PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
CURSO DE BACHARELADO EM ENGENHARIA DE SOFTWARE
AGES - AGÊNCIA EXPERIMENTAL DE ENGENHARIA DE SOFTWARE

MATEUS CAMPOS CAÇABUENA

MEMORIAL DE ATUAÇÃO NA AGÊNCIA EXPERIMENTAL DE ENGENHARIA DE SOFTWARE – PERÍODO 2023/1 A 2025/2

AGES I, II, III E IV

Porto Alegre, Rio Grande do Sul
2025

Dedicatória

Aos meus pais, pelo suporte
e por confiarem
em mim.

AGRADECIMENTOS

Primeiramente, agradeço aos meus pais, Denilso Segobia Caçabuena e Maria Soloi Campos Caçabuena, por todo o amor, apoio e incentivo incondicional ao longo da minha trajetória acadêmica. Sem o suporte e a confiança deles, esta caminhada não teria sido possível.

Um agradecimento especial à minha namorada e melhor amiga, Carolina Michel Ferreira, que esteve ao meu lado em todas as etapas desta caminhada. Com seu apoio, compreensão e motivação constantes para seguir em frente mesmo nos períodos mais desafiadores.

Agradeço também às pessoas que conheci durante o curso, que tornaram esta jornada mais leve e inspiradora, em especial aos colegas Felipe Freitas Silva e Luiza Heller Pla, pela parceria, amizade e colaboração em inúmeros momentos importantes.

Por fim, agradeço aos professores que marcaram minha formação e contribuíram diretamente para que eu tivesse certeza sobre o caminho profissional que desejo seguir — Marcelo Yamaguti, Leonardo Heredia e Rafael Chanin — pela dedicação e método de ensino que fizeram disciplinas complexas parecerem simples. Transmitiram não apenas conhecimento técnico, mas também inspiração e propósito mostrando que além de bons professores, são ótimas pessoas.

O sucesso é uma escolha.

Stephen Curry

RESUMO

Este trabalho retrata a minha atuação na Agência Experimental de Engenharia de Software (AGES), disciplina realizada quatro vezes durante o curso de bacharelado em Engenharia de Software da PUCRS. A AGES é o local onde ocorre grande troca de conhecimentos entre os alunos, assim como o aprendizado prático e integrador do curso, objetivando a imersão dos alunos no processo de desenvolvimento de software e o trabalho em equipe, similares ao mercado de trabalho. Em cada módulo, o aluno realiza diferentes papéis como tarefas de programação na primeira etapa, análise de dados da aplicação na segunda, arquitetura do software na terceira e, finalmente, a gestão do time na quarta. Será feita uma reflexão sobre minha evolução desde o início do curso até o momento atual, e o papel da AGES nessa evolução.

PALAVRAS CHAVES: AGES, Engenharia de Software, Aprendizado, Tarefas, Software, Desenvolvimento, Evolução.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

AGES – Agência Experimental de Engenharia de Software

API – *Application Programming Interface*

AWS – Amazon *Web Services*

BD – Banco de Dados

CEO – *Chief Executive Officer*

CRUD – *Create, Read, Update, Delete*

CSS – *Cascading Style Sheets*

CSV – *Comma-Separated Values*

DDD – *Domain-Driven Design*

ES – Engenharia de Software

E2E – *End-to-end*

HTML – *HyperText Markup Language*

JSON – JavaScript *Object Notation*

PDF – *Portable Document Format*

PUCRS – Pontifícia Universidade Católica do Rio Grande do Sul

SPA – *Single Page Application*

US – *User Stories*

SUMÁRIO

1 – APRESENTAÇÃO DA TRAJETÓRIA DO ALUNO

Minha trajetória na área da computação teve início na universidade, por meio das disciplinas de Fundamentos de Programação e Programação Orientada a Objetos, cursadas como requisitos prévios no curso de Engenharia de Software (ES). Esses componentes curriculares me proporcionaram a base lógica necessária para ingressar na Agência Experimental de Engenharia de Software (AGES), onde pude aplicar pela primeira vez os conhecimentos teóricos em um ambiente que simula o mercado de trabalho.

Paralelamente, no primeiro semestre de 2023, busquei aprofundar meus conhecimentos de forma autônoma por meio de plataformas de ensino online. Concluí formações em HTML (21 horas) e React (34 horas), o que ampliou minha compreensão sobre diferentes linguagens e *frameworks* de desenvolvimento, complementando o aprendizado em Java obtido na universidade.

Ainda em 2023, fui contratado pela empresa SoftKuka para atuar como desenvolvedor *frontend*, o que marcou minha primeira experiência profissional na área. No ambiente corporativo, adquiri conhecimentos práticos em aplicações *web*, aprendi sobre metodologias ágeis e participei de projetos com clientes externos e prazos reais, o que consolidou minha paixão pela Engenharia de Software.

A AGES sempre me despertou grande interesse por sua proposta de aproximar o ambiente acadêmico da realidade do mercado. Minha primeira participação ocorreu no projeto “Veículos via Montadora”, durante a AGES I, onde pude desenvolver habilidades técnicas em Python e React, além de aprimorar minhas *soft skills* — especialmente comunicação, proatividade e trabalho em equipe. Essa experiência foi determinante para o desenvolvimento das competências que me permitiram ingressar no mercado de trabalho.

Na sequência, durante a AGES II, conquistei uma vaga de estágio na Poatek, empresa multinacional de tecnologia. Essa oportunidade me proporcionou contato com equipes multidisciplinares e projetos de grande porte, aprofundando meus conhecimentos em *backend*, arquitetura de software e boas práticas de desenvolvimento. A vivência simultânea entre a Poatek e a AGES permitiu uma troca constante de aprendizados, em que o que eu aprendia em uma era aplicado na outra.

Já na AGES III, assumi responsabilidades maiores, tornando-me uma referência técnica dentro da equipe ao mesmo tempo em que fui efetivado no trabalho. Essa etapa me ensinou que o aprendizado é um processo contínuo e colaborativo: muitas vezes, profissionais menos experientes podem contribuir com perspectivas inovadoras, algo que apliquei tanto no trabalho quanto na universidade.

Na AGES IV, em um momento de amadurecimento profissional e pessoal. A experiência de atuar como líder de projeto me ensinou sobre gestão de pessoas, empatia e comunicação assertiva, habilidades que tenho levado também para minha atuação na Poatek.

Em retrospectiva, percebo que cada etapa da minha formação — acadêmica, profissional e na AGES — contribuiu para moldar não apenas minhas competências técnicas, mas também meu comportamento como engenheiro de *software*. Hoje, sinto-me preparado para encarar novos desafios com responsabilidade, visão sistêmica e compromisso com o aprendizado contínuo.

2 – PROJETO AGES I - “Veículos via Montadora”

Esta seção busca apresentar minha passagem como AGES I pelo projeto Veículos via Montadora. Aqui estão descritos os artefatos entregues, a atuação ao longo das sprints e os pontos de melhoria identificados no decorrer do projeto.

2.1 Introdução

O projeto Veículos via Montadora tem como objetivo desenvolver um sistema Web que pudesse auxiliar no cadastramento e atualização de informações dos produtos (carros) de uma maneira rápida e efetiva. Todos os dados vinham via PDF (*Portable Document Format)*, logo, o desafio deste projeto foi a organização e a estruturação de um banco de dados, tanto de captação iniciando com a leitura do PDF, como de entrega, exportando os dados para o sistema do cliente.

Inicialmente, o *stakeholder* registrado na documentação do projeto chamava-se Genaro Passos. Porém, na primeira reunião com o cliente, foi apresentado o Leonardo Cunha. Este foi o novo cliente que nos explicou o desligamento de Genaro da empresa. Desta maneira, o nosso *stakeholder* oficial era o Leonardo, que nos introduziu a Sinosserra: empresa cliente que trabalhava com consórcios de veículos. Nosso objetivo foi facilitar o processo de extração de PDF que são enviados, tanto da Sinoscar, quanto do Tramonto, para o nosso cliente.

Posteriormente, foram introduzidos outros 2 *stakeholders* do projeto: Fabiano Longaray e Luana Lima. Ambos foram apresentados na segunda reunião que tivemos, o Fabiano para entender os termos técnicos e a Luana, CEO (*Chief Executive Officer*) da Sinosserra e a usuária que iria utilizar o nosso programa na empresa.

A execução do projeto ocorreu no primeiro semestre de 2023, entre as datas 8 de março e 14 de junho, pelos estudantes de Engenharia de Software. Neste projeto, havia 8 AGES I, 4 AGES II, 3 AGES III e 3 AGES IV, totalizando 18 membros da equipe orientados pelo Prof. Daniel Antonio Callegari. A foto do time responsável pelo projeto pode ser vista na Figura 1:

Figura 1: Time do projeto Veículos via Montadora

Fonte: *Wiki* do projeto

2.2 Desenvolvimento do projeto

Esta seção apresenta informações referentes ao desenvolvimento do projeto: localização do código-fonte, banco de dados, protótipos de tela desenvolvidos, arquitetura e tecnologias utilizadas.

2.2.1 Repositório do código-fonte do projeto

O código-fonte do projeto foi organizado de maneira tradicional, separando o programa que interage diretamente com os usuários da aplicação que lida com a lógica de negócios, processamento de dados e outras funcionalidades que não são visíveis para os usuários finais.

O código-fonte de ambos os programas se encontra distribuído em dois repositórios, nomeados *frontend* e *backend*:

- *Backend*: https://tools.ages.pucrs.br/veiculos-via-montadora/backend
- *Frontend:* https://tools.ages.pucrs.br/veiculos-via-montadora/frontend

2.2.2 Banco de Dados utilizado

O banco de dados do projeto foi modelado utilizando uma abordagem não-relacional. Diferentemente dos bancos de dados relacionais tradicionais, que armazenam dados em tabelas e utilizam esquemas predefinidos, o modelo não-relacional do MongoDB permite uma estruturação mais flexível dos dados, baseada em documentos no formato JSON (*JavaScript Object Notation*). A figura 2 a seguir retrata o diagrama do BD utilizado:

Figura 2: Diagrama do Banco de Dados

Fonte: *Wiki* do projeto

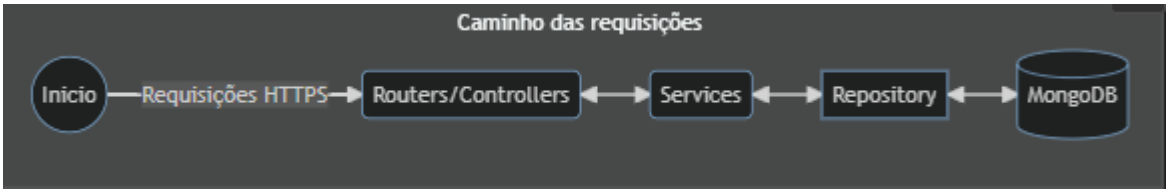
Para mais detalhes sobre a estrutura e funcionamento do banco de dados, consulte a documentação disponível no seguinte link: https://tools.ages.pucrs.br/veiculos-via-montadora/wiki/-/wikis/banco_dados.

2.2.3 Arquitetura utilizada

Como dito anteriormente, separamos o projeto em *frontend* e *backend*. Para a estrutura de repositórios do *backend*, organizamos da seguinte maneira:

- ***Routers***: Responsáveis por definir as rotas HTTP do aplicativo FastAPI.
- ***Controllers***: Lidam com a lógica de negócios para cada rota, processando as solicitações e retornando as respostas adequadas.
- ***Services***: Realizam a lógica de negócios principal do aplicativo, que pode envolver validações, chamadas a APIs (Application Programming Interface) externas, processamento de dados etc.
- ***Repository***: Faz a interação com o banco de dados MongoDB, executando operações de leitura/gravação.
- **MongoDB**: Banco de dados NoSQL utilizado para armazenar e recuperar os dados do aplicativo.

Figura 3: Diagrama de Sistema



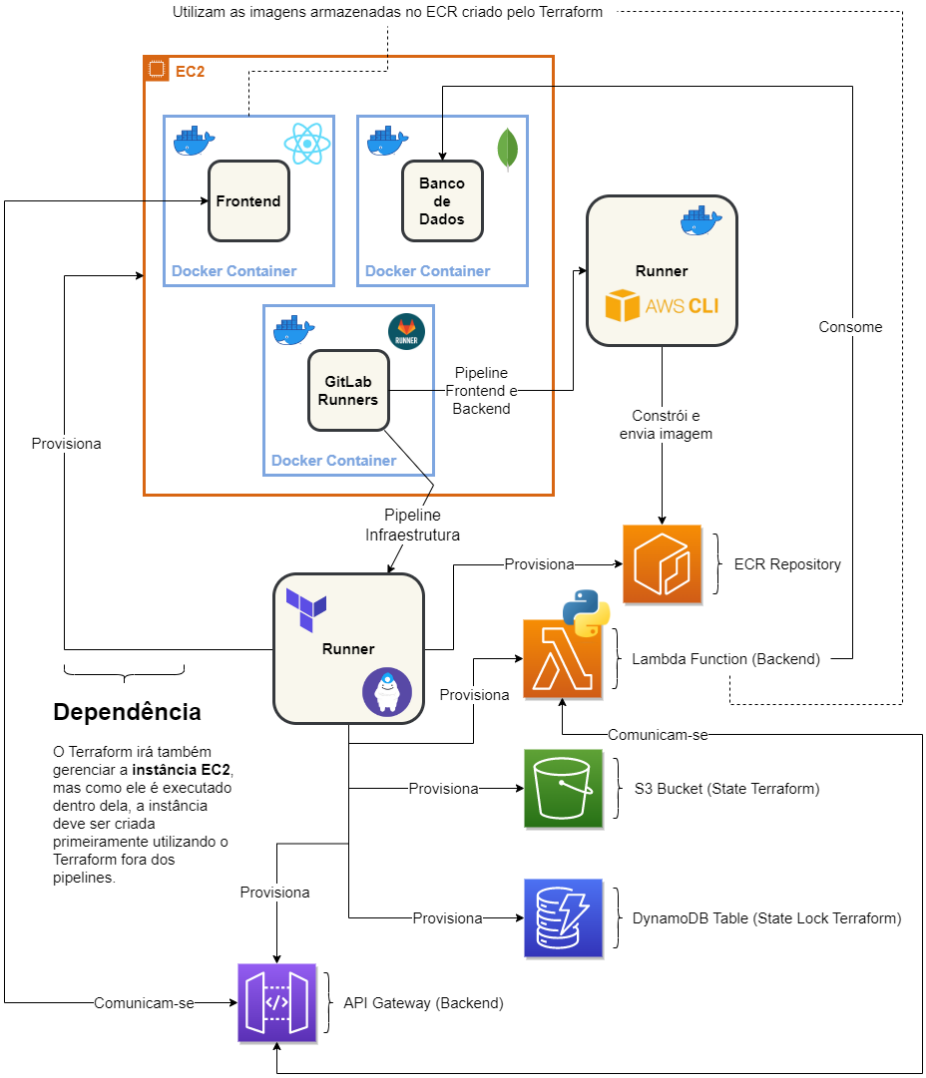
Fonte: *Wiki* do projeto

Para manter a infraestrutura do projeto por 1 semestre, esse será o custo financeiro estimado a cada mês e ao final do semestre:

- Custo Mensal: ~ 12,56 USD
- Custo Semestral: ~ (12,56 USD * 6) = 75,36 USD

A figura 4 a seguir retrata a arquitetura em alto nível e o processo de *deploy* da infraestrutura do projeto Veículos Via Montadora:

Figura 4: Diagrama de *Deploy*



Fonte: *Wiki* do projeto

Mais detalhes sobre a estrutura de arquivos, orçamento AWS e permissões do Terraform podem ser encontrados na wiki do projeto, localizado no link a seguir: https://tools.ages.pucrs.br/veiculos-via-montadora/wiki/-/wikis/arquitetura.

2.2.4 Protótipos das telas desenvolvidas

Para realizar o protótipo das telas desenvolvidas, utilizamos o Figma (FIELD,2012), por conta de sua capacidade de colaboração em tempo real. Nesta ferramenta, várias pessoas podem trabalhar simultaneamente em um mesmo projeto, visualizando as edições em tempo real. É extremamente útil e encaixa-se

perfeitamente com o projeto, pois muitas pessoas desenvolveram *mockups* simultaneamente.

O Stakeholder não havia nenhum padrão de cores pensado previamente, portanto, desenvolvemos do zero e utilizamos a criatividade para encaixar um design que agradasse os clientes.

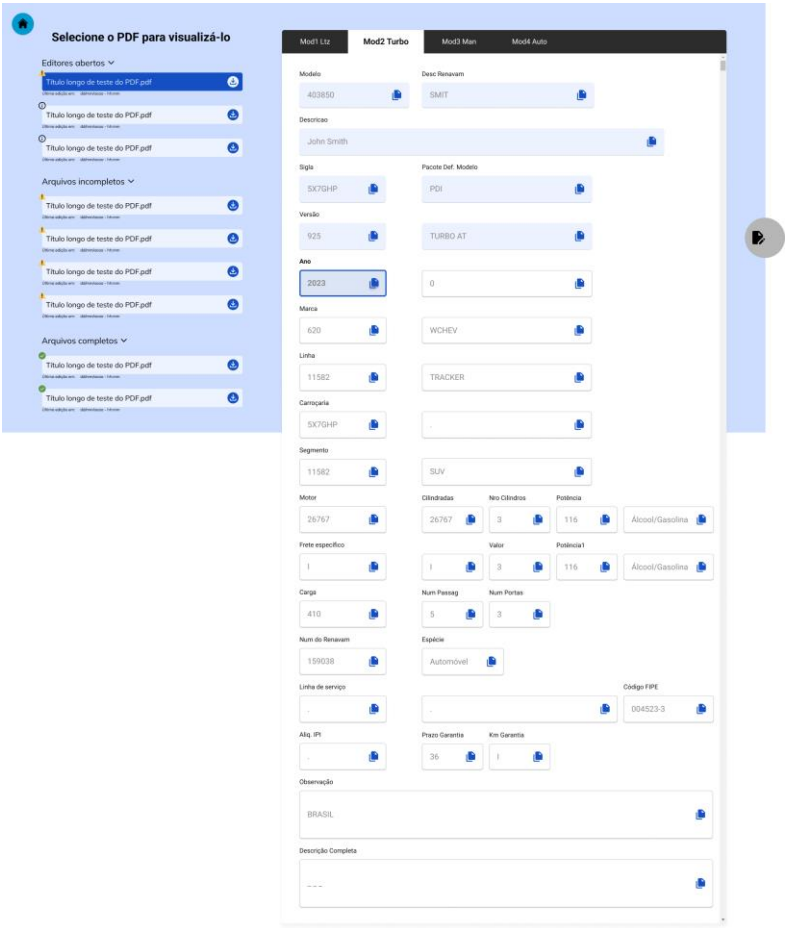
Buscamos seguir o mesmo padrão de aparência em todas as telas realizadas, a figura 5 a seguir apresenta um dos principais *mockups* que desenvolvi, junto com meus colegas de equipe:

Figura 5: Tela de Extração do PDF

Fonte: *Wiki* do projeto

Apesar de extensa, a segunda tela foi simples. Foram múltiplos campos para serem preenchidos com informações de determinado veículo e, para facilitar a experiência do usuário, foi colocado um botão de copiar automaticamente ao final de cada campo, assim, invés de usar o teclado para copiar, apenas um *click* já adiantava este processo que seria repetido várias vezes. A tela descrita está na figura 6 a seguir:

Figura 6: Tela de Visualização de Dados Extraídos



Fonte: *Wiki* do projeto

Outros *mockups* que envolvem a transição de tela e componentes desenvolvidos podem ser encontrados na página de *mockups* da *wiki*, localizado no link a seguir: https://tools.ages.pucrs.br/veiculos-via-montadora/wiki/-/wikis/mockups.

