GABRIELA MATOS GUEDES

Sistema de microsserviços para transformação digital do terceiro setor

São Paulo
2021

GABRIELA MATOS GUEDES

Sistema de microsserviços para transformação digital do terceiro setor

Trabalho de conclusão de curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia da Computação

São Paulo
2021

GABRIELA MATOS GUEDES

Sistema de microsserviços para transformação digital do terceiro setor

Trabalho de conclusão de curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia da Computação

Área de concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Jorge Luis Risco Becerra

São Paulo
2021

RESUMO

As instituições do terceiro setor brasileiro carecem de mão de obra [3] e hoje não há ferramentas acessíveis para que essas organizações possam utilizar e digitalizar seus processos. É necessário, portanto, o desenvolvimento de uma aplicação que atenda a essas demandas. O projeto aqui apresentado consiste em uma aplicação em microsserviços desenvolvida usando o framework Design Science [9] e com elementos da metodologia ágil. Sua implementação foi feita com tecnologias bem difundidas no mercado, permitindo, assim, que o sistema seja flexível, modularizado, aberto a mudanças e de fácil uso, o que é perfeito para atender os mais diversos casos de uso de diferentes organizações. As funcionalidades existentes no produto final são as que foram consideradas como essenciais para o cotidiano de uma empresa do terceiro setor, que são: suporte a doações, gestão de funcionários e voluntários, gerenciamento de projetos, controle financeiro, controle de marketing digital, geração de relatórios de transparência e cadastro de notas fiscais doadas. O resultado é uma aplicação Web com uma ampla cobertura de testes automatizados e que possui interface tanto para o administrador da organização quanto para seus potenciais doadores.

Palavras-chave: microsserviços. organização não-governamental. terceiro setor. sistema acessível. gerenciamento de empresa.

ABSTRACT

The organizations from the Brazilian third sector lack labor [3] and today there are no accessible tools that these institutions can use and transform their processes to a digital one. Therefore, it is necessary to develop an application to attend to these needs. The project done here is a microservice application developed using the Design Science framework [9] and with elements from the Agile methodology. The implementation was done with popular technologies, so the system is highly flexible, modularized, open to changes and easy to use, which is perfect to match the needs from the different organizations use cases. The features the product has are the ones that were considered as essentials for the third sector company daily routine, which are: support for donations, volunteers and employees management, projects management, financial control, digital marketing control, transparency reports generation and invoices registration. The result is a web application that is vastly covered by automated tests and that has an interface for both organization administrators and their potential donors.

Keywords: microservices. nonprofit organization. third sector. accessible system. company management.

LISTA DE FIGURAS

Figura 1	Monolito vs. Microsserviços	16
Figura 2	Diagrama Design Science	24
Figura 3	Arquitetura computacional em camadas	25
Figura 4	Modelo de implementação	29
Figura 5	Serviços no Docker	30
Figura 6	Estrutura das pastas	31
Figura 7	Inicialização dos containers	32
Figura 8	Exemplo de Dockerfile	32
Figura 9	Exemplo de .env	33
Figura 10	Diagrama do banco de funcionários	35
Figura 11	Diagrama do banco de voluntários	36
Figura 12	Diagrama do banco de projetos	37
Figura 13	Diagrama do banco de marketing digital	38
Figura 14	Diagrama do banco de controle financeiro	39
Figura 15	Execução de teste automatizado	42
Figura 16	Tela inicial da interface do usuário	43
Figura 17	Tela de doações da interface do usuário	44
Figura 18	Tela de doações da interface do usuário (cont.)	44
Figura 19	Tela de transparência da interface do usuário	45
Figura 20	Planilha exportada da tela de transparência	45
Figura 21	Tela inicial da interface do administrador	46
Figura 22	Tela de doações da interface do administrador	46
Figura 23	Filtros de doações da interface do administrador	47
Figura 24	Tela de doadores da interface do administrador	47
Figura 25	Tela de funcionários da interface do administrador	48
Figura 26	Tela de áreas de funcionários da interface do administrador	48
Figura 27	Tela de projetos da interface do administrador	49
Figura 28	Descrição de um projeto	49
Figura 29	Tela de projetos finalizados da interface do administrador	50
Figura 30	Tela de transações da interface do administrador	50
Figura 31	Tela de transações agrupadas da interface do administrador	51
Figura 32	Tela de metas da interface do administrador	51

