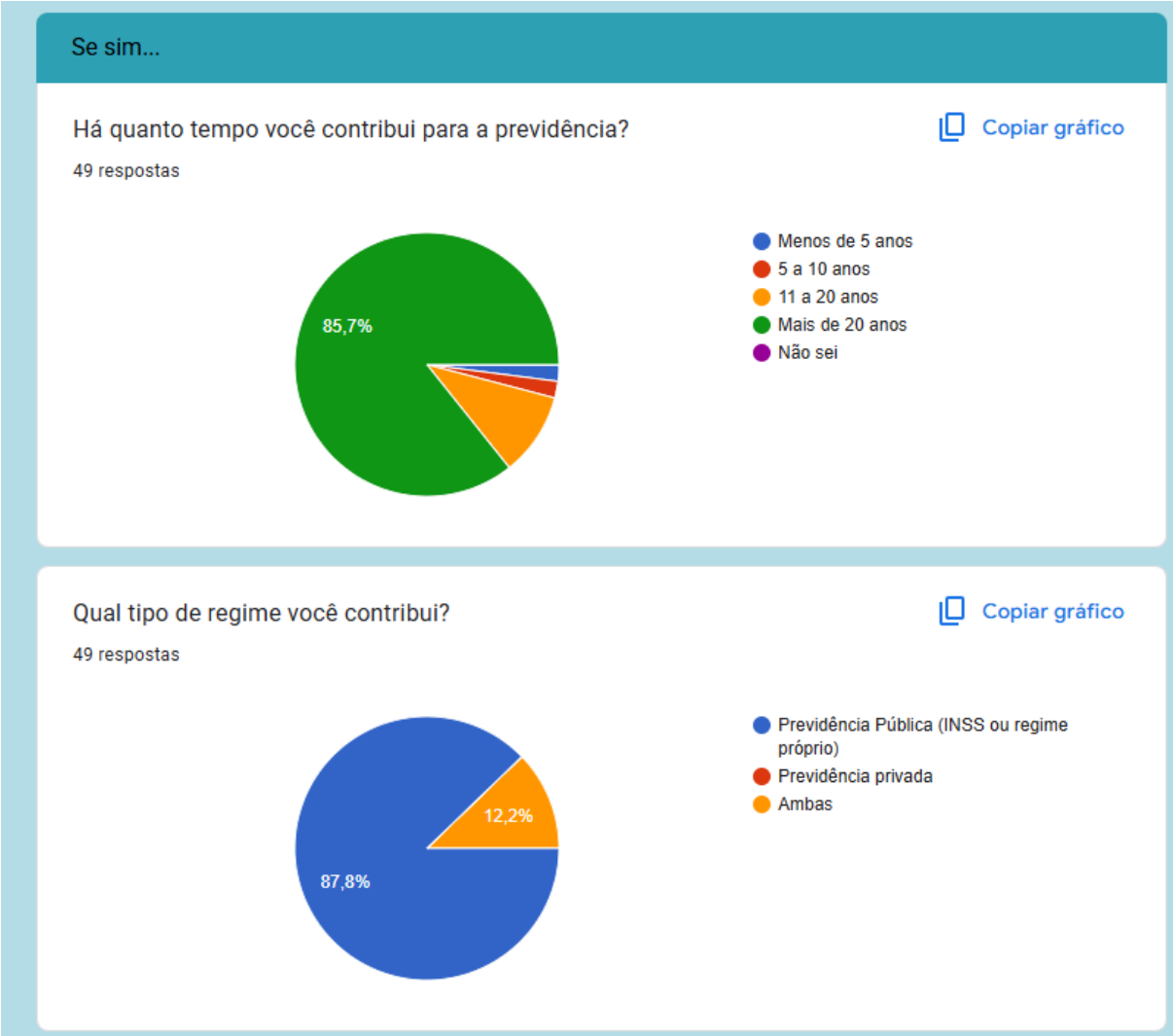
Anexo 6 – Evidências de entrevistas para levantamento de requisitos e criação de personas.

Anexo 7 – Evidências de entrevistas para levantamento de requisitos e criação de personas.