2.2.5 Tecnologias Utilizadas

Para o lado do *frontend*, foi utilizado o React (WALKE, 2011), biblioteca JavaScript (EICH, 1995) de código aberto com foco em criar interfaces de usuário em páginas web que precisam ser atualizadas em tempo real. Ele é ideal para esse projeto, pois permite que atualize a interface de usuário de forma rápida e eficiente.

O desenvolvimento do código em React foi realizado com TypeScript (HEJLSBERG, 2012): linguagem de programação de código aberto desenvolvida pela Microsoft. É um superconjunto sintático estrito de JavaScript e adiciona tipagem

estática opcional à linguagem, além disso, sua estilização foi implementada com o Material UI (TASSINARI, 2014): linguagem de design desenvolvida pela Google.

Não obstante, preferimos usar o Swagger (TAM, 2010) para utilizar as APIs necessárias no *frontend*: linguagem de descrição de interface para descrever APIs RESTful expressas usando JSON. É usado junto com um conjunto de ferramentas de software de código aberto para projetar, construir, documentar e usar serviços da Web RESTful.

Do lado do *backend*, optamos pelo Python (VAN ROSSUM, 1980) a: linguagem de programação de alto nível com sintaxe mais simplificada e próxima da linguagem humana, utilizada nas mais diversas aplicações, como desktop, web, servidores e ciência de dados. Para seu desenvolvimento, utilizamos o framework FastAPI (RAMIREZ, 2018), assim como Poetry (EUSTACE, 2018) e Pytest (KREKEL, 2004) para gerenciamento de dependências e testes automatizados, respectivamente.

Para a montagem do banco de dados do projeto, foi realizado com o MongoDB (MERRIMAN, 2007): software de banco de dados NoSQL orientado a documentos livre, de código aberto e multiplataforma, escrito na linguagem C++.

No quesito infraestrutural do projeto, usamos tanto o Docker (HYKES, 2013), quanto Amazon Web Services (AMAZON, 2002). Docker é um conjunto de produtos de plataforma como serviço que usam virtualização de nível de sistema operacional para entregar software em pacotes chamados contêineres. Já a Amazon Web Services, também conhecido como AWS, é uma plataforma de serviços de computação em nuvem, que formam uma plataforma de computação na nuvem oferecida pela Amazon.

2.3 Atividades desempenhadas pelo aluno no projeto

Nesta seção, estão todas as atividades individuais feitas por mim durante o projeto, assim como as ferramentas utilizadas e contribuições. Estão separadas por *sprints*, que são períodos de mais ou menos 2 semanas que, em cada uma, são determinados objetivos e entregas para o final dela.

2.3.1 Sprint 0

Após a primeira reunião com o stakeholder, iniciou-se a *sprint* 0, que se consistia em compreender o projeto, escolher as melhores linguagens e frameworks, além de estudarmos estas tecnologias para conseguirmos desenvolver com êxito.

Compreendi que o principal desafio do projeto seria transformar o PDF que era lido por humanos para uma máquina. Não obstante, precisaria adaptar para arquivos que eram padronizados de duas maneiras diferentes, já que a Sinoscar e a Tramonto são empresas diferentes, possuem distintas maneiras de padronização de PDF.

A partir disto, juntamente com a equipe, realizei uma pesquisa que totalizou a procura de 8 ferramentas diferentes, a fim de encontrar uma que seja a ideal para uma leitura de PDF. Encontramos o Tabula, ferramenta para liberar tabelas de dados bloqueados em arquivos PDF. Assim, era extraído estes dados em uma planilha CSV ou Microsoft Excel.

Posteriormente, participei da primeira reunião com a equipe para discutirmos quais linguagens e frameworks usaríamos. Debatemos sobre as possibilidades entre Python e Java para o *backend* e, inicialmente, foi decidido que seria feito em Java, além do framework Spring Boot.

Em relação a tecnologias no *frontend*, havia a discussão entre React, Next e Vue, mas prontamente foi decidido que seria React pela votação da equipe, não foi gerada muita discussão.

Apesar de tudo ser novidade, entrei no projeto com a mentalidade de sair da zona de conforto e, por ter aprendido e desenvolvido Java nos últimos semestres da faculdade, não queria desenvolver novamente na linguagem.

Portanto, respaldei minha vontade para a equipe que gostaria de desenvolver o *frontend* deste projeto e me comprometi tanto a estudar e aprender, quanto a contribuir positivamente no projeto. Felizmente, eu já havia realizado o curso de HTML e, por conta disso, já obtinha a mínima noção para aprender React. Posteriormente, enquanto meus colegas que já possuíam conhecimento montavam o Figma, intensifiquei meus estudos em CSS, além do React e Typescript para começar a *sprint* 1 podendo ter condições de programar.

Na última reunião desta sprint, houve uma discussão pela possível demora da execução do programa *backend* com o Java. Então, foi abordado a possibilidade de começarmos a usar 2 linguagens (Java e Python). Entretanto, percebemos que facilitaria muito se houvesse apenas Python, pois, além de Java não ser tão performático com *lambda* em uma comparação, Python é a melhor linguagem para

extração de PDF graças ao Pandas, sem contar na complexidade a mais que seria se houvesse que aprender a fazer em 2 linguagens diferentes.
Percebo que um grande problema causado nas sprints posteriores foi causada por um erro nesta sprint: indecisão em relação ao *backend*. O escopo do projeto estava confuso, por mais que eu soubesse que seria usado o Python e deveria estudar o Pandas, não sabia se ele realmente seria usado.

Acredito que pequei na minha habilidade de comunicação, por mais que eu fosse do *frontend* nesse começo, gostaria de entender o que aconteceria no *backend* para futuramente participar dele também.

2.3.2 Sprint 1

Com o início da segunda *sprint* do projeto, participei da apresentação da *User Stories*, juntamente com os *mockups* produzidos no Figma na reunião com o *stakeholder* Fabiano. Ele gostou muito da apresentação, surpreendeu-se positivamente com o resultado e principalmente com a estilização que foi proposta. Por termos recebido um *feedback* tão positivo, tanto eu, quanto a equipe, achamos que estávamos no caminho certo.

Finalmente, foi decidido que usaríamos o Tabula-py para extrair os PDFs, de acordo com os AGES III, ele seria mais “inteligente” para o tipo de PDF da Sinosserra.

Não obstante, foram definidas as *squads* da equipe e, em consenso da equipe, a função determinada para minha *squad* era começar o desenvolvimento do *frontend.* Graças aos estudos feitos durante a primeira sprint, participei ativamente e ajudei a implementar a tela para baixar o arquivo PDF, além da opção de escolha de grupo de veículos e análise deles.

Esta *sprint*, devido ao mau planejamento dos gerentes do projeto da equipe (AGES IV), não houve muitas sessões de desenvolvimento do código e, por conta disso, minha *squad* foi a única que se reuniu para programar. As consequências disso foram grandes: na apresentação do projeto, não conseguimos alcançar os objetivos propostos no começo da Sprint e isso preocupou os stakeholders, ligando um sinal de alerta para toda a equipe.

Por mais que esta sprint tenha sido um desastre, eu fiquei muito feliz em ter participado, pela primeira vez, do desenvolvimento de um projeto com responsabilidades e colegas de equipe. Trabalhar utilizando os métodos que estudei

nos últimos 2 semestres me inspirou a dar mais valor aos ensinamentos das cadeiras do curso. Ressaltei este ponto no discurso final do projeto com a equipe e repito: foi um momento muito especial para mim e creio que nessa sprint eu tive a certeza de que gosto de ser um desenvolvedor de software. A figura a seguir retrata a descrição e status das *User Stories* na *sprint* 1:

Figura 7: Descrição e Status da *User Stories* da Sprint 1

Sprint 1

- Começo: 29/03/2023
- Fim: 12/04/2023
- Apresentação para os Stakeholders: **10/04/2023**

US	Descrição	Status
US01	Tela de Envio de PDFs	Não Aceito
US03	Tela de Extração de Dados	Não Aceito
US04	Tela de Conclusão da Extração	Não Aceito

Fonte: *Wiki* do projeto

2.3.3 Sprint 2

Por conta do fracasso das entregas estipuladas na *sprint* 1 para o *stakeholder*, foi reorganizado a *User Stories* e as *tasks* para cada equipe. Continuei no mesmo *squad*, porém, nesta etapa foi determinado que os AGES I são os responsáveis de desenvolver as *tasks*. Portanto, fiquei responsável de entregar duas tarefas:

- Montadora do PDF – Identificar qual montadora o PDF pertence, tanto pelo nome do arquivo, quanto pela possibilidade de o usuário selecionar.
- Preparação para extração - Com o recebimento do PDF pelo *frontend*, selecionar que tipo de Arquivo (MEV, Jeep, Outro) para selecionar o tópico de algoritmos de cada tipo.

Graças as intensivas tentativas do Arthur Ibarra (AGES III), descobrimos que não há padronização em nenhum dos 20 PDFs que nos foi disponibilizado da Chevrolet (MEV), ou seja, não tem como fazer um sistema para isso. Em uma decisão coletiva, optamos por descartar o MEV do escopo do projeto.

Durante o andamento da *sprint* me envolvi em uma discussão com o meu *squad*: tanto o Luiz (AGES IV), quanto o Kevin (AGES III), se reuniram com o Lucas Susin (AGES III de outro *squad*) sem a presença de nenhum AGES I para desenvolver a *task* da Montadora do PDF. Eu questionei o porquê da realização da tarefa em que eu era o responsável, sem a minha presença ou a de qualquer outro AGES I para aprender, sendo nós os responsáveis pelo desenvolvimento. Ambos se defenderam alegando que era aceitável o meu ponto, porém, era necessário codificar quando houvesse disponibilidade. Foi proposto por mim, na reunião seguinte, termos sessões de codificação todas as quintas e em um dia do fim de semana.

Observando este atrito hoje em dia, fico orgulhoso do meu posicionamento. Pensei muito se deveria discutir com os AGES veteranos, estava inseguro em abrir o debate de terem feito algo errado. Contudo, o respaldo da Carolina e do Gabriel (outros AGES I do squad) me fizeram crer que o meu ponto de argumentação estava correto. De fato, estava. Meus colegas concordaram comigo e a proposta gerou uma nítida melhora no rendimento da equipe depois do combinado das reuniões.

2.3.4. Sprint 3

Durante a terceira *sprint* o time resolveu se dividir em equipes ainda menores, a fim de acelerarmos o processo de desenvolvimento das *tasks* devido ao problema encontrado no planejamento: os AGES IV separaram as tarefas de cada *squad* 5 dias depois do início da *sprint,* nos gerando uma grande quantia de afazeres em um curto período.

Finalmente, as *tasks* em que fiquei responsável eram:

- Criar o componente *input* de dados
- Integrar a estrutura base com os inputs
- Criar o botão de salvar

Eu senti uma grande insegurança de programar nesta *sprint*, vendo o programa encorpado, não me senti confiante de desenvolver. Pensei que era incapaz e que não possuía o conhecimento suficiente, podendo implicar erros no código. Por mais que a *sprint* tenha sido um sucesso na entrega, acredito que tenha sido a pior em questão de desempenho individual.

O ápice desta *sprint* ocorreu em apenas 1 dia: dos 18 integrantes da equipe, 10 participaram de uma ligação que aconteceu durante a madrugada do dia de

apresentação ao *stakeholder*. Nesta reunião, desenvolvemos todas as *tasks* que faltavam e, por mais que tenha dado certo, foi uma das piores experiências que passei na AGES.

A “cultura do herói” que o professor Callegari enfatizou no início do projeto para não fazermos, acabou sendo feita nessa *sprint*. Por mais desgastante que tenha sido dormir em ligação implementando funcionalidades, foi nesta reunião que me aproximei de muitos membros da equipe e o próprio professor percebeu que a equipe tinha amadurecido muito com este fato.

Acredito que eu poderia ter pedido o auxílio de algum colega para vencer minha insegurança de desenvolver nesta sprint. Pequei na comunicação e, assistindo o Felipe (AGES I) desenvolver enquanto todos os outros colegas auxiliavam ele, me encorajou novamente a programar.

2.3.5. Sprint 4

Na última *sprint* o projeto entrou em fase final de desenvolvimento, com o time focado em finalizar o máximo possível para a última entrega da aplicação. Levando as lições aprendidas de planejamento da quarta *sprint*, as *tasks* foram determinadas com antecedência para termos maior período de desenvolvimento.

Em conjunto, o time decidiu acabar com as *squads*. Pois, devido aproximação do grupo na reunião da última *sprint*, percebemos a união do time e a separação entre subequipes só prejudicaria o processo.

A *task* que escolhi realizar nesta última etapa era a exportação de dados do PDF para JSON e CSV, além de realizar a integração. Como eu nunca fiz a integração do *frontend* com o *backend* de um projeto, tive várias dúvidas de como implementar, porém, em nenhum momento senti a insegurança de desenvolver, como na última *sprint*.

Consegui implementar a *task* perfeitamente, desenvolvendo com os meus conhecimentos adquiridos no decorrer do processo e, as dúvidas pertinentes, eram solucionadas com auxílio do Felipe (AGES I). Felipe foi uma peça extremamente importante para o meu desenvolvimento como programador neste projeto.

Ademais, as outras *tasks* foram feitas continuamente pelos outros integrar e, infelizmente, não conseguimos implementar o funcionamento do botão de salvar a tempo.

Durante a apresentação, foi apontado dúvidas e detalhes finais para serem acertados antes da entrega final. Dentre elas, cometi um erro ao escolher o ícone de exportação de dados para JSON ou CSV que confundiu o *stakeholder*.

2.4 CONCLUSÃO

Minha primeira passagem pela AGES foi diferente de qualquer disciplina que tive até então, o que resultou em diversos aprendizados valiosos. Esta primeira experiência marcou por introduzir-me ao trabalho em equipe e à prática de desenvolvimento de software real.

Apesar de eu ter facilidade em me comunicar com as pessoas, tinha receio se isso valeria para o lado profissional. Neste projeto, pude transformar esta facilidade em *soft skills* que contribuíram muito com o meu crescimento profissional, visto que, foi fundamental para eu conquistar o meu primeiro emprego.

Ademais, acredito que meu desempenho técnico foi insuficiente comparado ao que desejava ter realizado. Porém, meu conhecimento adquirido neste período para realização das tarefas superou minhas expectativas. Antes da AGES, obtinha conhecimento de como implementar apenas alguns componentes de um site, como um texto ou link. Agora, graças ao meu estudo e prática nas *sprints*, tenho a capacidade de desenvolver um site inteiro estilizado conforme demanda. Este é o meu maior orgulho deste projeto.

No começo do projeto, apesar de estar nervoso por não saber se realmente desfrutaria da experiência, entrei entusiasmado pela oportunidade de obter uma amostra do que é o mercado de trabalho. Entretanto, nas *sprints* 2 e 3, percebi que deixei de priorizar o andamento do projeto e passei a me preocupar muito mais com as cadeiras adjacentes da faculdade. Acredito que este feito tenha sido consequência da data de entrega de trabalhos e provas coincidirem nas semanas em que se passaram estas *sprints*. Me desconectei muito do projeto e perdi um processo que obtinha muita curiosidade de descobrir: a transição do começo para o acabamento de um projeto. Tirarei esta lição como aprendizado para não cometer novamente quando atuarei como AGES II, III e, IV.

A presença de *stakeholders* reais, que esperam um projeto entregue conforme desejado, mudou minha forma de pensar em relação aos afazeres profissionais e

acadêmicos. A partir de agora, darei muito valor a todos os ensinamentos da faculdade que, ocasionalmente, achava que não seriam tão úteis para mim. Em resumo, todas as cadeiras cursadas até o momento desta AGES obtinham algum conhecimento que foi utilizado nessa experiência. Principalmente a disciplina de Fundamentos de Programação e Programação Orientada a Objetos, foram essenciais para compreender a lógica do código.

Como conclusão final, afirmo que aprendi a me expressar melhor em uma equipe profissional e ser mais resiliente frente aos desafios de um projeto de software. Ademais, reconheço um grande avanço tanto em *soft skills* como *hard skills* justaposto ao começo do projeto. Porém, creio que ainda tenho um grande caminho pela frente antes de me tornar um bom Engenheiro de Software.

3 – PROJETO AGES II - “ENSportive – Estilo de Vida Esportivo”

Esta seção busca apresentar minha passagem como AGES II pelo projeto ENSportive – Estilo de Vida Esportivo. Aqui estão descritos os artefatos entregues, a atuação ao longo das sprints e os pontos de melhoria identificados no decorrer do projeto.