Figura 33 Tela de rascunhos de postagens da interface do administrador 52
Figura 34 Tela de edição de postagem da interface do administrador 52
Figura 35 Tela de publicações postadas da interface do administrador 53
Figura 36 Tela de configuração dos relatórios da interface do administrador 53
Figura 37 Tela de notas fiscais doadas da interface do administrador 54
Figura 38 Tela de configurações da interface do administrador 54
Figura 39 Tela de configurações da interface do administrador (cont.) 55
Figura 40 Tela de configurações da interface do administrador (cont.) 55

LISTA DE TABELAS

Tabela 1	Cronograma	15
Tabela 2	Cronograma de Sprints	26
Tabela 3	Principais pontos positivos e negativos do Node.js	28
Tabela 4	Pontos positivos destacados de cada banco de dados.	29

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CSV	*Comma Separated Values*
IPEA	*Instituto de Pesquisa Econômica Aplicada*
MVP	*Minimum Viable Product*
ONG	*Organização Não Governamental*
OSC	*Organizações da Sociedade Civil*

SUMÁRIO

INTRODUÇÃO **11**

1 CONTEXTO **12**

1.1 CENÁRIO ATUAL 12

1.2 OBJETIVO 12

1.3 JUSTIFICATIVA 13

1.4 METODOLOGIA 14

1.5 ORGANIZAÇÃO DO TRABALHO 15

2 ASPECTOS CONCEITUAIS **16**

3 PRODUTO PRINCIPAL **18**

3.1 CONTEXTO DA APLICAÇÃO 18

3.1.1 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS 20

3.2 MODELO CONCEITUAL 23

3.3 PRODUTO PRINCIPAL 24

3.3.1 ESTRUTURA DO PRODUTO 24

3.3.2 CICLO DE OPERAÇÃO 25

3.3.3 DESENVOLVIMENTO DO PRODUTO 26

3.4 IMPLEMENTAÇÃO 27

3.4.1 TECNOLOGIAS UTILIZADAS 27

3.4.2 DESCRIÇÃO TECNOLÓGICA DO PRODUTO 30

3.4.2.1 SERVIÇO DE DOAÇÕES 33

3.4.2.2 SERVIÇO DE AUTENTICAÇÃO 33

3.4.2.3 SERVIÇO DE FUNCIONÁRIOS 34

3.4.2.4 SERVIÇO DE VOLUNTÁRIOS 35

3.4.2.5 SERVIÇO DE PROJETOS 37

3.4.2.6 SERVIÇO DE MARKETING DIGITAL 38

3.4.2.7 SERVIÇO DE CONTROLE FINANCEIRO 38

3.4.2.8 SERVIÇO DE RELATÓRIOS 40

3.4.2.9 SERVIÇO DE NOTAS FISCAIS 40

3.4.2.10 SERVIÇO DE CONFIGURAÇÕES 40

3.4.2.11 SERVIÇO DE INTERFACE DO ADMINISTRADOR 40

3.4.2.12 SERVIÇO DE INTERFACE DO USUÁRIO 41

3.5 TESTES 41
3.6 RESULTADOS 42
4 CONSIDERAÇÕES FINAIS 56
4.1 CONCLUSÕES 56
4.2 EVOLUÇÕES DA MONOGRAFIA 56
REFERÊNCIAS 58

Introdução

O terceiro setor brasileiro possui um número significativo de instituições, mais de 800 mil [1]. Além de enfrentar dificuldades para conseguir recursos financeiros, essas organizações também sofrem com falta de mão de obra para realizar suas atividades cotidianas e não há uma plataforma de sistemas de baixo custo no mercado que seja capaz de atender às mais diversas demandas de todas ONGs, pois são aplicações monolíticas, com uma interface simples e não escaláveis.

A solução é, então, oferecer para essas instituições um sistema de microsserviços que permita sua transformação digital e que atenda às suas necessidades de forma simples e modular.