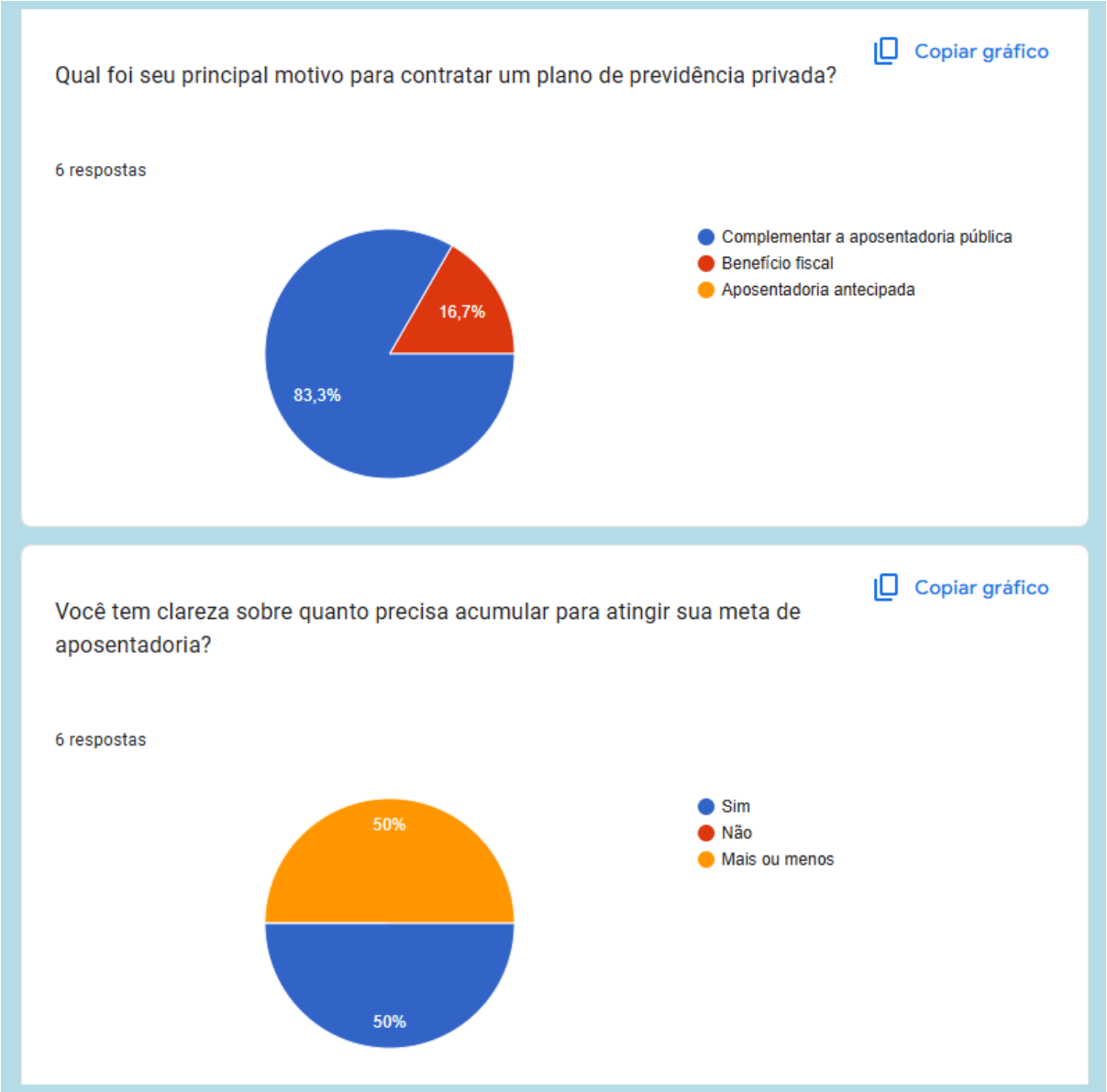
Anexo 8 – Evidências de entrevistas para levantamento de requisitos e criação de personas.



Anexo 9 – Evidências de entrevistas para levantamento de requisitos e criação de personas.