3.1 Introdução

O projeto ENSportive tem como objetivo desenvolver um programa que inicialmente seria um aplicativo, porém, após conversar e alinhar as ideias com o stakeholder, foi decidido que seria um sistema web. Com o objetivo de ensinar esportes que envolvam raquetes dentro da metodologia Ensportive, o papel designado para nossa equipe era de facilitar o processo da Fernanda e dos professores de admnistração das aulas, assim como de ajudá-los a trazer mais pessoas para suas aulas em busca de uma vida mais saudável.

O grande desafio do projeto é a implementação do calendário: um componente similar ao Google Calendar, em que será possível marcar/desmarcar aula e administrar o conteúdo de cada aula, além de ser o mais intuitivo possível, assim como o exemplo trazido é.

O stakeholder do projeto chama-se Fernanda Ens, que nos introduziu a ENSportive: escola de raquetes que auxilia pessoas que querem praticar uma atividade física que tenha diversão e aprendizado, bem como auxiliar no emagrecimento e saúde.

A execução do projeto ocorreu no primeiro semestre de 2024, entre as datas 6 de março e 12 de junho, pelos estudantes de Engenharia de Software. Neste projeto, havia 5 AGES I, 6 AGES II, 3 AGES III e 2 AGES IV, totalizando 16 membros da equipe orientados pelo Prof. Rafael Chanin. A foto do time responsável pelo projeto pode ser vista na figura 8 a seguir:

Figura 8: Time do projeto ENSportive

Fonte: *Wiki* do projeto

3.2 Desenvolvimento do projeto

Esta seção apresenta informações referentes ao desenvolvimento do projeto: localização do código-fonte, banco de dados, protótipos de tela desenvolvidos, arquitetura e tecnologias utilizadas.

3.2.1 Repositório do código-fonte do projeto

O código-fonte do projeto foi organizado de maneira tradicional, separando o programa que interage diretamente com os usuários da aplicação que lida com a lógica de negócios, processamento de dados e outras funcionalidades que não são visíveis para os usuários finais.

O código-fonte de ambos os programas se encontra distribuído em dois repositórios, nomeados *frontend* e *backend*:

- *Frontend*: https://tools.ages.pucrs.br/ensportive/ensportive-frontend
- *Backend*: https://tools.ages.pucrs.br/ensportive/ensportive-backend

Além disso, há a wiki do projeto que explica sobre todas as tecnologias usadas no projeto, além do que foi implementado. Se encontra no presente link: https://tools.ages.pucrs.br/ensportive/ensportive-wiki/-/wikis/home.

3.2.2 Banco de Dados utilizado

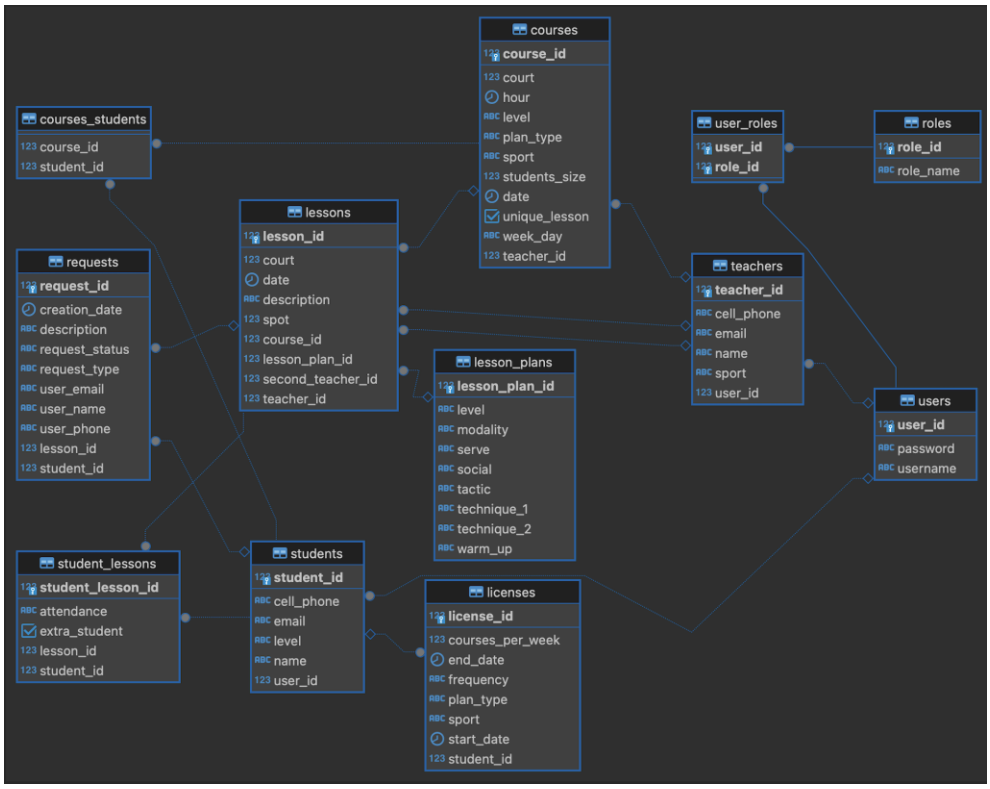
O banco de dados utilizado no projeto foi o PostgreSQL, escolhido por sua robustez, estabilidade e ampla adoção no mercado. O modelo relacional foi selecionado por permitir a organização eficiente dos dados em tabelas interligadas por relacionamentos bem definidos, garantindo integridade referencial, consistência e facilidade de manutenção. Para a conexão com o banco e o mapeamento objeto-relacional, foi utilizado o Hibernate (KING, 2001), que permitiu integrar de forma eficiente o *backend* desenvolvido em Java (Spring Boot) ao banco de dados relacional.

A decisão pelo modelo relacional foi tomada de forma coletiva entre os integrantes da equipe, após análise das necessidades funcionais do sistema ENSportive. O sistema envolve entidades fortemente conectadas — como alunos, professores, aulas e planos de ensino — e exige controle rigoroso das relações entre elas. Por esse motivo, um modelo não relacional, como o MongoDB, foi considerado inadequado, uma vez que a normalização e os relacionamentos explícitos do modelo relacional garantem maior segurança e previsibilidade na manipulação dos dados.

O PostgreSQL foi escolhido como provedor devido aos seus recursos avançados de transações ACID, suporte a chaves estrangeiras e restrições de integridade, além da facilidade de integração com o Hibernate. Outro ponto decisivo foi a sua compatibilidade com os serviços de *deploy* em AWS, o que viabiliza a escalabilidade futura do projeto.

O banco de dados foi projetado para armazenar informações essenciais à operação do sistema, como usuários, professores, alunos, aulas, planos de aula, licenças e solicitações. O diagrama da Figura 9 ilustra a estrutura do banco de dados relacional utilizado no projeto. Cada tabela representa uma entidade do domínio e os relacionamentos são definidos por chaves primárias e estrangeiras, garantindo a integridade entre os registros.

Figura 9: Diagrama de Entidades



Fonte: *Wiki* do projeto

A seguir, apresenta-se uma descrição das principais entidades:

- **users**: armazena os dados de autenticação e credenciais de acesso.
- **roles** e **user_roles**: definem os papéis de usuário e seus vínculos, controlando os níveis de permissão dentro do sistema.
- **students** e **teachers**: representam os usuários que participam ativamente das aulas, contendo informações de contato, nível e modalidade esportiva.
- **courses** e **lessons**: armazenam informações sobre os cursos e suas respectivas aulas, vinculando-as aos professores responsáveis e planos de ensino.
- **lesson_plans**: detalha a estrutura pedagógica de cada aula, incluindo modalidade, nível, aspectos técnicos e táticos.
- **requests**: registra solicitações realizadas pelos usuários, como pedidos de aula ou cancelamento.
- **licenses**: controla as licenças ativas dos alunos, relacionando a quantidade de aulas por semana e período de vigência.

- **student_lessons** e **courses_students**: realizam o relacionamento muitos-para-muitos entre alunos, aulas e cursos, permitindo o controle de frequência e participação.

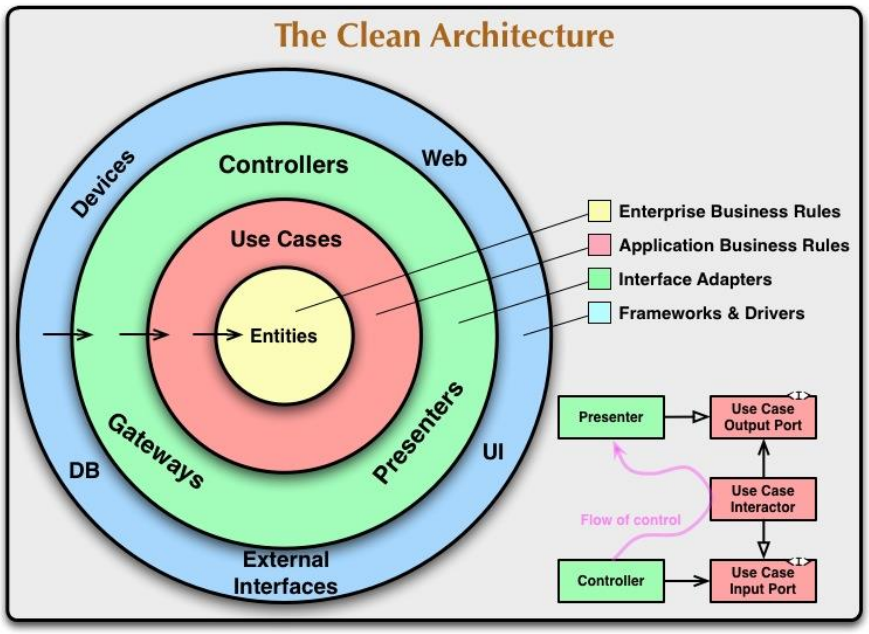
Essa modelagem buscou refletir com fidelidade o domínio da aplicação, garantindo escalabilidade, desempenho e clareza nas consultas SQL. Além disso, o uso de chaves estrangeiras permitiu uma navegação estruturada entre os dados, o que se mostrou fundamental para as operações de CRUD e para os módulos de relatórios do sistema.

Para mais detalhes sobre a estrutura e funcionamento do banco de dados, consulte a documentação disponível no seguinte link: https://tools.ages.pucrs.br/ensportive/ensportive-wiki/-/wikis/banco_dados.

3.2.3 Arquitetura utilizada

Como dito anteriormente, separamos o projeto em *frontend* e *backend*, para a estrutura de repositórios no *backend*, utilizamos o estilo Clean Architecture (MARTIN, 2020): estilo arquitetural que visa criar sistemas que sejam independentes de frameworks, testáveis e adaptáveis a mudanças ao longo do tempo. A principal ideia por trás da Clean Architecture é promover a separação de preocupações e a manutenção de uma arquitetura clara e modular.

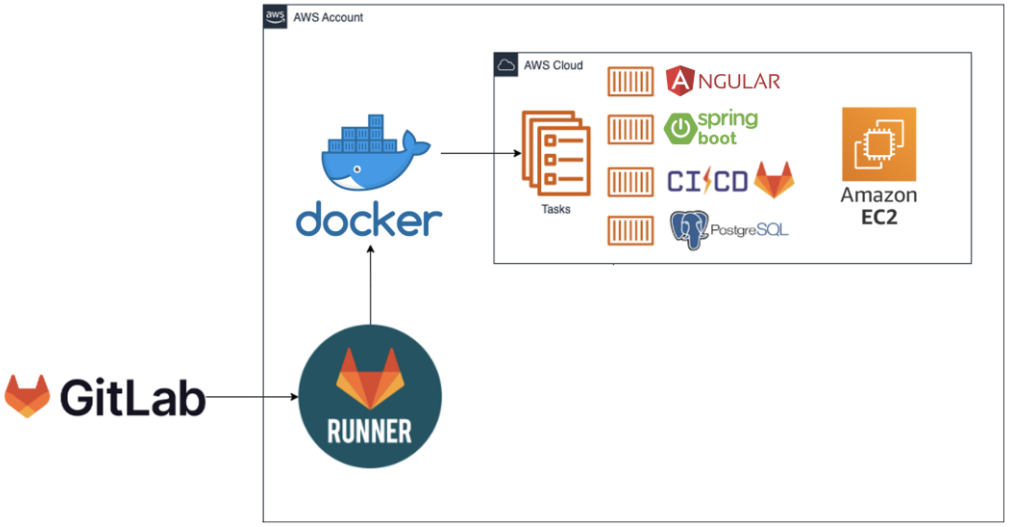
Figura 10: Estrutura do Clean Architecture



Fonte: *Wiki* do projeto

Sobre a Arquitetura *Cloud*, O fluxo do *deploy* será criar uma pipeline no Gitlab CI/CD com *runner* para fazer *build*, rodar os testes unitários, criar a imagem conteinerizada da aplicação e subi-lá no EC2 da AWS. Teremos duas instâncias: uma para *frontend* e outra para *backend*.

Figura 11: Diagrama de *Deploy*



Fonte: *Wiki* do projeto

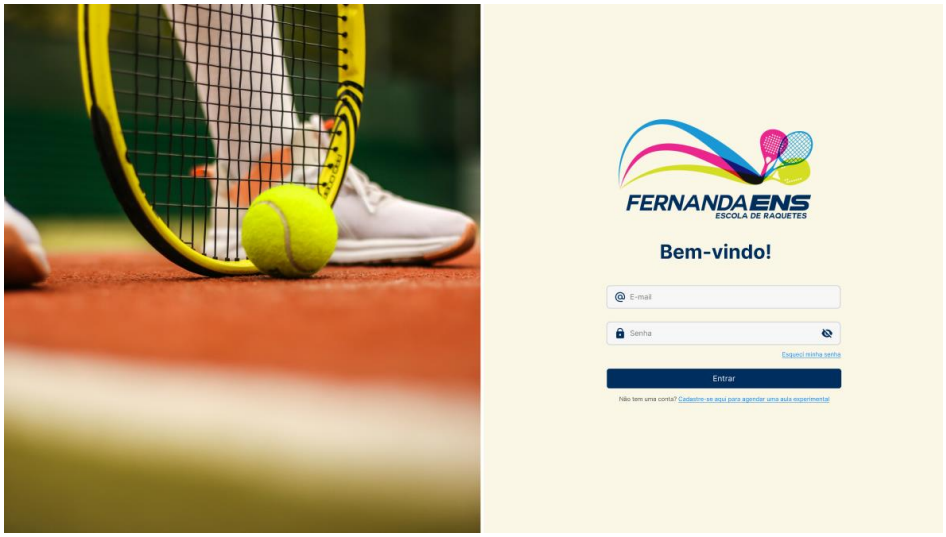
Mais detalhes sobre o *frontend*, *backend*, Infraestrutura e orçamento AWS podem ser encontrados na seção arquitetura da *wiki* em: https://tools.ages.pucrs.br/ensportive/ensportive-wiki/-/wikis/arquitetura.

3.2.4 Protótipos das telas desenvolvidas

Para realizar o protótipo das telas desenvolvidas, utilizamos o Figma, pois ninguém da equipe possuía experiência em desenvolver *mockups*. A ferramenta possui interface intuitiva e fácil de usar, permitindo que designers e equipes de desenvolvimento criem e iterem designs rapidamente. Fizemos uma pesquisa dos sites já existentes do Stakeholder e de concorrentes, para podermos achar um padrão e inspiração de práticas que poderiam acrescer no desenvolvimento. Além disso, solicitamos o padrão de cores que o Stakeholder desejava, assim, estivemos mais próximos do resultado que agradaria o cliente.

A figura a seguir representa o primeiro passo dado, no qual fiquei responsável e desenvolvi a tela de login:

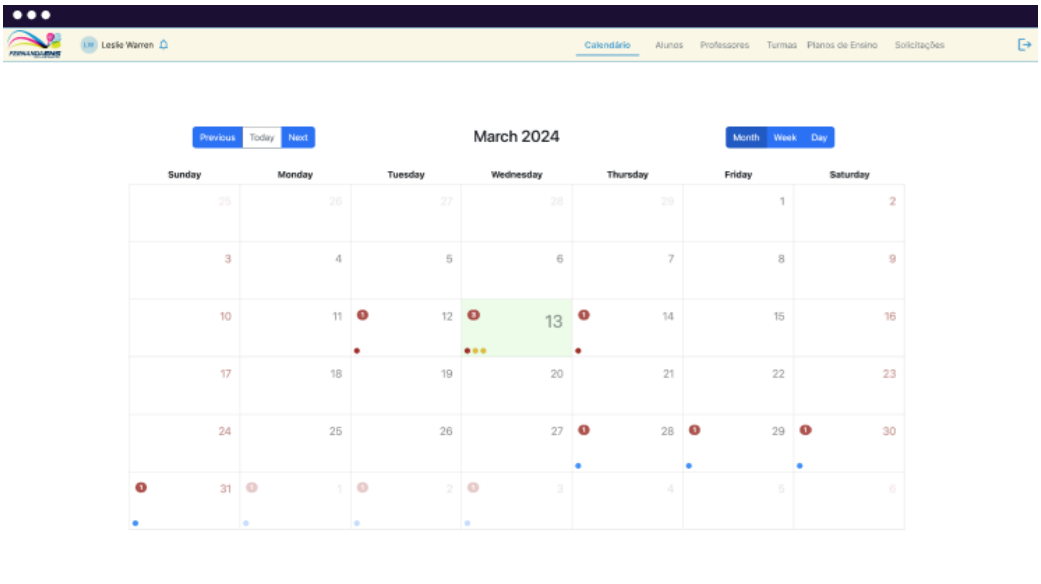
Figura 12: Tela de *Login*



Fonte: *Wiki* do projeto

A seguir, há a imagem do principal desafio do projeto que obtém o maior número de ações e funcionalidades. Esta tela será a mais utilizada e, portanto, precisa estar adequada com o desejo do cliente e intuitiva para o usuário final, constituído por professores, alunos e administradores.

Figura 13: *Home Page* do Sistema



Fonte: *Wiki* do projeto

Outros *mockups* que envolvem modais e tabelas desenvolvidos podem ser encontrados na página de *mockups* da *wiki*, localizado no link a seguir: https://tools.ages.pucrs.br/ensportive/ensportive-wiki/-/wikis/mockups.