1. Contexto

1.1. Cenário atual

O terceiro setor é constituído por entidades privadas e de utilidade pública, também conhecidas como Organizações da Sociedade Civil (OSCs). Segundo dados extraídos do próprio IPEA (Instituto de Pesquisa Econômica Aplicada) [1], havia mais de 800 mil OSCs no Brasil em 2016 e não há uma concentração das instituições nas capitais - a proporção é a mesma de acordo com o número de habitantes. Embora trate-se de organizações sem fins lucrativos, as OSCs precisam de recursos para manter sua gestão e operação e, portanto, algumas recebem recursos federais, que, no entanto, sofreram uma diminuição significativa nos últimos anos [2], dificultando ainda mais seus trabalhos.

Outro dado importante a ser ressaltado é que 90% de todas as OSCs têm até 2 empregos formais [3], o que representa um número baixíssimo de pessoas trabalhando nas organizações com um foco principal nelas. Dessa forma, fica complicado lidar com toda a gestão administrativa exigida para sua existência.

É nítido que as organizações carecem de um sistema que auxiliem-as em diversas etapas de seu processo de gestão, tal como gerência de funcionários e voluntários, suporte para controle financeiro, entre outros. Trata-se de um problema social, em que as empresas de Tecnologia da Informação não entendem os problemas das instituições do terceiro setor e criam produtos que não são necessários para elas.

1.2. Objetivo

O objetivo principal é o desenvolvimento de uma solução que atenda às demandas do terceiro setor. Um dos conceitos utilizados no desenvolvimento é o de arquitetura de microsserviços. Esse tipo de arquitetura tem ganhado popularidade nos últimos anos e isso se deve ao fato de que ela permite que as aplicações sejam mais escaláveis, flexíveis e modularizadas [4], nos trazendo menos erros durante a implementação de novas funcionalidades. Além disso, embora existam linguagens

específicas para o desenvolvimento de microsserviços, como a Jolie, a sua implementação também é possível em uma vasta gama de tecnologias disponíveis.

Uma arquitetura baseada em microsserviços permite a evolução do sistema de forma organizada e sem grandes dificuldades tecnológicas, pois cada serviço é independente. Logo, a manutenção terá baixo custo, diferentemente de um sistema monolítico onde a manutenção será no sistema como um todo.

Para oferecer um bom suporte à gestão das organizações, os principais serviços que a plataforma deve oferecer são:

- Serviço de doação
- Gestão de funcionários
- Gestão de voluntários
- Gerenciamento de projetos
- Suporte para controle financeiro
- Suporte para controle de marketing digital
- Serviço de relatórios de transparência
- Estatísticas de notas fiscais doadas

Cada uma destas funcionalidades é um microsserviço e, portanto, ficará a critério da instituição quais ferramentas fazem sentido serem utilizadas ou não.

1.3. Justificativa

Sabendo que as OSCs são significativas na sociedade brasileira, tem-se de forma um pouco mais clara a necessidade da criação de um sistema que automatize diversos processos dessas instituições. Hoje, já existem algumas ferramentas disponíveis com esse fim, como por exemplo o Portal Ongfácil [5] ou Kad ONG [6], mas trata-se de soluções que são precárias, com sistemas monolíticos, sem uma interface tão intuitiva para o usuário. Além disso, esses sistemas que existem não permitem a flexibilização necessária para que seja capaz de atender às necessidades das mais diversas OSCs, cujas áreas de atuação variam desde o setor da Saúde até a área de Cultura e Esportes e, portanto, possuem demandas distintas.

Outro diferencial do software implementado, quando comparamos com as soluções já existentes no mercado, é a sua disponibilização para as instituições: se trata de uma ferramenta totalmente gratuita e *open source*, que está disponível para que outros desenvolvedores também possam adicionar novos serviços caso queiram e que as organizações sintam-se livres para alterar o código fonte conforme sua necessidade, além de toda customização que a plataforma já oferece. Assim, além de ser financeiramente acessível para as OSCs, também terá toda a flexibilidade necessária.