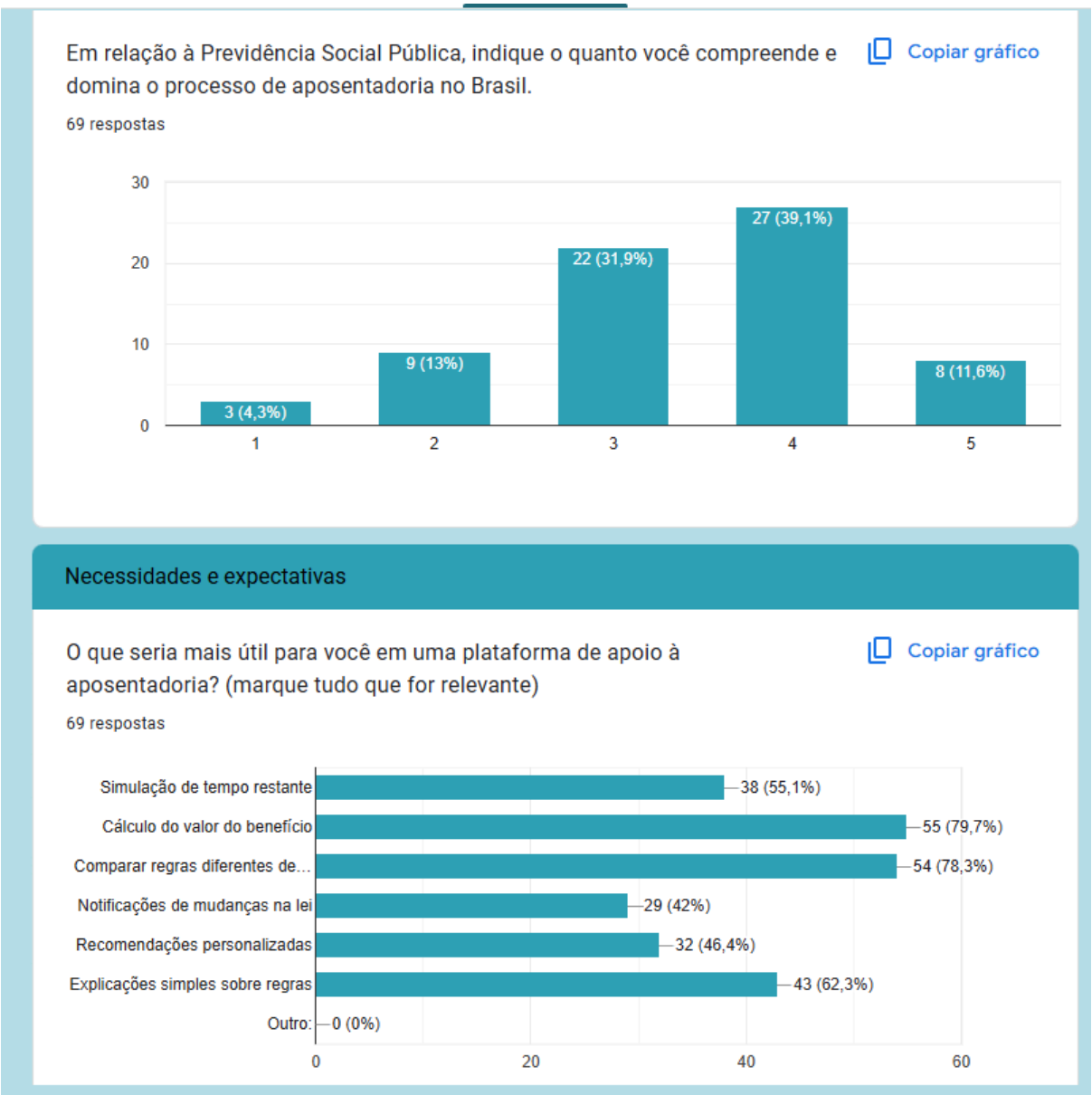


Anexo 10 – Evidências de entrevistas para levantamento de requisitos e criação de personas.