3.2.5 Tecnologias Utilizadas

Para o lado do *frontend*, foi utilizado o Angular (GOOGLE, 2010), um framework de desenvolvimento mantido pelo Google e uma comunidade de desenvolvedores. Ele é projetado para criar aplicativos da web de página única (SPA) escaláveis e robustos. É o ideal para criar interfaces de usuário de forma rápida e eficiente, aproveitando seu sistema de componentes reutilizáveis e sua arquitetura baseada em módulos.

O desenvolvimento do código em Angular foi realizado com TypeScript: linguagem de programação de código aberto desenvolvida pela Microsoft. É um superconjunto sintático estrito de JavaScript e adiciona tipagem estática opcional à linguagem, além disso, sua estilização foi implementada através do Material UI/Design: linguagem de design desenvolvida pela Google.

Do lado do *backend*, optamos pelo Java (GOSLING, 1991) com a ferramenta Spring Boot (JOHNSON, 2014): um framework de desenvolvimento de aplicações Java que oferece uma abordagem rápida e simplificada para criar aplicativos robustos

e escaláveis. Com foco na produtividade, o Spring Boot elimina a necessidade de configuração manual, oferecendo padrões e convenções predefinidos. Ele é ideal para o desenvolvimento de aplicativos da web que requerem atualizações em tempo real.

Para a montagem do banco de dados do projeto, foi usado o PostgreSQL (STONEBRAKER, 1986): um sistema de gerenciamento de banco de dados relacional de código aberto, conhecido por sua confiabilidade, robustez e recursos avançados. Com sua capacidade de lidar eficientemente com atualizações em tempo real, o PostgreSQL é uma escolha sólida para armazenar e gerenciar dados em aplicações web e móveis que exigem uma experiência de usuário dinâmica e responsiva.

No quesito infraestrutural do projeto, usamos tanto o Docker, quanto AWS. Docker é um conjunto de produtos de plataforma como serviço que usam virtualização de nível de sistema operacional para entregar software em pacotes chamados contêineres. Já a Amazon Web Services, também conhecido como AWS, é uma plataforma de serviços de computação em nuvem, que formam uma plataforma de computação na nuvem oferecida pela Amazon.com.

3.3 Atividades desempenhadas pelo aluno no projeto

Nesta seção, estão todas as atividades individuais feitas por mim durante o projeto, assim como as ferramentas utilizadas e contribuições. Estão separadas por *sprints*, que são períodos de mais ou menos 2 semanas que, em cada uma, são determinados objetivos e entregas para o final dela.

3.3.1 Sprint 0

Após a primeira reunião com o stakeholder, iniciou-se a *sprint* 0, consistida em compreender o projeto, escolher as melhores linguagens e frameworks, além de estudarmos estas tecnologias para conseguirmos desenvolver com êxito.

Compreendi através das duas reuniões a proposta do stakeholder, juntamente com o maior desafio do projeto: um sistema de aulas em esportes de raquetes, focada em alunos agendarem e cancelarem suas aulas através do calendário, com um administrador que controla tudo isso.

Decidimos as tecnologias do projeto: Java no *backend*, Angular no *frontend* e PostgreSQL no banco de dados. Tive muitas discordâncias com os AGES III sobre a tecnologia tanto para o *backend*, quanto para o *frontend*. No *backend*, defendi a ideia de usar NodeJS (DAHL, 2009), pois seria uma grande oportunidade para todos aprenderem o Javascript, uma linguagem muito usada no mercado de trabalho. Discordaram de mim e escolheram o Java, com o argumento de equilibrar o número de colegas que aprenderão com os que já sabem e podem ajudar. No *frontend*, defendi o uso de React no projeto, usando este mesmo argumento: tanto eu, quanto outros colegas possuíam conhecimento para ajudar todos que não conheciam, que representava a grande maioria da equipe. Novamente, discordaram e escolheram Angular, buscando segurança no programa e, divergindo nos argumentos, a aventura de aprendermos todos juntos Angular, pois ninguém no projeto obtinha conhecimento. Também tivemos dificuldades na escolha da tecnologia para o banco de dados, apesar de querermos a experiência de um banco de dados com MongoDB, o diagrama de entidades retratou que o banco de dados relacional é o que melhor se encaixa para o projeto.

Minha contribuição nesta *sprint* foi baseado no Figma e no banco de dados: padronizei os estilos e cores para todos os integrantes que contribuíram no Figma possuírem uma base, além de contribuir na página de login e introduzir a *landing page.* Também ajudei o Henrique Juchem, AGES I do projeto a continuar a implementação da *landing page*. Foi a primeira vez que exerci minha função de AGES II, ajudando um colega que possuía menos conhecimento. Além disso, fiz o diagrama de entidades no banco de dados, identificando as relações entre os elementos que contribuiu na escolha de ser um modelo relacional.

Neste começo já havia percebido que seria uma AGES totalmente diferente do que foi a minha primeira. Quando fui AGES I, meus colegas de equipe eram totalmente abertos a sugestões, pois entediam que todos ali estavam aprendendo. Eu não senti isso em nenhum momento deste projeto. No final da *sprint* 4, um dos pontos negativos abordados por um dos AGES IV foi como a escolha do Angular dificultou o desenvolvimento do projeto, pois ninguém possuía um grande conhecimento com este framework.

3.3.2 Sprint 1

Com o início da segunda *sprint* do projeto, participei ativamente da escolha do PostgreSQL como tecnologia do banco de dados. Não consegui participar da modelagem em si, havia um colega que estava muito decidido do que queria fazer e, consequentemente, não estava aceitando ideias contrárias. Eu estava com muita vontade de fazer o banco de dados do projeto e isso me deixou bastante frustrado, mas serviu de aprendizado de como lidar com diferentes tipos de personalidades em um projeto.

Assim como na AGES I, a equipe foi dividida em *squads* e eu fui o único integrante da minha *squad* que possuía interesse no *backend*. Felizmente, com os conhecimentos obtidos no *frontend* realizado na AGES I, tive capacidade de auxiliar meus colegas no *frontend*. Desta vez, senti dificuldades de explicar corretamente o código e acredito que a responsabilidade de ajudar os colegas como AGES II contribui muito para a comunicação, também é importante saber explicar para transmitir o conhecimento e notei nesta *sprint* como estou em constante melhora em relação a isso.

Houve um episódio em que encontrei um calendário pronto em Angular com o código disponível para alterarmos e usarmos como acharmos melhor. Perguntei durante 3 semanas se eles havia pelo menos considerado a possibilidade do uso dele, a reposta foi negativa.

Não obstante, com ajuda do Pedro Carlucci e da Laura Caetano implementei a lógica de criação, leitura, edição e exclusão (CRUD) de professores no *backend*. Tive muita dificuldade de fazer esta tarefa por conta da minha inexperiência, mas ao mesmo tempo em que fui ajudado pelos AGES III, aprendi como funciona e me sinto muito animado para implementar mais códigos no *backend*.

3.3.3 Sprint 2

Nesta *sprint* passamos pela metade do projeto e, assim como no começo, devemos manter nossa regularidade que costuma a diminuir intensidade nesta etapa. Um grande motivador foi o encontro com o *Stakeholder*. Apresentamos o projeto e houve um grande elogio do cliente retratando que estamos no caminho certo com poucos reparos necessários.

Realizei duas tarefas neste período, ambos relacionados ao plano de ensino do sistema: primeiro desenvolvi o CRUD do plano de ensino do sistema, buscando

melhor familiaridade tanto com o *backend*, quanto com a tecnologia. Escolhi fazer mais um CRUD para reforçar meu conhecimento e implementar de forma independente uma *feature*. Assim como o esperado, tive mais facilidade com esta tarefa e me senti mais confiante com o Java. Posteriormente, assumi uma tarefa do *frontend* do projeto, pois enxerguei como uma boa oportunidade de aprender um pouco de Angular neste projeto também. Por conseguinte, implementei a tela do plano de ensino do projeto. Após a realização desta tarefa, me senti mais confortável de dar suporte aos meus colegas.

Após a realização de ambas as tarefas, na parte final da *sprint,* ocorreu a maior enchente da história do Rio Grande do Sul. O projeto teve foi pausado por 2 semanas e, devido este infortúnio, explicamos ao *Stakeholder* que se solidarizou e concordou em estender o projeto 2 semanas. Foi um acontecimento que, felizmente, não afetou diretamente nenhum integrante do time. Mesmo assim, foi difícil retomarmos e por sorte já havíamos feito todas as tarefas para entrarmos em reunião com o *Stakeholder* após estes 14 dias e estarmos com tudo entregue.

3.3.4 Sprint 3

A terceira sprint do projeto foi realizada inteiramente online. Com o time recuperado da catástrofe do Rio Grande do Sul, conseguimos realizar uma boa *sprint*. Ao apresentarmos o que realizamos para a Fernanda (*Stakeholder*), ela novamente mostrou-se contente e confiante com o projeto.

Minha primeira *task* foi adicionar o campo “Categoria” na entidade do plano de ensino do projeto. Além disso, dentro deste *enum,* havia apenas as opções Iniciante, Intermediário, Avançado e Elite. Alterei adicionando opções extras e ficou Iniciante 1 e 2, Intermediário 1 e 2, Avançado e Elite.

Minha segunda tarefa desta *sprint* foi a alteração do nome de todos os esquemas dos *Data Transfer Objects (DTOs)* do projeto. Com a implementação de outros colegas, os nomes nos *DTOs* estavam separados e, consequentemente, dando conflito com o *Swagger* do projeto.

Ambas as tarefas que realizei nessa *sprint* foram reparos, a primeira tarefa foi um ajuste solicitado pela Fernanda e a segunda tarefa uma padronização para evitar erros, dando suporte para minha equipe desenvolver com um código mais organizado e limpo.

Foi muito gratificante ver o cliente satisfeito com o rumo que o projeto está tomando, entrando nas etapas finais com a constante aprovação dela mostra que realmente estávamos entregando algo de valor. Na minha primeira AGES, desde o início o próprio Stakeholder não sabia direito o que éramos capazes de fazer e, comparando a alta expectativa que ele teve na Sprint 0 e o resultado que obtivemos, foi nítido que no final ele não estava totalmente satisfeito. Essa é a primeira vez dentro de um projeto real que lido com um cliente e ele está feliz com o resultado.

3.3.5 Sprint 4

A última *sprint* do projeto é focada em realizar as tarefas restantes para conseguirmos apresentar o programa desenvolvido até aqui com o mínimo de falhas possíveis.

O sistema foi inicialmente idealizado para haver 3 tipos de usuários diferentes: alunos, professores e administrador. Decidimos descartar o usuário de tipo Professor, pois enxergamos que não conseguiríamos entregar a tempo.

A tarefa que realizei nesta *sprint* foi de correção de turmas no *backend*. Consistia inicialmente de adicionar professor, lista de alunos e número de quadra na entidade da turma. Além disso, corrigir todos *DTOs* que estavam escritos errados no código para não ocasionar mais erros no *Swagger* e garantir que funcione tanto no cadastro, quanto na edição para que seja retornado nos *endpoints*.

Na busca de realizar mais tarefas para conseguirmos entregar dentro do tempo previsto, notei uma delas era de adicionar professor na turma, que foi algo que já fiz. Quando consultei com meus colegas de equipe, foi percebido pela Sofia (AGES IV) que houve um erro da parte dela na organização das tarefas. Apesar de já ter implementado este funcionamento, notei que era o Lucas (AGES I) que estava fazendo a tarefa e, para impulsionar o aprendizado dele, retirei o que desenvolvi do professor na minha *branch* e ajudei ele a implementar este funcionamento.

Após a finalização da tarefa, chegou o dia de apresentarmos o projeto para o *Stakeholder* e, assim como as outras apresentações, foi demonstrado bastante satisfação do nosso cliente. Foi elogiado o nosso comprometimento, restando alguns pequenos ajustes para a apresentação final.

Entre o meio e o final do projeto, percebi que não ajudei tanto os AGES I quanto desejado. O Lucas é o AGES I da minha squad e, apesar de ele ter evoluído tanto a

ponto de eu não notar a necessidade de ajudá-lo, sei que poderia ter contribuído melhor para aumentar o conhecimento dele. Sinto que esta foi a falha que cometi como AGES II.

3.4 CONCLUSÃO

O projeto ENSportive foi o segundo projeto na AGES em que trabalhei durante minha trajetória no curso de Engenharia de Software, com mais experiência em comparação ao primeiro, aproveitei ao máximo para explorar meus pontos fracos em *hard skills* e melhorar ainda mais minhas *soft skills*.

Infelizmente, não consegui participar muito do desenvolvimento do banco de dados do projeto, o que me deixou frustrado. Um dos colegas responsáveis por esta tarefa possuía um discurso que não era disponível alterações. Por outro lado, o professor me aconselhou como se comportar diante pessoas desta personalidade e achei muito válido o aprendizado em *soft skills* que o professor me ensinou. Vale ressaltar que a disciplina de Banco de Dados me auxiliou na compreensão do modelo relacional aplicado no projeto.

Acredito que o meu suporte fornecido aos AGES I do meio para o final do projeto poderia ter sido melhor, a enchente contribuiu negativamente neste processo e isso me inspirou para fazer este trabalho na minha próxima experiência da AGES.

Por ser AGES II e ter adquirido melhor conhecimento em tecnologias até chegar nesta etapa, comecei no projeto me sentindo confortável para opinar e aconselhar sobre isso, mas a comunicação com os AGES III quebrou toda a minha expectativa. Foquei bastante no *backend* para compensar a minha boa experiência na AGES I como desenvolvedor *frontend*. Fiquei muito satisfeito com o conhecimento adquirido nesta área trabalhada durante o projeto. Realizei diversas tarefas nesta área e, em comparação com o meu conhecimento no começo do projeto, acredito que aprendi bastante e foi o que mais aproveitei da minha experiência como AGES II.

Nenhuma AGES é igual a outra, foi um projeto desafiador que não foi o mais completo que participei, principalmente pelos desafios de comunicação e pela enchente que afetou o andamento. Porém, definitivamente carregarei comigo as lembranças desse time e o aprendizado que tive em minhas próximas passagens pela AGES.

4 – PROJETO AGES III - “Dashboard Operacional – Polícia Civil”

Esta seção busca apresentar minha passagem como AGES III pelo projeto Dashboard Operacional – Polícia Civil. Aqui estão descritos os artefatos entregues, a atuação ao longo das sprints e os pontos de melhoria identificados no decorrer do projeto.

4.1 Introdução

O projeto Dashboard Operacional tem como objetivo desenvolver um sistema web responsivo para ajudar a Polícia Civil de Porto Alegre a visualizar e analisar os dados de criminosos. Atualmente, o processo de análise dos dados obtidos é manual e fragmentado, dificultando a extração de insights e atrasando o tempo de resposta das investigações. Um *dashboard* operacional permitirá consolidar, categorizar e correlacionar informações de maneira mais ágil e precisa, melhorando a eficácia das investigações e contribuindo para uma maior eficiência na aplicação da lei.

O *stakeholder* do projeto chama-se Ricardo Milesi, delegado da Polícia Civil. Visto que precisaremos ajudá-lo na visualização de dados, este projeto requer muitos gráficos, o desafio dele consiste em entendermos como o cliente deseja estes gráficos e o que deve ser realizado para sincronizá-los com diferentes alvos (algumas vezes na mesma operação) com inúmeros interceptadores. Ao mesmo tempo, trabalharemos com dados sigilosos, sendo necessário garantir a segurança e confidencialidade das informações tratadas e criar um banco de dados robusto para armazenar e gerenciar as informações de interceptações.

A execução do projeto ocorreu no primeiro semestre de 2025, entre as datas 6 de março e 12 de junho, pelos estudantes de Engenharia de Software. Neste projeto, havia 5 AGES I, 4 AGES II, 5 AGES III e 3 AGES IV, totalizando 17 membros da equipe orientados pelo Prof. Marcelo H. Yamaguti. A foto do time responsável pelo projeto pode ser vista na figura 14, apresentada a seguir:

Figura 14: Time do projeto *Dashboard* Operacional

Fonte: *Wiki* do projeto

4.2 Desenvolvimento do projeto

Esta seção apresenta informações referentes ao desenvolvimento do projeto: localização do código-fonte, banco de dados, protótipos de tela desenvolvidos, arquitetura e tecnologias utilizadas.

4.2.1 Repositório do código-fonte do Projeto

O código-fonte do projeto foi organizado de maneira tradicional, separando o programa que interage diretamente com os usuários da aplicação que lida com a lógica de negócios, processamento de dados e outras funcionalidades que não são visíveis para os usuários finais.