Vale ressaltar que a monografia é uma continuação do trabalho *Microsserviços como abordagem arquitetônica aplicada para viabilizar a transformação digital de instituição do terceiro setor brasileiro* [7], tendo o mesmo como base. Neste projeto base, foi feito uma análise detalhada da tecnologia no terceiro setor brasileiro e um sistema para cadastramento de notas fiscais com testes de usabilidade na organização Instituto Adiante [8], tendo resultados satisfatórios uma vez que agradou os gestores responsáveis pela organização.

1.4 Metodologia

Como metodologia do trabalho desenvolvido, foi utilizado como base o *framework* Design Science [9], cuja ideia principal é produzir um artefato para um contexto, visando melhorá-lo para resolver um problema do mundo real. Segundo o *framework*, há somente uma solução (artefato) para o problema (*design problem*), que é alcançada levando em conta a base teórica (*knowledge context*).

Para conseguir produzir esse artefato (ou solução), é necessário entender o contexto dos *stakeholders* e o objetivo do projeto. No caso deste trabalho, pode-se definir os *stakeholders* como sendo os desenvolvedores e as pessoas que terão um contato direto com o sistema final, ou seja, administradores das organizações e até mesmo seus doadores de recursos financeiros. O objetivo principal, conforme já esclarecido acima, é resolver, por meio de uma plataforma em arquitetura de microsserviços, o *design problem* representado pela dificuldade de gerenciamento das OSCs. Isso foi feito considerando como *knowledge context* conceitos de Computação.

Pensando na implementação do sistema em si, foi utilizada também a metodologia ágil, definida pelo Manifesto Ágil [10], um conjunto de princípios a serem seguidos para um bom desenvolvimento de software. A metodologia ágil foi a utilizada para a implementação deste projeto, pois foi necessário um desenvolvimento incremental e uma alta adaptabilidade às mudanças que possam ocorrer. Entre os exemplos desta metodologia, temos o Scrum, que também foi usado no projeto. No Scrum, o desenvolvimento é quebrado em iterações (*sprints*) e cada uma delas possui um planejamento de ferramentas a serem implementadas e um tempo para conclusão.

Considerando a construção de todo o trabalho, desde sua especificação até a finalização de sua monografia, foi seguido o seguinte cronograma:

Tabela 1.4.1 - Cronograma

	Mês											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Especificações e elaboração do plano de implementação												
Implementação do sistema												
Ajustes necessários e correção de *bugs*												
Finalização da monografia												

Fonte: Produção própria

1.5. Organização do Trabalho

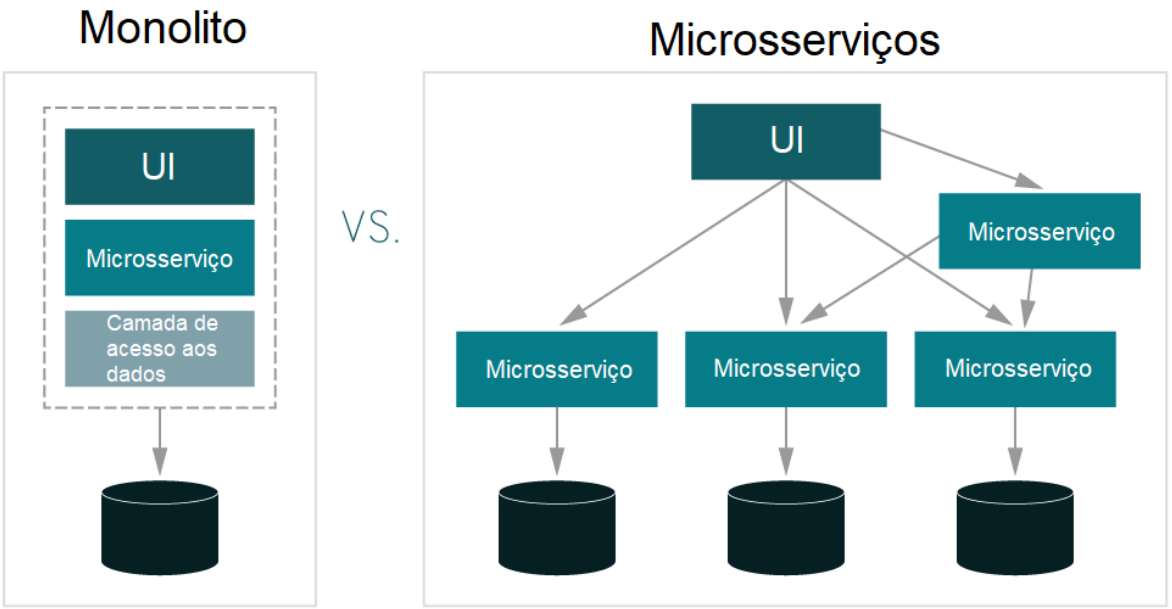
Este trabalho foi confeccionado de acordo com as *Diretrizes para Apresentação de Dissertações e Teses* da Escola Politécnica da Universidade de São Paulo. Neste primeiro capítulo, é feita uma introdução do tema a ser apresentado. Em “Aspectos conceituais”, são apresentados os principais conceitos técnicos empregados, relativos à área de aplicação de trabalho. No capítulo de “Tecnologias Utilizadas”, há uma apresentação e explicação das principais ferramentas usadas para a construção do projeto. Em “Metodologia do Trabalho”, é mostrado quais foram as