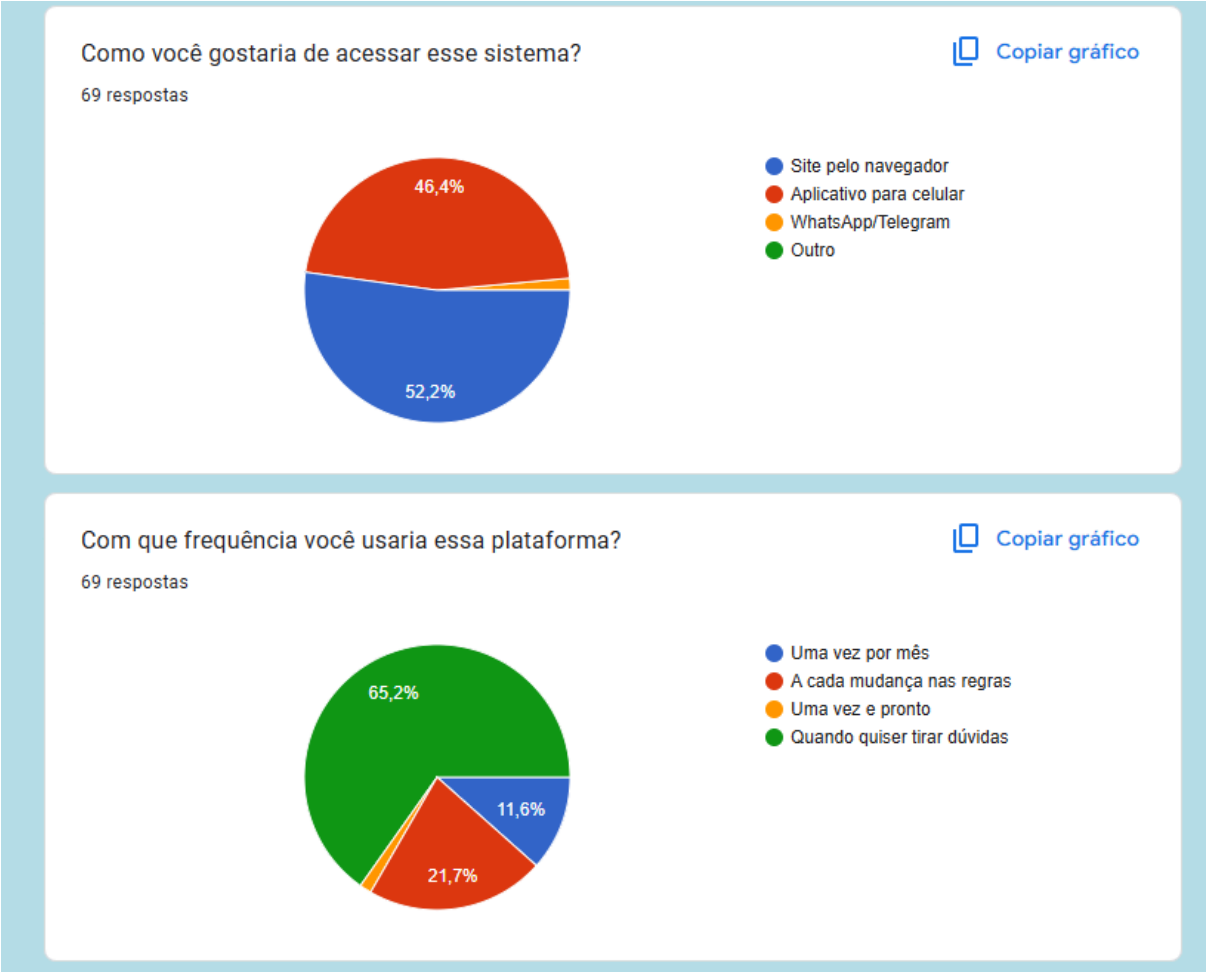


Anexo 11 – Evidências de entrevistas para levantamento de requisitos e criação de personas.

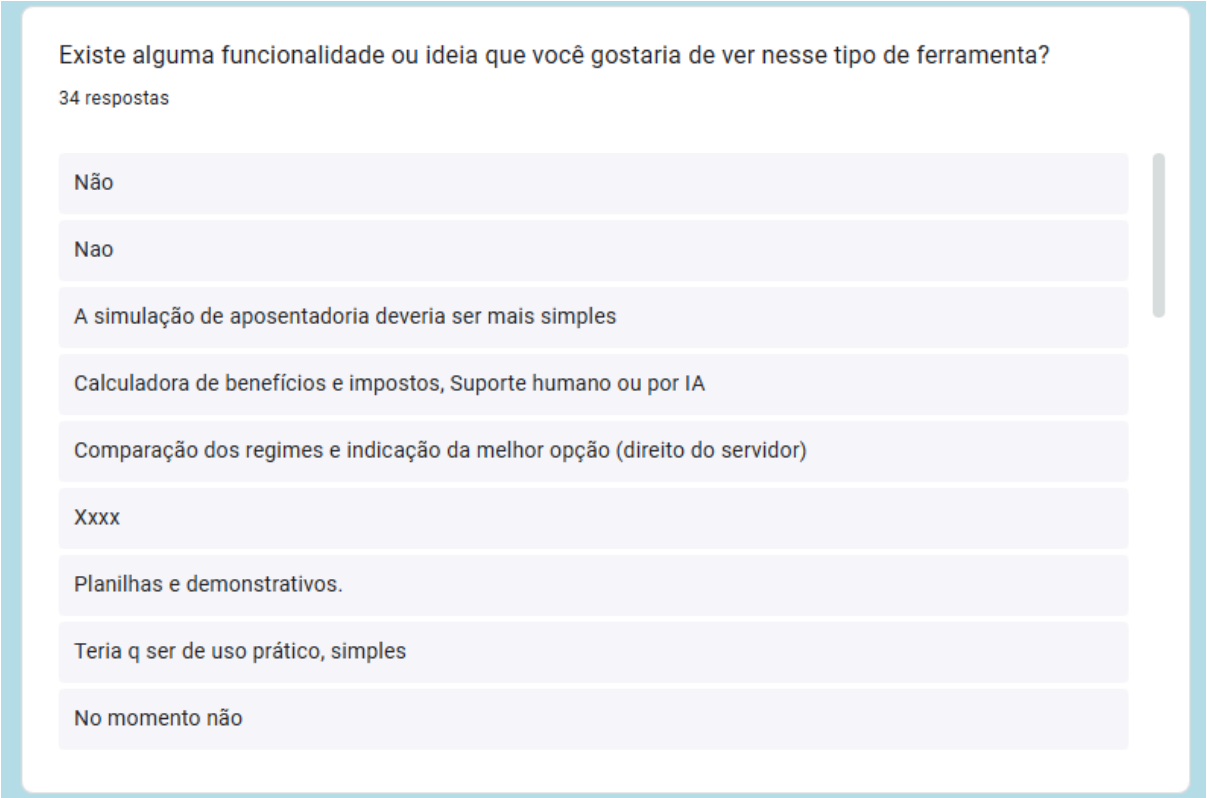
Anexo 12 – Evidências de entrevistas para levantamento de requisitos e criação de personas.