O código-fonte de ambos os programas se encontra distribuído em dois repositórios, nomeados *frontend* e *backend*:

- *Frontend*: https://tools.ages.pucrs.br/dashboard-operacional/frontend
- *Backend*: https://tools.ages.pucrs.br/dashboard-operacional/backend

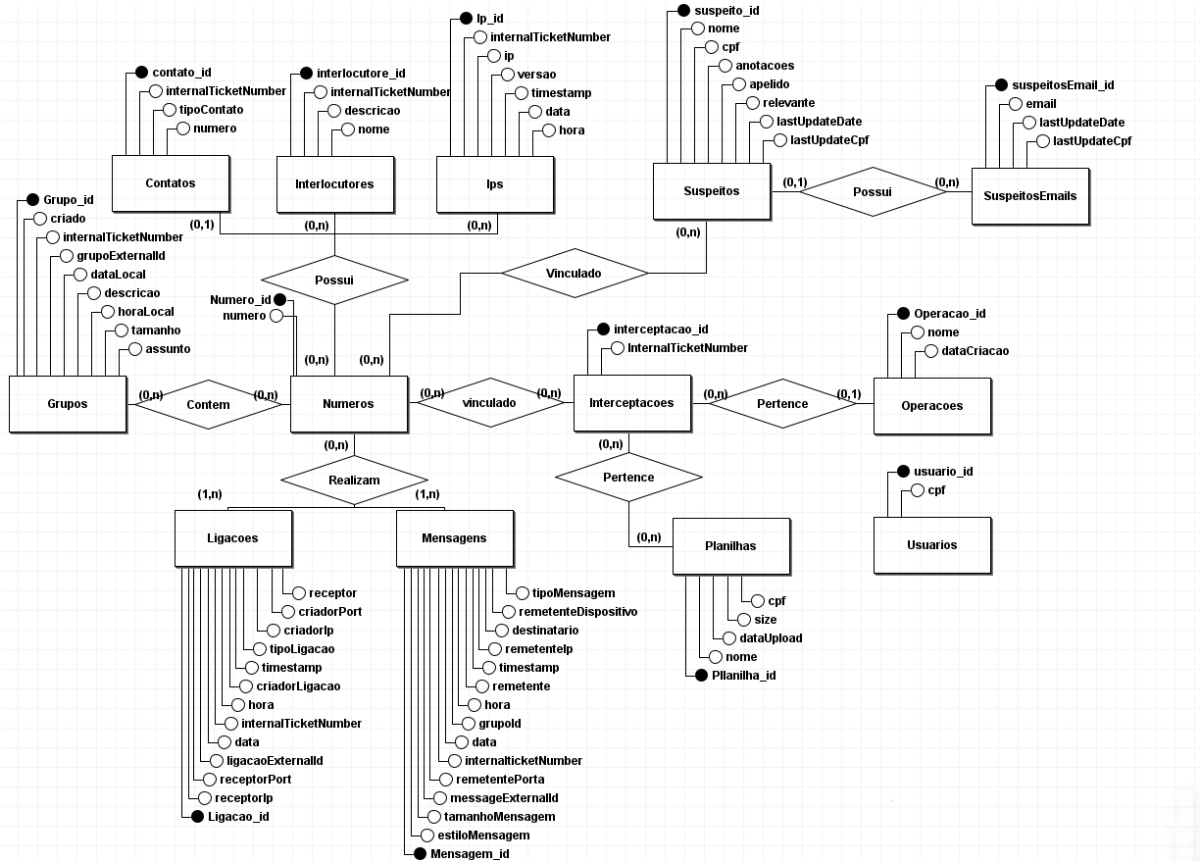
Além disso, há a wiki do projeto que explica sobre todas as tecnologias usadas no projeto, além do que foi implementado. Se encontra no presente link: https://tools.ages.pucrs.br/dashboard-operacional/wiki/-/wikis/home.

4.2.2 Banco de Dados utilizado

Ao analisar a aplicação como um todo, definimos o banco de dados do projeto como um modelo relacional e, como dito anteriormente, foi escolhida a tecnologia PostgreSQL para ser utilizada. A comunicação entre a aplicação e o banco foi facilitada pelo SQLAlchemy, uma biblioteca de mapeamento objeto-relacional (ORM) que permite manipular os dados por meio de objetos Python, promovendo maior legibilidade e organização no código.

Na figura 15 a seguir, há a modelagem conceitual do projeto que descreve as principais entidades, seus relacionamentos e respectivas cardinalidades, fornecendo uma visão clara e simplificada das interações entre as entidades-chave.

Figura 15: Modelagem Conceitual do Banco de Dados

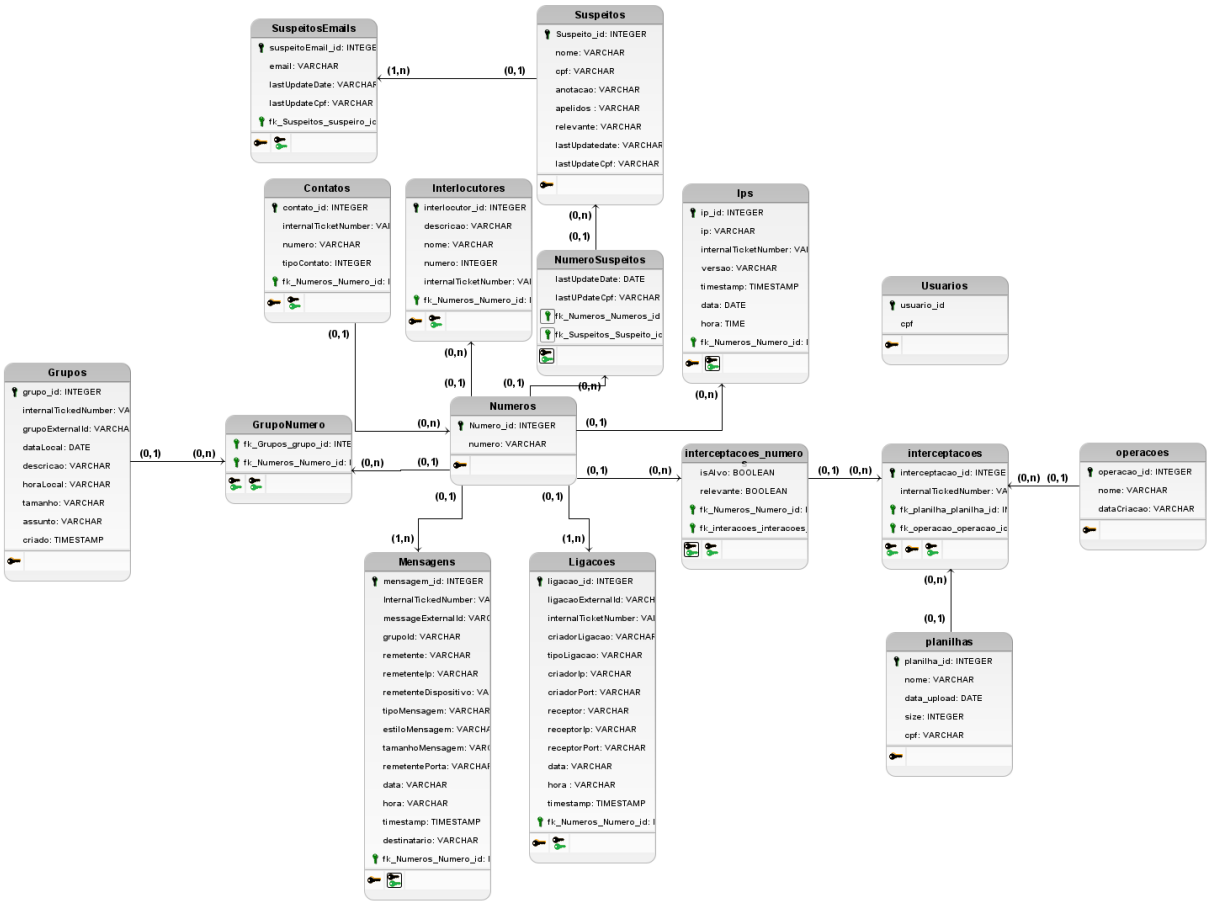


Fonte: *Wiki* do projeto

Na modelagem lógica, é convertido essa estrutura conceitual visualizada anteriormente em um esquema de tabelas relacionais, com definição de chaves

primárias, chaves estrangeiras e tabelas associativas, preparando o modelo para sua implementação prática no banco de dados. Pode ser visualizado na figura 16 a seguir:

Figura 16: Modelagem Lógica do Banco de Dados



Fonte: *Wiki* do projeto

O link desta seção está disponível para acesso em: https://tools.ages.pucrs.br/dashboard-operacional/wiki/-/wikis/banco_dados.

4.2.3 Arquitetura utilizada

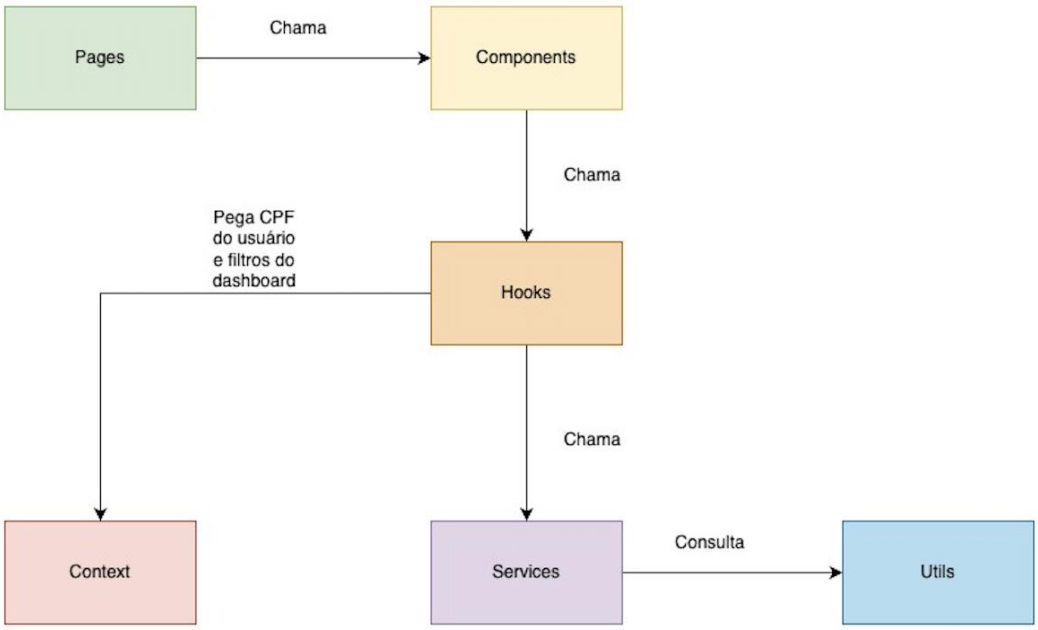
Como dito anteriormente, separamos o projeto em *frontend* e *backend*. Por ter sido uma das referências técnicas no *frontend*, todos os detalhes sobre esta arquitetura passaram pela minha revisão.

Em relação ao *frontend,* a estrutura do projeto segue uma organização modular clara, com separação de responsabilidades bem definida. O diretório principal *src* é dividido em módulos funcionais: *components* para componentes reutilizáveis, *routes*

para as páginas da aplicação, *hooks* para lógica de negócio customizada, *context* para gerenciamento de estado global, interface para definições de tipos TypeScript, e *utils* para funções utilitárias. A arquitetura de componentes segue o padrão de composição, onde componentes menores e especializados são combinados para formar interfaces complexas. Os componentes são organizados em subdiretórios funcionais, como *dashboard* para componentes específicos do painel principal, *layout* para estruturas de página, modal para componentes de *overlay* e *filters* para elementos de filtragem.

Na figura 17 a seguir, há o diagrama de componentes para facilitar a compreensão do que foi descrito sobre a arquitetura do *frontend*:

Figura 17: Diagrama de Componentes do *Frontend*



Fonte: *Wiki* do projeto

Além disso, há outros detalhes técnicos que valem a citação, resumida, mas explicativa, do que foi implementado neste sistema:

- A aplicação utiliza uma abordagem híbrida, combinando um estado global compartilhado com *custom hooks* para lógica de negócio e integração com APIs;
- Roteamento hierárquico com rotas protegidas e *layout* comum;

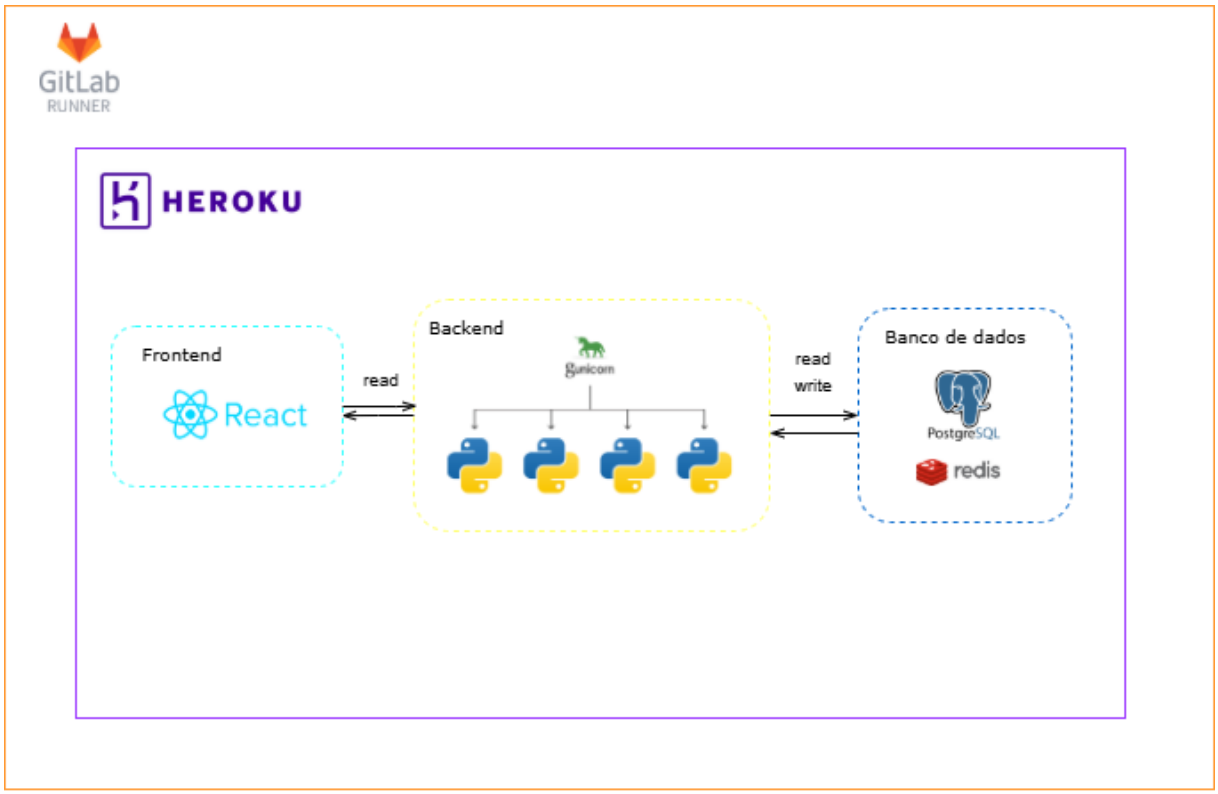
- São utilizados testes unitários, de integração e *end-to-end*. Há suporte para execução *headless*, cobertura de código e integração contínua via *CI/CD*;
- Build feito com Vite (YOU, 2020), gerando artefatos otimizados. *Deploy* configurado no Heroku (HENRY, 2007) com *serve* e *Profile*, incluindo *source maps* e melhorias automáticas de performance.

O *backend* do Dashboard Operacional é desenvolvido em Python com Flask. Apesar de não ter trabalhado neste setor, frequentemente tive contato com os desenvolvedores dele e frequentemente analisava o código. A arquitetura segue os princípios da Clean Architecture, o que garante separação clara entre camadas, essa organização favorece a modularidade. Isso resulta em testes facilitados e torna o sistema mais manutenível e escalável. A aplicação utiliza um *stack* moderno com PostgreSQL, Redis, SQLAlchemy, Pandas e Flask-RESTful, além de oferecer documentação automática via Swagger. A infraestrutura é containerizada com Docker e Docker Compose, integrando serviços como Flask, PostgreSQL, Redis e pgAdmin.

O sistema gerencia entidades complexas ligadas a operações investigativas, como suspeitos, números, IPs, mensagens e planilhas. Possui funcionalidades como upload assíncrono com acompanhamento de progresso, análise de redes de relacionamento, exportação de dados e geração de dashboards temporais. Segue o padrão PEP 8, realiza testes automatizados com Pytest, controla migrações de banco via versionamento e oferece *hot reload* e configuração por variáveis de ambiente, garantindo um ambiente de desenvolvimento eficiente e uma base sólida para aplicações críticas.

A infraestrutura da aplicação, tanto backend quanto *frontend*, está hospeada na plataforma Heroku. Possui com integração contínua via GitLab Runner facilitando o *deploy* automático após *commits* ou *merges*. Na figura 18 a seguir pode ser visualizado o diagrama da infraestrutura:

Figura 18: Diagrama da Infraestrutura



Fonte: *Wiki* do projeto

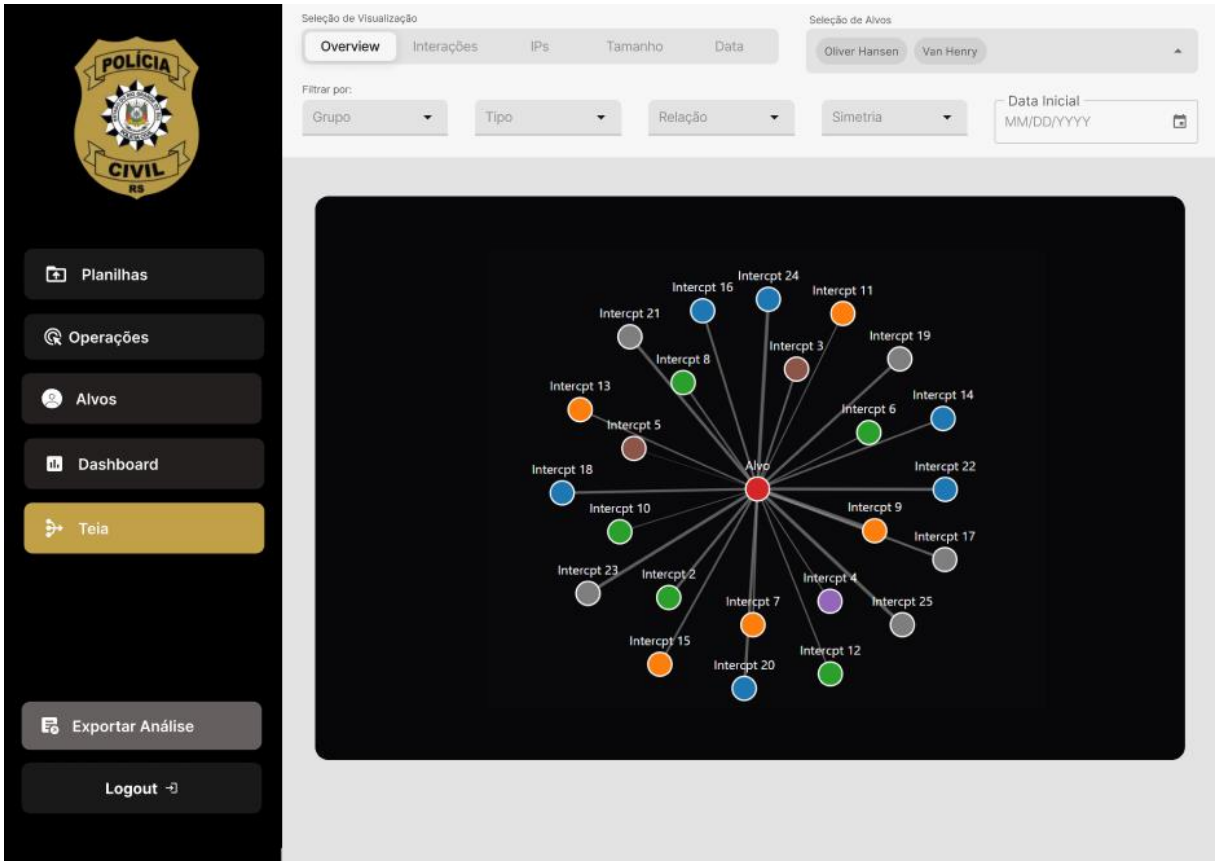
Mais informações sobre o *frontend*, *backend*, Infraestrutura e orçamento AWS poderão ser acessados na seção arquitetura da wiki em: https://tools.ages.pucrs.br/dashboard-operacional/wiki/-/wikis/arquitetura.

4.2.4 Protótipos das telas desenvolvidas

Para realizar o protótipo das telas desenvolvidas, utilizamos o Figma. A ferramenta possui interface intuitiva e fácil de usar, permitindo que designers e equipes de desenvolvimento criem e iterem designs rapidamente. Além disso, alguns integrantes da equipe já possuíam experiência adquirida no trabalho ou nas AGES anteriores.

Perguntamos para o delegado Ricardo dos sistemas já existentes, para podermos achar um padrão e inspiração de práticas que poderiam contribuir no desenvolvimento. A figura a seguir representa a tela do gráfico de teia, um pedido relizado pelo *Stakeholder* para poder visualizar melhor os seus alvos e tirar as conclusões da investigação de uma forma mais rápida e apurada:

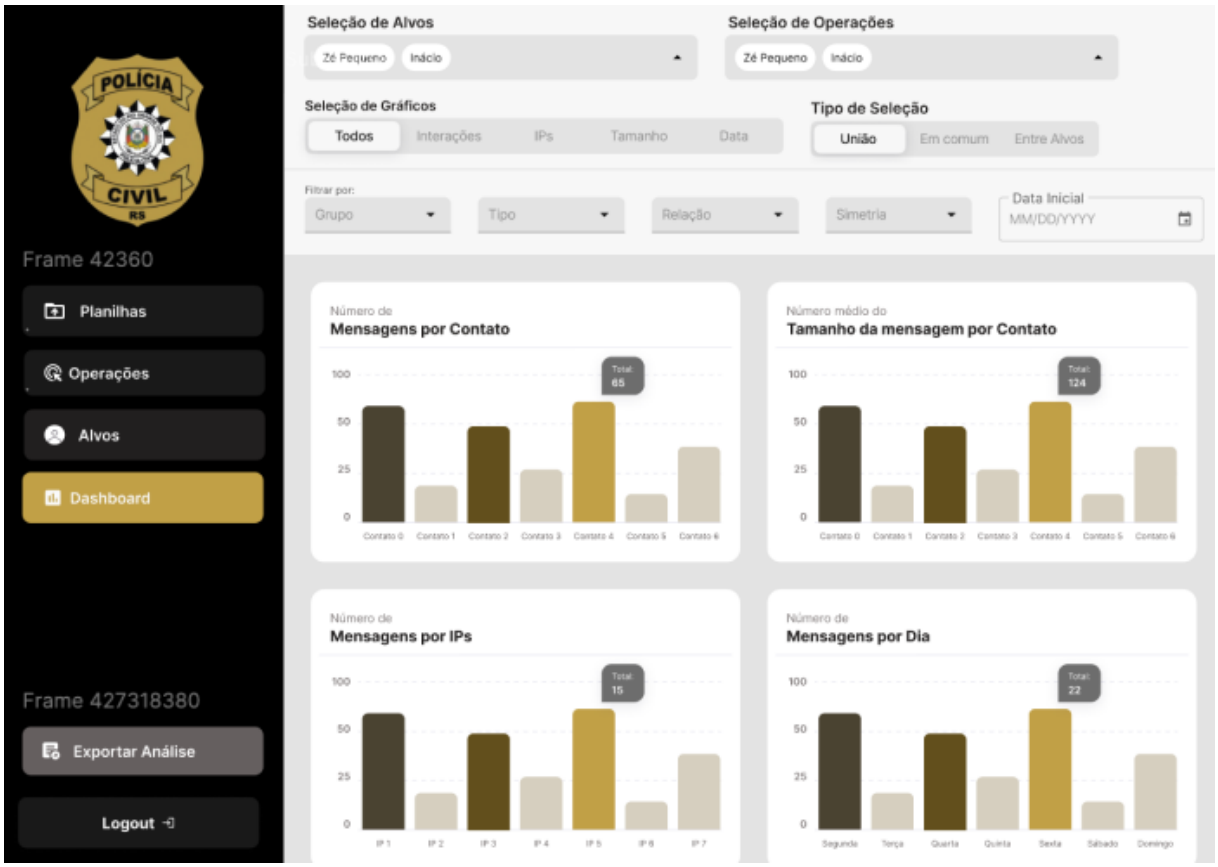
Figura 19: Tela do Gráfico de Teia



Fonte: *Wiki* do projeto

A seguir, há a figura 20 que representa um dos principais desafios do projeto que obtém o maior número de ações e funcionalidades. Nesta tela será mostrado os gráficos com diferentes dados sobre o alvo e suas intercepções, possibilitando os clientes de visualizarem com mais clareza e economizarem significativamente o tempo de pesquisa para junção de evidências:

Figura 20: Tela de *Dashboards*



Fonte: *Wiki* do projeto

4.2.5 Tecnologias Utilizadas

Para o lado do *frontend*, foi utilizado o React, uma biblioteca de desenvolvimento mantida pelo Meta e uma comunidade de desenvolvedores. Ele é projetado para criar interfaces de usuário interativas e reativas, sendo amplamente utilizado para o desenvolvimento de aplicações web de página única (SPA). Sua abordagem baseada em componentes reutilizáveis permite um desenvolvimento ágil e organizado. O desenvolvimento deste sistema foi realizado com TypeScript: uma linguagem de programação de código aberto desenvolvida pela Microsoft. TypeScript é um superconjunto de JavaScript que adiciona tipagem estática opcional, tornando o código mais seguro e fácil de manter. Além disso, sua estilização foi implementada através do Material UI: uma biblioteca de componentes baseada no sistema de design do Google, proporcionando uma interface moderna e responsiva.

Foram adotadas práticas de testes automatizados para garantir a qualidade do código. No *frontend*, utilizamos o Vitest (YOU, 2020), uma ferramenta moderna e

rápida de testes unitários para aplicações em TypeScript e JavaScript, proporcionando uma experiência semelhante ao Jest, mas com melhor desempenho. Já para testes end-to-end (E2E), utilizamos o Cypress (MANN, 2014), um *framework* poderoso que permite simular interações do usuário e validar o funcionamento da aplicação de forma eficiente e confiável.

Do lado do *backend*, optamos pelo Python com a ferramenta Flask (RONACHER, 2010): um *microframework* leve e flexível para o desenvolvimento de aplicações web. O Flask permite criar APIs de maneira rápida e eficiente, sendo ideal para aplicações escaláveis e de alto desempenho. Para a interação com o banco de dados, utilizamos SQLAlchemy (BAYER, 2006): uma poderosa biblioteca ORM (Object-Relational Mapping) que facilita a manipulação de bancos de dados relacionais dentro do ambiente Python.

Para a montagem do banco de dados do projeto, utilizamos PostgreSQL integrado a um banco de dados relacional, garantindo confiabilidade e eficiência no armazenamento e gerenciamento de dados. Sua flexibilidade e suporte a diversas engines de banco de dados fazem dele uma excelente escolha para aplicações modernas.

No quesito infraestrutural do projeto, usamos tanto o Docker quanto a AWS. Docker é uma plataforma que permite a criação e gerenciamento de contêineres, facilitando a portabilidade e escalabilidade da aplicação. Já a Amazon Web Services (AWS) é uma plataforma de serviços em nuvem que oferece soluções robustas para hospedagem, armazenamento e escalabilidade, garantindo alta disponibilidade e desempenho para a aplicação.

4.3 Atividades desempenhadas pelo aluno no projeto

Nesta seção, estão todas as atividades individuais feitas por mim durante o projeto, assim como as ferramentas utilizadas e contribuições. Estão separadas por *sprints*, que são períodos de mais ou menos 2 semanas que, em cada uma, são determinados objetivos e entregas para o final dela.

4.3.1 Sprint 0

A primeira *sprint* é sempre de apresentações e conversas para conhecer a equipe e o stakeholder. Foi neste primeiro momento que percebi que havia 5 AGES III, logo, o sistema teria que ser de muita qualidade.

Conversando com Ricardo (stakeholder), entendi a proposta do projeto e o que a Polícia Civil procura neste sistema: economizar tempo ao analisar os dados através de gráficos. Fiquei muito engajado ao citarem que planejam usar o sistema e, se ajudá-los como imaginam, poderiam repassar para outros centros da Polícia Civil no resto do país.

Após a compreensão do que seria necessário fazer neste semestre da AGES, me reuni com os outros 4 AGES III do projeto para entender onde cada um gostaria de trabalhar no projeto e definir as tecnologias que seriam usadas. Por eu trabalhar bastante no *frontend*, decidi trabalhar junto com a Luiza neste setor do projeto. Por conta de o cliente solicitar o uso de Python e Flask no *backend e* PostgreSQL no banco de dados (para seguir o padrão já utilizado na Polícia Civil), precisávamos apenas nos preocupar com a tecnologia do *frontend.* Por conseguinte, tive uma conversa com a Luiza sobre quais tecnologias ela se sentia mais confortável e, por sorte, além de concordarmos com o React, também houve a coincidência de preferirmos o Chakra UI como biblioteca.

Me deixou entusiasmado o fato de ter a Luiza no frontend com tantas concordâncias comigo. Após o projeto ENSportive ter tantos conflitos em relação a tecnologia, ter uma colega que concordou com minhas escolhas me deu um grande alívio e conforto para poder contribuir no projeto ajudando bastante os AGES I e II.

Durante esta *sprint*, estruturei a wiki do projeto colocando um padrão a ser seguido em todas as páginas da wiki. Além disso, fiz testes com o Cypress e Vitest para garantir a funcionalidade correta no código, estudei o que seria preciso para implementar um *runner* no gitlab (seria necessário para rodar a pipeline) e sugeri cursos para os AGES I e II estudarem no *frontend*. Foi uma *sprint* que ajudei bastante a equipe, mas a minha principal contribuição ainda estava por vir.

Todos os AGES III ficaram preocupados em encontrar ferramentas para a implementação de gráficos, pois foi citado na reunião com o Ricardo que seria muito conveniente um gráfico de teia, em que o interceptador que entra em contato muitas vezes com o alvo possui o fio de teia mais “grosso” que os demais. Assim, Luiza e Lucas encontraram uma biblioteca que eu testei a implementação, acredito que foi minha principal contribuição nesta *sprint*. Após um bom tempo customizando este

gráfico, consegui implementar exatamente o que o cliente precisava e, na reunião de finalização da *sprint* com o stakeholder, ele definiu o gráfico de teia como “perfeito”.

4.3.2 Sprint 1

Com o início da *sprint* 1 do projeto, começamos a desenvolver no *frontend* do projeto e, para ter maior organização em *commits* e *merge requests*, implementei a *pipeline* do projeto que havia preparado na *sprint* passada. Além disso, implementei testes unitários com alguns exemplos para que os AGES I e II pudessem se basear e construir desde o começo, a fim de aumentar a qualidade do sistema.

A equipe foi dividida em *squads* lideradas por um ou dois AGES III em cada. A minha *squad* foi formada por Eduardo Ballico (AGES IV), Leonardo Simon (AGES II) e João Sbroglio (AGES I), nossa missão para a *sprint* era desenvolver o Menu Lateral que seria usado em todo o sistema, além da Tela de Seleção de Operações. Fiquei muito satisfeito com meu desempenho neste tópico, pois consegui ajudar bastante o João a desenvolver e entender o que estava fazendo, além disso, consegui liderar ele e o Leonardo a realizarem suas tarefas, sempre me pondo a disposição de ajudar. Durante a aula, também ajudei o Lucas (AGES I), que era de outra *squad*, a finalizar a implementação da Tela de Login.

Nos últimos dias de *sprint* ajustei a responsividade destas tarefas e também refatorei o código inteiro, aplicando boas práticas para manter o código mais limpo possível. Apesar de isso ajudar muito o desenvolvimento para as próximas *sprints*, sinto que isso me atrasou bastante a refinar a *pipeline*. Eu precisaria montar o Diagrama de Deploy, Orçamento da AWS para podermos ter uma *pipeline* e *deploy* adequados para o desenvolvimento no sistema. Entendi nesta *sprint* que, apesar de ser AGES III, devo fazer minhas tarefas também. Dívidas técnicas as vezes são necessárias, o *deploy* e a *pipeline* já poderiam estar prontas se estivesse com este pensamento.

4.3.3 Sprint 2

Com o início da *sprint* 2 do projeto, demos início a implementação dos gráficos. Apesar do *backend* não ter os *endpoints*, começamos com dados fictícios para já

termos algo de valor a mostrar ao cliente no fim destas 2 semanas de desenvolvimento.

Com o acúmulo de provas e iniciando um novo projeto no meu emprego, o tempo para me dedicar a AGES começou a ser mais curto, portanto, minha contribuição foi menor em relação as primeiras *sprints*:

- Refatorei o menu lateral, foi retirado a informação do alvo neste menu, além de deixá-lo um pouco mais discreto no sistema excluindo botões de navegação que não seriam tão utilizados e adicionando a opção de adicionar planilhas;
- Guiei e revisei a tarefa de implementar o gráfico de mensagens por contato e o gráfico de tamanho de mensagens por contato realizada pelo Leonardo (AGES II);
- Implementei a página do gráfico de teia com dados fictícios, tarefa que agregou muito na apresentação ao cliente. Ricardo gostou bastante do que viu e ficou muito animado com o potencial resultado final que poderíamos atingir.

No final desta *sprint*, nós (AGES III) esquecemos de relembrar a equipe sobre o *code freeze*. Isto resultou em terminar muitas tarefas em cima da hora. Foi nesse acontecimento que percebi como é importante a comunicação com a equipe. Uma mensagem não enviada para relembrar isso quase resultou em dívidas técnicas de uma *sprint* inteira.

4.3.4 Sprint 3

A *sprint* 3 do projeto iniciou-se com uma grande surpresa: Lucas (AGES III) realizou todo o Diagrama de Deploy, orçamento e serviços da AWS para termos nosso sistema publicado. Por um lado, pensei que gostaria de ter implementado isso para aprender todos os detalhes de como desenvolver isso. Mas por outro lado, reconheci que o tempo apertado das tarefas somado aos afazeres externos me impossibilitaram de focar nisso, então fiquei muito contente que minha equipe estava sendo tão participativa e contribuinte no projeto.

Foi neste momento que comecei a concluir um pensamento que se concretizou até o fim do projeto: todos os meus colegas foram participativos. Até aqui, em outras AGES, eu sempre tive um ou outro colega que acabava não se comprometendo com o projeto. Porém, neste todos participaram (uns mais que outros) e isso me deixou a

sensação reconfortante de que seria uma das minhas melhores experiências na AGES.

Assim como nas *sprints* anteriores, mantive minha atenção nos AGES I e II do meu *squad* ao mesmo tempo que tentei contribuir com tarefas mais complexas. Além de adicionar o *Context* (globalizando variáveis no sistema como um todo), também ajudei revisando e corrigindo as tarefas de implementar a página do Gráfico de Teia com Ips (dados fictícios) do Leonardo (AGES II) e o Modal de Adicionar/Editar Emails do João (AGES I).

A partir desta *sprint* me senti com um papel de AGES III mesclado com AGES IV, pois comecei a criar tarefas e designar para a minha *squad*, além de já estar participando bastante nas reuniões guiando a equipe.

O problema desta *sprint* foi não ter integrado nenhum dos gráficos de teia, encaminhar-se para as últimas duas semanas do projeto sem nenhuma integração nas principais *features* do projeto me deixou bastante preocupado, fazendo-me começar a tentar acompanhar mais de perto o *backend*.

4.3.5 Sprint 4

As últimas duas semanas do projeto mostraram-se desafiadoras: integrar várias *features* que não possuíam *endpoints* prontos, resolver *bugs* e prever os possíveis erros que ainda poderíam vir.

No decorrer da *sprint*, consegui integrar o gráfico de teia de mensagens usando o exemplo da planilha disponibilizada pelo Ricardo. O *backend* do gráfico de teia de ips foi implementada no último fim de semana antes da apresentação, mesmo assim consegui implementar de forma parcial para apresentar tanto para o *Stakeholder*, quanto para o Ministério Público que estava na apresentação por convite ao Prof. Yamaguti. Além disso, revisei o teste de sistema que o Gabriel (AGES I) implementou e adicionei vários outros que complementavam o sistema. Acredito que estas foram as 2 semanas que mais consegui trabalhar como no começo do projeto, em que havia mais tempo para desenvolver.

Infelizmente, boa parte do motivo de entregarmos parcialmente o gráfico de teia de ips foi por conta de um AGES IV que assinou a tarefa para si mesmo e postergou a realização dela até a última hora. Apesar disso, em nenhum momento senti medo de não a entregar, pois sabia que em último caso eu teria algum dos AGES III para

fazê-la. Isso demonstra o quanto nós (AGES III) estivemos em sintonia durante toda esta experiência.

Trabalhei ativamente com o Mauro (AGES III) e a Luiza (AGES III) durante todo o projeto, mas nessa *sprint* nos comunicamos ainda mais, pois estávamos fazendo tarefas paralelamente para entregar o máximo possível.

4.4 CONCLUSÃO

O projeto Dashboard Operacional foi o terceiro projeto na AGES em que trabalhei durante minha trajetória no curso de Engenharia de Software, desta vez como uma referência técnica no projeto, tentei ao máximo ajudar a minha equipe ao mesmo tempo que realizava as tarefas mais complexas do sistema.

Eu acabei conversando com todos os integrantes do time, mas tive mais contato com o meu *squad*, os AGES III e o Eduardo Ballico (AGES IV). Esta subequipe em que mantive mais contato durante o desenvolvimento do Dashboard Operacional foi crucial para que eu pudesse desempenhar tão bem quanto desempenhei. Além de excelentes colegas de equipe, contribuíram muito com sua disposição e conhecimento técnico.