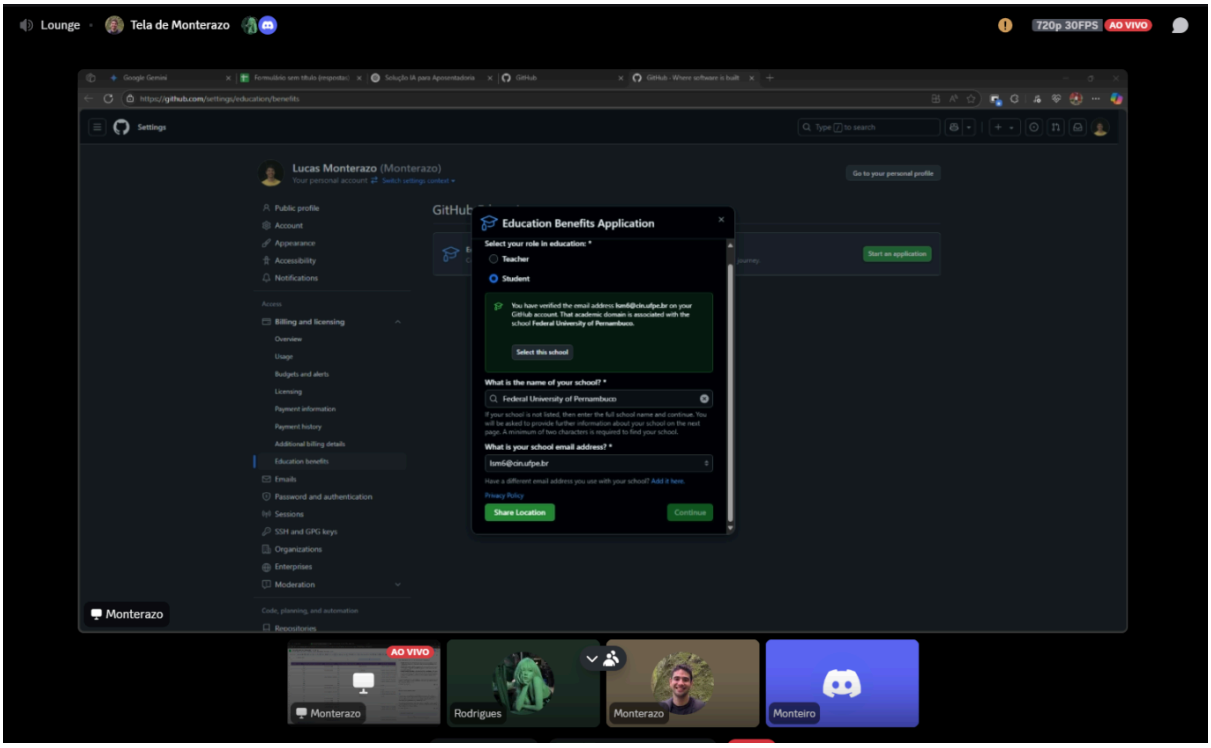
Anexo 13 – Evidências de entrevistas para levantamento de requisitos e criação de personas.



Anexo 14 – Evidências de entrevistas para levantamento de requisitos e criação de personas.



Anexo 15 – Evidências de entrevistas para levantamento de requisitos e criação de personas.



Anexo 16 – Reunião para definição do Projeto.

Link do Github: https://github.com/Monterazo/AposentAI
Link do Deploy do site: https://aposentai.lovable.app/