Felizmente, conseguimos entregar um produto de qualidade para o cliente, que gostou muito do que viu. Fiquei muito orgulhoso do que fiz durante o semestre e muito contente com a equipe que participei. Além de realizar um projeto divertido, também fiz amigos que manterei contato mesmo com o projeto finalizado.

Durante minha trajetória como AGES I trabalhei no *frontend* e, para compensar, na AGES II trabalhei no *backend*. Nesta AGES foquei em trabalhar no que mais contribuiria para a equipe como uma referência técnica, por isso optei por seguir no *frontend*. Apesar de estar em uma zona que me sinto mais confortável, aprendi muito fazendo o gráfico de teia, por exemplo, além dos serviços do GitLab. O que mais senti orgulho foi o quanto eu ajudei a minha equipe, sempre que pediam ajuda, eu era capaz de ajudar.

O retrospecto da experiência como AGES III é totalmente positiva, consegui me provar como uma referência técnica e aprendi muito ao mesmo tempo. Me sinto capaz e animado para ingressar como AGES IV futuramente.

5 – PROJETO AGES IV - “Plataforma de Doações para o Pão dos Pobres”

Esta seção busca apresentar minha passagem como AGES IV pelo projeto Plataforma de Doações para o Pão dos Pobres. Aqui estão descritos os artefatos entregues, a atuação ao longo das sprints e os pontos de melhoria identificados no decorrer do projeto.

5.1 Introdução

O projeto Plataforma de Doações para o Pão dos Pobres tem como objetivo desenvolver um sistema web responsivo para apoiar a Fundação O Pão dos Pobres de Santo Antônio na captação de recursos. A instituição atua há mais de 130 anos na transformação da vida de crianças, adolescentes e jovens em situação de vulnerabilidade social, oferecendo acolhimento, convivência, educação integral e aprendizagem profissional. Atualmente, o processo de engajamento de apoiadores e arrecadação de doações ocorre de forma dispersa e pouco otimizada, o que dificulta a expansão do impacto social da fundação. A plataforma digital permitirá consolidar informações, promover campanhas e oferecer diferentes formas de contribuição de maneira mais ágil, transparente e segura, fortalecendo a sustentabilidade das ações da entidade.

Os *stakeholders* do projeto são Bianca de Souza Nunes e Nilson Ayala, representantes da Fundação O Pão dos Pobres. Como o foco está na visualização de campanhas, gestão de doadores e transparência de resultados, o desafio consiste em estruturar interfaces claras e atrativas, que facilitem a interação tanto para apoiadores quanto para a administração da fundação. Também será necessário permitir que a instituição se comunique com seus parceiros e apoiadores de forma efetiva.

A execução do projeto ocorreu no segundo semestre de 2025, entre os dias 5 de agosto e 27 de novembro, pelos estudantes de Engenharia de Software. A equipe é composta por 4 AGES I, 7 AGES II, 3 AGES III e 4 AGES IV, totalizando 18 membros, orientados pelo Prof. Marcelo H. Yamaguti. A foto do time responsável pelo projeto pode ser vista na figura correspondente apresentada a seguir.

Figura 21: Time da Plataforma de Doações para o Pão dos Pobres

Fonte: *Wiki* do projeto

5.2 Desenvolvimento do projeto

Esta seção apresenta informações referentes ao desenvolvimento do projeto: localização do código-fonte, banco de dados, protótipos de tela desenvolvidos, arquitetura e tecnologias utilizadas.

5.2.1 Repositório do código-fonte do Projeto

O código-fonte do projeto foi organizado de maneira tradicional, separando o programa que interage diretamente com os usuários da aplicação que lida com a lógica de negócios, processamento de dados e outras funcionalidades que não são visíveis para os usuários finais.

O código-fonte de ambos os programas se encontra distribuído em dois repositórios, nomeados *frontend* e *backend*:

- *Frontend*: https://tools.ages.pucrs.br/plataforma-de-doa-es-para-o-p-o-dos-pobres/front-end.
- *Backend*: https://tools.ages.pucrs.br/plataforma-de-doa-es-para-o-p-o-dos-pobres/back-end.

Além disso, há a wiki do projeto que explica sobre todas as tecnologias usadas no projeto, além do que foi implementado. Se encontra no presente link: https://tools.ages.pucrs.br/plataforma-de-doa-es-para-o-p-o-dos-pobres/wiki/-/wikis/home.

5.2.2 Banco de Dados utilizado

O link desta seção está disponível para acesso em: https://tools.ages.pucrs.br/plataforma-de-doa-es-para-o-p-o-dos-pobres/wiki/-/wikis/banco%20de%20dados.

5.2.3 Arquitetura utilizada

A arquitetura do sistema foi estruturada de forma a garantir modularidade, escalabilidade e manutenibilidade, separando claramente as responsabilidades entre *frontend* e *backend*.

O *frontend* foi desenvolvido seguindo uma arquitetura baseada em componentes reutilizáveis. Nesse modelo, a interface do usuário é decomposta em pequenas unidades independentes que podem ser compostas para formar telas e fluxos mais complexos. Além disso, a aplicação adota o padrão SPA, no qual a navegação entre páginas ocorre de forma dinâmica, sem a necessidade de recarregamento completo do navegador. Esse comportamento é viabilizado pelo uso do React Router, que gerencia rotas e estados de forma eficiente. A comunicação com o *backend* funciona por meio de requisições HTTP utilizando o padrão RESTful, garantindo separação de responsabilidades entre a camada de apresentação e a camada de lógica de negócio.

O *backend* adota princípios de modularidade, injeção de dependências e programação orientada a objetos, além de ter forte inspiração em arquiteturas como Clean Architecture e *Domain-Driven Design* (DDD).

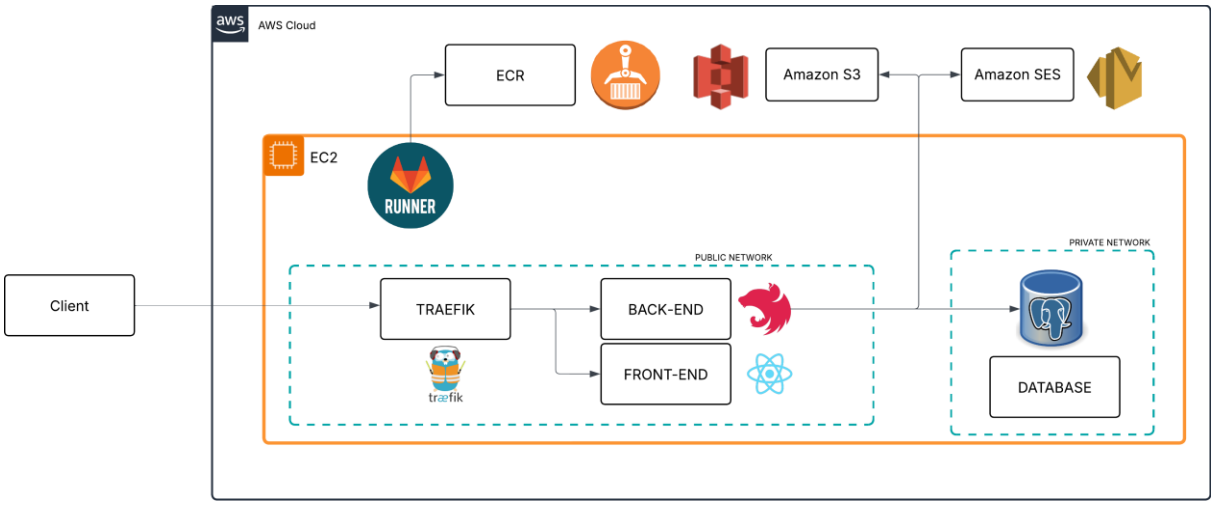
No projeto em questão, a organização das pastas demonstra claramente a aplicação desses conceitos:

- ***domain/***: representa a camada de entidades e regras de negócio puras, independentes de frameworks ou bibliotecas externas. É onde ficam os objetos de domínio, que modelam a lógica central do sistema.
- ***application/***: implementa a camada de casos de uso, que orquestram as regras de negócio e definem como o sistema responde a cada operação (ex.: criar doador, atualizar evento, autenticar usuário). Aqui também estão os DTOs, que garantem que a comunicação entre camadas seja padronizada e validada.
- ***controllers* e *modules* (implícitos no NestJS)**: embora não apareçam na listagem inicial, o NestJS organiza a aplicação em módulos, cada um agrupando controladores e serviços relacionados. Os *controllers* são responsáveis por receber as requisições HTTP e delegar as operações para os serviços, que por sua vez chamam os use cases definidos na camada de aplicação.
- ***services***: encapsulam a lógica de integração com repositórios e casos de uso, mantendo a separação entre infraestrutura e domínio.

Essa arquitetura baseada em módulos e camadas promove baixo acoplamento e alta coesão, facilitando a manutenção, os testes unitários e a escalabilidade do sistema.

Na infraestrutura do projeto, a aplicação está hospedada na Amazon Web Services (AWS) em uma instância EC2, responsável pela orquestração dos contêineres Docker. O processo de deploy é automatizado por meio de pipelines no GitLab Runner, que realizam o build e os testes da aplicação, geram imagens Docker e as publicam no Elastic Container Registry (ECR); em seguida, a EC2 executa o *pull* das imagens atualizadas e reinicia os serviços. O tráfego é gerenciado pelo Traefik, que atua como proxy reverso e balanceador de carga, garantindo roteamento eficiente das requisições e suporte a certificados SSL/TLS. Na imagem X está o diagrama de deploy:

Figura 22: Diagrama de *Deploy*



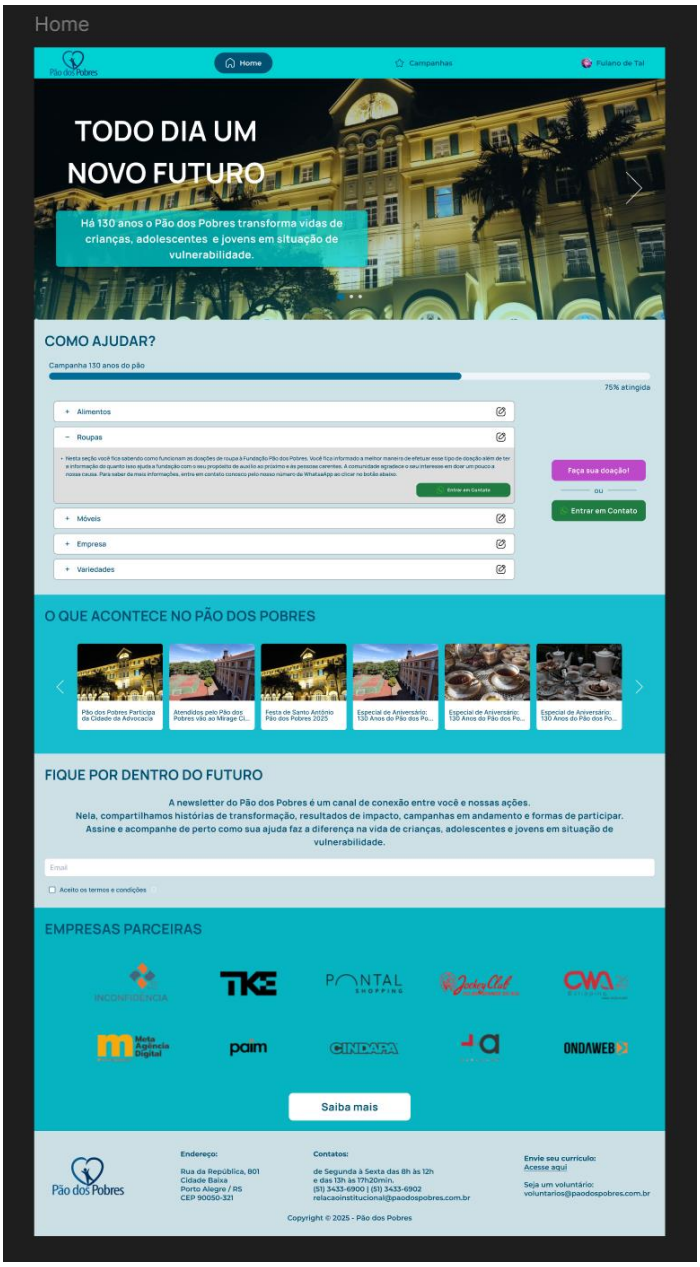
Fonte: *Wiki* do projeto

5.2.4 Protótipos das telas desenvolvidas

Para realizar o protótipo das telas desenvolvidas, utilizamos o Figma. A ferramenta possui interface intuitiva e fácil de usar, permitindo que designers e equipes de desenvolvimento criem e iterem designs rapidamente. Separamos apenas 4 membros do projeto para montar os *mockups*, alguns deles já possuíam experiência e isso adiantou o processo.

Perguntamos para os *Stakeholders* dos sistemas já existentes, para podermos achar um padrão e inspiração de práticas que poderiam contribuir no desenvolvimento. A figura 23 a seguir representa a *home page*, a principal página da aplicação, onde todos os doadores poderão navegar e obter informações da aplicação inteira:

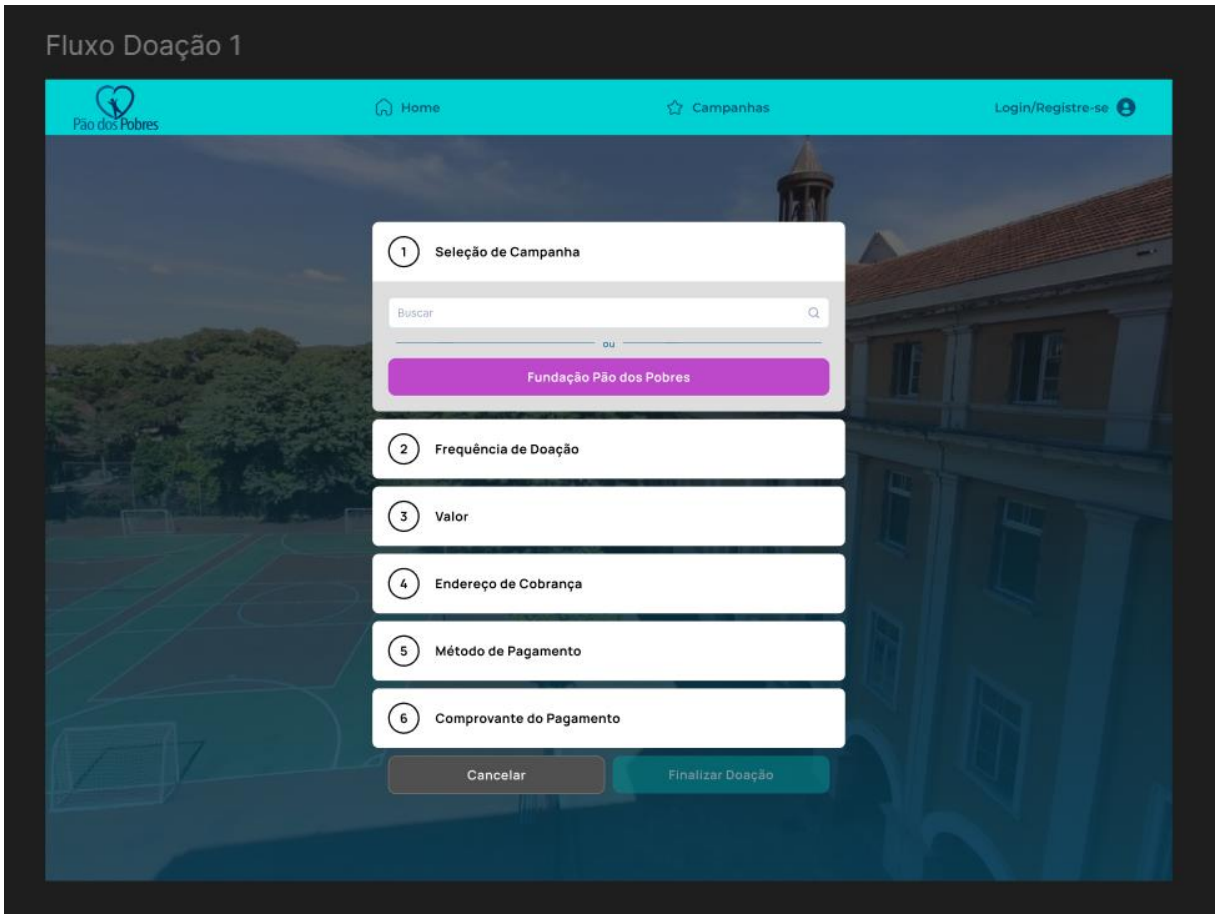
Figura 23: Tela de *Home Page*



Fonte: *Wiki* do projeto

De acordo com os Stakeholders, o objetivo deste projeto é ser o centro das doações de toda a Fundação Pão dos Pobres. A figura 24 representa o fluxo de doação, a principal tela para a maioria dos usuários (doadores):

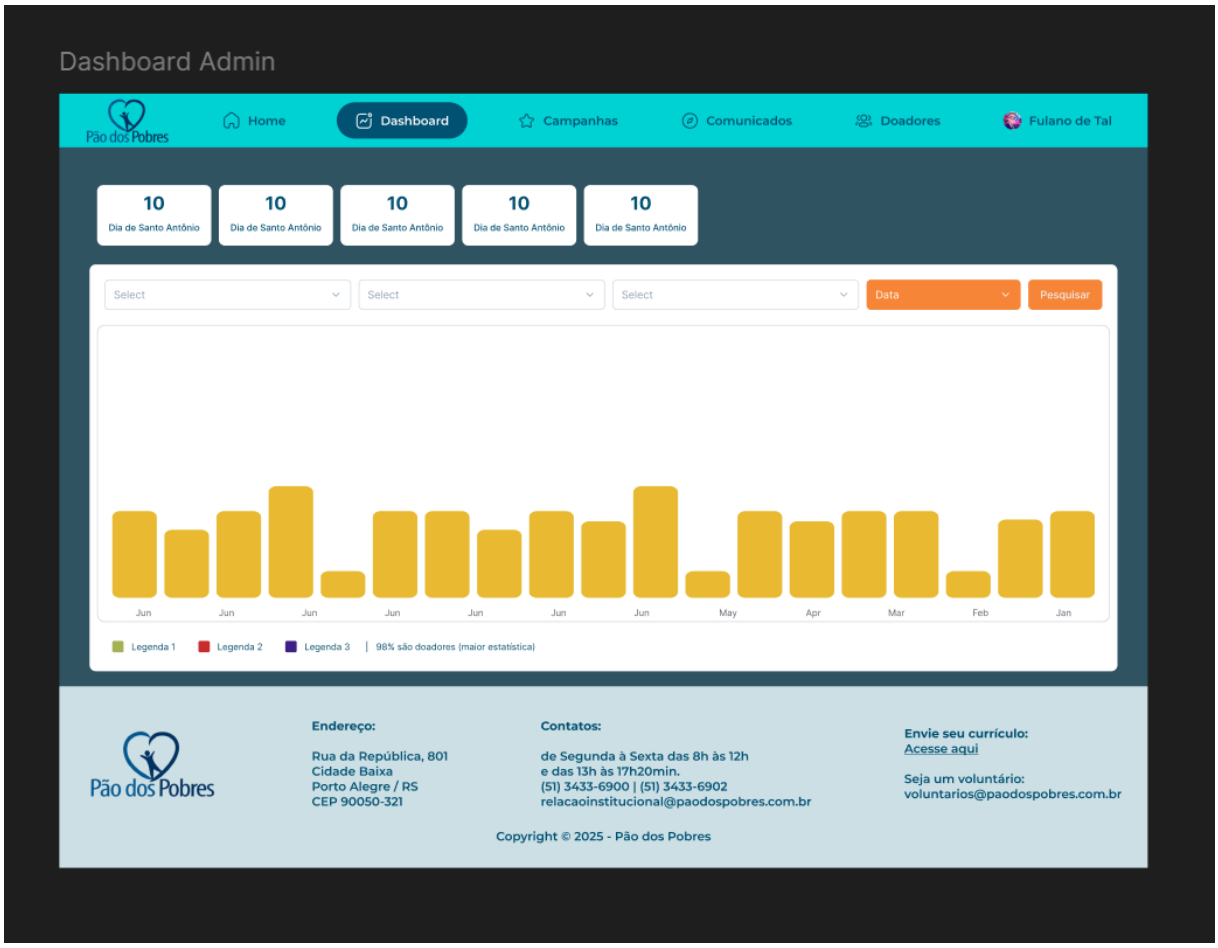
Figura 24: Fluxo de Doação



Fonte: *Wiki* do projeto

Na tela de *dashboard*, os administradores poderão visualizar todas as métricas disponíveis sobre os doadores incluindo quantidade doada, quando foi doado e entre outros filtros. A figura 25 a seguir demonstra o protótipo desta tela:

Figura 25: Tela de *Dashboard*



Fonte: *Wiki* do projeto

Todos os protótipos da aplicação estão disponíveis na wiki do projeto, pode ser visualizada neste link: https://tools.ages.pucrs.br/plataforma-de-doa-es-para-o-p-o-dos-pobres/wiki/-/wikis/design%20e%20mockups.

5.2.5 Tecnologias Utilizadas

Para o desenvolvimento do *frontend*, optou-se pela utilização do React, uma biblioteca JavaScript de código aberto mantida pela Meta (antiga Facebook) e por uma ampla comunidade de desenvolvedores. O React é amplamente utilizado para a criação de interfaces de usuário dinâmicas e responsivas, permitindo a construção de aplicações de página única de forma eficiente. Sua arquitetura baseada em componentes reutilizáveis possibilita maior modularidade, escalabilidade e facilidade de manutenção, características essenciais para sistemas de médio e grande porte.

A implementação em React foi realizada em conjunto com o TypeScript, linguagem de programação desenvolvida pela Microsoft. O uso do TypeScript no projeto contribui para a detecção precoce de erros, melhoria na legibilidade do código e aumento da produtividade da equipe, uma vez que fornece suporte a tipagem estática e recursos avançados de orientação a objetos. Para a camada de apresentação, empregou-se o Shadcn, uma biblioteca de componentes que fornece elementos visuais consistentes e acelerando o processo de desenvolvimento da interface gráfica.

No *backend*, foi adotado o NestJS (MYSLIWIEC, 2017), um framework progressivo construído sobre o Node.js. O NestJS segue princípios de modularidade, injeção de dependências e arquitetura em camadas, favorecendo a organização e escalabilidade do código. Inspirado em boas práticas de desenvolvimento de frameworks maduros, como o Angular, ele oferece suporte nativo ao TypeScript e integra-se de forma flexível a diversos bancos de dados e bibliotecas. Tais características o tornam adequado para o desenvolvimento de APIs robustas, escaláveis e de fácil manutenção.

O banco de dados escolhido foi o PostgreSQL, um sistema de gerenciamento relacional de código aberto amplamente reconhecido por sua robustez, confiabilidade e conformidade com padrões ACID. Além de recursos tradicionais de bancos relacionais, o PostgreSQL fornece suporte a dados semiestruturados (como JSON), extensões avançadas e capacidade de lidar com alto volume de transações de forma eficiente. Tais atributos tornam o PostgreSQL uma solução apropriada para aplicações que demandam consistência, escalabilidade e flexibilidade no gerenciamento de dados.

No que se refere à infraestrutura, utilizou-se Docker em conjunto com a Amazon Web Services (AWS**)**. O Docker permite a criação e execução de contêineres, garantindo portabilidade, isolamento e consistência entre diferentes ambientes de desenvolvimento, teste e produção. Por sua vez, a AWS disponibiliza uma ampla gama de serviços em nuvem, incluindo provisionamento de servidores, armazenamento escalável e monitoramento de aplicações. A integração entre Docker e AWS assegura maior agilidade no processo de *deploy*, escalabilidade elástica e resiliência da solução, características fundamentais para sistemas modernos em produção.

5.3 Atividades desempenhadas pelo aluno no projeto

Nesta seção, estão todas as atividades individuais feitas por mim durante o projeto, assim como as ferramentas utilizadas e contribuições. Estão separadas por *sprints*, que são períodos de mais ou menos 2 semanas que, em cada uma, são determinados objetivos e entregas para o final dela.

5.3.1 Sprint 0

Estive bastante ansioso para a minha última primeira *sprint*. Estou gerenciando este projeto junto com o João, Luis e Pablo, AGES IV que já se conheciam antes deste projeto, mas que me incluíram muito bem na equipe. Por conta do grande número de AGES IV, ficou combinado que ajudaríamos no desenvolvimento do sistema.

No início do projeto, juntamente com os demais integrantes do AGES IV, realizei dinâmicas de integração com a equipe, a fim de conhecer melhor os membros. Nessas atividades, levantamos as preferências de funções de cada integrante, experiências profissionais anteriores, aplicamos um formulário para mapear o conhecimento em diferentes tecnologias e organizamos os canais de comunicação do grupo (Discord e WhatsApp) para manter contato também fora do ambiente de aula.

Apresentamos o termo de abertura do projeto sob a perspectiva técnica, abordando a forma de trabalho, o conceito de *code freeze*, as ferramentas que seriam utilizadas e o fluxo de desenvolvimento. Em reunião inicial com o cliente, o stakeholder apresentou a instituição e respondeu às dúvidas levantadas. Durante a conversa, surgiu a sugestão de utilização de Inteligência Artificial, proposta que esclarecemos não estar contemplada no escopo do projeto. Uma das tarefas mais difíceis para um AGES IV é confrontar o cliente e, felizmente, consegui conduzir bem este primeiro desacordo.

Nas reuniões seguintes, alinhamos internamente a preferência pelo uso de NestJS no *backend*, em detrimento do Python, considerando a maior familiaridade da equipe com essa tecnologia. Também conduzi a primeira *daily*, explicando sua dinâmica e servindo de exemplo para os demais. Colaborei com o designer responsável (Pablo) no detalhamento do fluxo do *frontend* para estruturar

corretamente os protótipos no Figma e corrigi fluxos de usuários elaborados por outros integrantes.

Além disso, orientei integrantes mais novos em práticas de desenvolvimento, iniciei a criação de tarefas para implementação de componentes globais definidos no Figma, auxiliei na finalização do fluxo de administrador e participei do planejamento sobre a condução das próximas interações com o cliente.

Na apresentação da primeira sprint, expliquei as entregas relacionadas às *User Stories,* Estrutura Analítica do Projeto e próximos passos do projeto. Também registrei e organizei as solicitações feitas pelo cliente, como a diferenciação de permissões de administradores, inclusão de vídeo para campanhas, alteração do texto da seção “como doar” e ajustes em funcionalidades de pagamento.

Na primeira retrospectiva da *sprint*, fui responsável por conduzir a dinâmica, apresentar a ferramenta utilizada (EasyRetro) e incentivar a participação da equipe, contribuindo ativamente com comentários e sugestões. Os AGES IV iriam fazer no quadro como tradicionalmente é realizado a retrospectiva, mas acabaram gostando bastante do EasyRetro e deixaram claro o desejo de continuar realizando as retrospectivas com essa ferramenta. Além disso, expliquei algumas das tarefas já registradas no ClickUp, reforçando o alinhamento das próximas etapas de desenvolvimento.

5.3.2 Sprint 1

Começando a *sprint* de desenvolvimento do projeto, colaborei na definição de soluções para as seções de notícias e eventos, além de apoiar diretamente colegas do time, designando tarefas e garantindo que todos estivessem em atividade sem dificuldades. Não obstante, orientei integrantes mais novos (AGES III) no processo de revisão de *merge requests*, guiando-os em práticas de análise de código e qualidade.

No aspecto técnico, atuei na implementação e evolução das principais telas do sistema, iniciando pela tela de login. Contribuí para a base de componentes da tela, adicionei o Vitest, organizei o ESLint e o *Formatter*, além de apoiar ativamente nas revisões de *merge requests* de colegas (Bernardo, Léo, Arthur, Carolina e Gabriel), garantindo qualidade e alinhamento com o design no Figma. Também criei e organizei tarefas relacionadas à *home page* (notícias, *newsletter* e como doar) e à tela de perfil.

Por fim, participei da implementação do fluxo de cadastro, consolidando sua integração com a tela de login já existente.

Fui o principal apresentador na reunião do encerramento da *sprint* 1 ao cliente, na qual enfrentamos imprevistos de última hora devido à ausência presencial da principal stakeholder. Apesar da instabilidade na condução da reunião, conseguimos esclarecer dúvidas relevantes sobre eventos, notícias e gráficos, além de apresentar as entregas realizadas. Apesar de não termos entregado tudo o que foi prometido (componentes globais, tela de login e fluxo de cadastro), fiquei muito feliz com o desempenho da equipe. Muitas tarefas foram entregues e agora tenho uma noção melhor do ritmo de cada um. Também aprendi bastante em como se comportar com situações adversas. Foi inesperado que a principal *Stakeholder* não estaria presente e, infelizmente, isso me deixou bastante confuso durante a apresentação, por ter que me comunicar online e presencialmente ao mesmo tempo. Por outro lado, estou satisfeito que eu tenha aprendido isso sem tendo apresentado completamente tudo que foi desenvolvido nessas duas semanas.

Sinto que estamos indo bem no projeto, estou tentando me aproximar dos meus colegas de AGES I e II, mas percebo que ser a referência também acaba me afastando deles. Irei trabalhar para que me enxerguem como um companheiro, não um “chefe” do projeto.

5.3.3 Sprint 2

Durante a segunda sprint do projeto, atuei ativamente tanto na área de gerência de projeto quanto na parte técnica. No aspecto de gestão, continuei mantendo comunicação constante com os AGES I e II acompanhando o andamento das tarefas e garantindo que todos estivessem cientes dos prazos e responsabilidades. Também participei de reuniões com os demais AGES IV para avaliar possíveis melhorias no projeto, incluindo um breve alinhamento com o professor orientador a respeito de atrasos recorrentes na entrega de tarefas por parte da equipe de *backend*. Essa conversa me ajudou a saber como lidar com alguns colegas que não estavam entregando tarefas.

Entre as atividades de planejamento, fui responsável pela criação de novas tarefas, como a de integração com o *login* via Google**,** além de ter participado de reuniões com os clientes para discutir a viabilidade da implementação de um *gateway*

de pagamento. Nessa frente, conduzimos ainda uma reunião com a empresa parceira Payer, responsável pela API de pagamentos. Durante o encontro, os representantes da empresa explicaram o funcionamento de seu sistema e confirmaram a viabilidade da integração do Pix ao projeto, embora a funcionalidade de pagamentos recorrentes via Pix tenha sido descartada por limitações técnicas.

No âmbito técnico, realizei a revisão de diversos *MRs*, entre eles os relacionados à integração da tela de *login* com o fluxo de cadastro e a *newsletter* da página inicial. Também ofereci suporte à colega Carolina na tarefa de padronização de modais, auxiliando-a na atualização de sua *branch* com a *develop* e refatorando o componente de botão para que pudesse utilizá-lo adequadamente em sua implementação. Além disso, efetuei ajustes gerais no código, com foco em refatoração, padronização visual e responsividade dos componentes.

De forma geral, o desempenho da equipe de AGES IV foi elogiado pelo professor e pelos colegas, destacando nossa organização, capacidade de coordenação e clareza na distribuição de tarefas. Como ponto de melhoria, foi ressaltada a necessidade de maior planejamento nas apresentações com o cliente, especialmente após um imprevisto que levou à realização da reunião de forma online e sem a estrutura ideal. Pessoalmente, identifiquei a falta de comunicação dos AGES II e o não preenchimento da wiki e do banco de dados como fatores que exigiram uma postura mais firme da minha parte, o que considerei um momento importante de amadurecimento na liderança. Apesar das dificuldades, a *sprint* foi muito produtiva e recebi diversos elogios pela minha atuação e comprometimento com o projeto.

5.3.4 Sprint 3

5.3.5 Sprint 4

5.4 CONCLUSÃO

6 – CONSIDERAÇÕES FINAIS

REFERÊNCIAS

AMAZON. **AWS**. [S.l.: s.n.], 2002. Disponível em <https://aws.amazon.com/>. Acesso em 14 jun. 2023.

GOOGLE. **Angular**. [S.l.: s.n.], 2016. Disponível em: <https://angular.io/>. Acesso em 10 abr. 2024.

MARTIN, Robert C. **Clean Architecture: A Craftsman's Guide to Software Structure and Design**. [S.l.: s.n.], 2020. Livro publicado pela editora Literare Books.

MANN, Brian. **Cypress**. [S.l.: s.n.], 2014. Disponível em <https://www.cypress.io/>. Acesso em 20 jun. 2025.

HYKES, Solomon. **Docker**. [S.l.: s.n.], 2013. Disponível em <https://www.docker.com/>. Acesso em 14 jun. 2023.

HENRY, Orion. **Heroku**. [S.l.: s.n.], 2007. Disponível em <https://www.heroku.com/>. Acesso em 20 jun. 2025.

KING, Gavin. **Hibernate**. [S.l.: s.n.], 2001. Disponível em <https://hibernate.org/>. Acesso em 10 abr. 2024.

RAMIREZ, Sebastian. **FastAPI**. [S.l.: s.n.], 2018. Disponível em <https://fastapi.tiangolo.com/>. Acesso em 14 jun. 2023.

FIELD, Dylan. **Figma**. [S.l.: s.n.], 2012. Disponível em <https://www.figma.com/>. Acesso em 14 jun. 2023.

RONACHER, Armin. **Flask**. [S.l.: s.n.], 2010. Disponível em <https://flask.palletsprojects.com/>. Acesso em 20 jun. 2025.

GOSLING, James. **Java**. [S.l.: s.n.], 1991. Disponível em <https://www.java.com/>. Acesso em 10 abr. 2024.

EICH, Brendan. **JavaScript**. [S.l.: s.n.], 1995. Disponível em <https://www.javascript.com/>. Acesso em 14 jun. 2023.

TASSINARI, Olivier. **Material UI**. [S.l.: s.n.], 2014. Disponível em <https://material-ui.com/>. Acesso em 14 jun. 2023

MERRIMAN, Dwight. **MongoDB**. [S.l.: s.n.], 2007. Disponível em <https://www.mongodb.com/>. Acesso em 14 jun. 2023.

MYSLIWIEC, Kamil. **NestJS**. [S.l.: s.n.], 2017. Disponível em <https://nestjs.com/>. Acesso em 9 set. 2025.

DAHL, Ryan. **NodeJS**. [S.l.: s.n.], 2009. Disponível em <https://nodejs.org/pt>. Acesso em 9 set. 2025.

EUSTACE, Sebastian. **Poetry**. [S.l.: s.n.], 2018. Disponível em: <https://python-poetry.org/>. Acesso em 14 jun. 2023.

STONEBRAKER, Michael. **PostgreSQL**. [S.l.: s.n.], 1986. Disponível em: <https://www.postgresql.org/>. Acesso em 10 abr. 2024.

KREKEL, Holger. **Pytest**. [S.l.: s.n.], 2004. Disponível em: https://docs.pytest.org/ Acesso em 14 jun. 2023.

VAN ROSSUM, Guido. **Python**. [S.l.: s.n.], 1980. Disponível em: <https://www.python.org/>. Acesso em 14 jun. 2023.

WALKE, Jordan. **React**. [S.l.: s.n.], 2011. Disponível em: <https://reactjs.org/>. Acesso em 14 jun. 2023.

JOHNSON, Rod. **Spring Boot**. [S.l.: s.n.], 2014. Disponível em <https://spring.io/>. Acesso em 10 abr. 2024.

BAYER, Michael. **SQLAlchemy**. [S.l.: s.n.], 2006. Disponível em <https://www.sqlalchemy.org/>. Acesso em 20 jun. 2025.

TAM, Tony. **Swagger**. [S.l.: s.n.], 2010. Disponível em <https://swagger.io/>. Acesso em 14 jun. 2023.

HEJLSBERG, Anders. **TypeScript**. [S.l.: s.n.], 2012. Disponível em <https://www.typescriptlang.org/>. Acesso em 14 jun. 2023.

YOU, Evan. **Vite**. [S.l.: s.n.], 2020. Disponível em <https://vite.dev/>. Acesso em 20 jun. 2025.

YOU, Evan. **Vitest**. [S.l.: s.n.], 2020. Disponível em <https://vitest.dev/>. Acesso em 20 jun. 2025.