metodologias utilizadas e o planejamento seguido para seu desenvolvimento. Por fim, há “Especificação preliminar de Requisitos do Sistema”, em que são apresentados os requisitos do sistema, com diagramas e esquematizações de arquitetura.

2. Aspectos conceituais

Para a construção de sistemas, é possível utilizar uma arquitetura monolítica ou de microsserviços. Em uma arquitetura monolítica, temos o sistema inteiro construído em uma única unidade, com seus módulos e funcionalidades altamente acoplados. Por um lado, os monolitos têm seus benefícios uma vez que são mais simples de construir, sendo ótimas soluções para sistemas que não sejam tão robustos. Por outro lado, eles têm desvantagens por não permitirem uma fácil alteração de suas funcionalidades e a escalabilidade é mais limitada. A arquitetura em microsserviços, conceito central deste projeto, soluciona esses contras da arquitetura monolítica. A ideia é que o sistema seja composto por vários pequenos módulos independentes e que se comunicam entre si por APIs, sendo cada módulo um *microsserviço*. Dessa forma, é possível a construção de aplicações escaláveis, flexíveis e modularizadas [4].

Figura 2.1 - Monolito vs. Microsserviços



Fonte: Traduzida de RedHat [9]

De acordo com o livro *Microservice Architecture* [11], as melhorias que a adoção de uma arquitetura de microsserviços traz são várias, tais como: confiabilidade, segurança, usabilidade, facilidade de implementar mudanças, entre outras.

Outro conceito pertinente ao trabalho é o de *Clean Architecture* [12], um conjunto de princípios para o desenvolvimento de arquiteturas de forma que minimize os esforços de manutenção e extensão a longo prazo. Entre os princípios a serem seguidos, podemos citar um dos mais importantes, o Dependency Inversion Principle (DIP), representado pela letra D na sigla SOLID[1]. Esse princípio defende a ideia de que as dependências de um software deve se referir às abstrações e não aos módulos e classes concretos, permitindo, assim, uma maior flexibilidade do código, uma vez que não há uma forte dependência entre funcionalidades distintas.

[1] Conjunto dos cinco principais princípios de Clean Architecture [11], onde cada letra representa um princípio

3. Produto Principal

3.1 Contexto da aplicação

O produto principal resultante deste trabalho trata-se de um sistema baseado em microsserviços para o gerenciamento das atividades de uma instituição do terceiro setor. Por ora, trata-se de um MVP já completo para as necessidades do usuário. No total, terão oito principais microsserviços a serem utilizados, são eles:

- **Serviço de doação:** é interessante para as OSCs que tenham diversos meios de arrecadar doações. A ideia é que este serviço permita a doação por meio de cartão de crédito utilizando o PagSeguro [13]. Diferentemente da maioria dos outros serviços aqui listados, este será utilizado não apenas por membros da organização, mas também por outras pessoas que não tenham vínculo com a mesma e que queiram contribuir financeiramente. Do lado do administrador da organização, será possível ver quanto foi doado por este meio e qual o perfil dos doadores.
- **Gestão de funcionários:** permitirá que o administrador do sistema faça a gestão dos funcionários da instituição, podendo adicionar ou remover funcionários, armazenar informações sobre os mesmos, designar cargos e até mesmo separar por áreas e times.
- **Gestão de voluntários:** a princípio, este serviço pode parecer ser similar ao de funcionários, mas eles diferem entre si uma vez que os voluntários não têm vínculo empregatício com a organização, muito menos recebem algum salário para trabalhar. Da mesma forma que o serviço acima, na gestão de voluntários o administrador também irá poder adicionar ou remover voluntários, armazenar informações sobre os mesmos (mas não em um nível de detalhes tão grande como os funcionários) e separar por áreas e times.
- **Gerenciamento de projetos:** por meio deste serviço, o administrador pode agendar e visualizar projetos que irão ocorrer, quais as áreas envolvidas, se há receita prevista a entrar, qual será a despesa envolvida, a descrição do projeto, entre outros detalhes pertinentes.
- **Suporte para controle financeiro:** neste serviço, o administrador poderá fazer o controle das finanças da organização, verificando quanto a

organização arrecadou e gerou de receita em um período específico, anotar metas financeiras, indicar as despesas e entradas recorrentes e analisar algumas métricas sobre o gasto de dinheiro da instituição, como por exemplo, ver onde se concentra o seu maior gasto.

- **Suporte para controle de marketing digital:** administrar as redes sociais para impulsionar campanhas de marketing é comum nos dias de hoje, sendo fundamental para que a instituição ganhe mais relevância na internet e, por consequência, consiga mais doações e impactar positivamente mais pessoas. Pensando nisso, esse serviço irá permitir que a organização armazene ideias e fotos a serem compartilhadas nas mídias sociais, além de também poder indicar quais postagens foram feitas e quantas pessoas foram atingidas por cada uma destas publicações. Com esse número em mãos, o serviço irá organizar um ranking indicando para os administradores quais publicações foram mais engajantes.
- **Serviço de relatórios de transparência:** visto que as instituições do terceiro setor não visam o lucro, é fundamental que elas sejam transparentes em relação às suas finanças, pois isso ajuda a conseguir mais doadores e traz uma confiança maior aos voluntários e a outros tipos de colaboradores. O serviço irá permitir que o administrador exporte dados e gráficos a respeito dos gastos e entradas da organização durante o período desejado. Juntamente com o serviço de doações, este também poderá ser acessado por qualquer pessoa mesmo não fazendo parte da organização.
- **Estatísticas de notas fiscais doadas:** uma vez que temos estudos sobre sistemas de microsserviços para ONGs que permitem o registro de notas fiscais e a doação das mesmas para estas instituições [7], resta a este trabalho criar um outro serviço que, por meio do qual, seja capaz de analisar os dados de cada nota fiscal doada e extrair possíveis métricas para que, assim, a organização possa entender melhor o seu público alvo de doadores e elaborar campanhas de arrecadação de doações de forma mais precisa.

A modularização e a separação em microsserviços do sistema a ser entregue permitirá que a organização não seja obrigada a utilizar todos os serviços listados acima, além de poder criar novos sem que os já existentes sejam prejudicados. Todos esses microsserviços juntos irão compor um sistema de gestão completo

para as organizações, tal como faz um *Enterprise Resource Planning* (ERP) - Sistema Integrado de Gestão Empresarial. Assim como demonstrado pelo artigo *Design of Information System Architecture of Garment Enterprises Based on Microservices* [14], implementar microsserviços em sistemas do tipo ERP tem o potencial de reduzir significativamente o tempo necessário para conclusão de processos internos de uma organização, logo a solução implementada trará bons resultados para as ONGs.

O sistema possui, no entanto, algumas limitações já conhecidas:

- **Escalabilidade da infraestrutura**: dado que o resultado final é o código-fonte de uma plataforma, para efetivamente utilizar o sistema, é necessário que a organização, com auxílio de sua equipe técnica, contrate um serviço em nuvem para hospedar o servidor. Sendo assim, a escalabilidade da infraestrutura pode ficar limitada dependendo da robustez do serviço contratado.
- **Adição de registros:** toda a adição de registros - como inclusão de novo funcionário ou voluntário - existente na plataforma é feita manualmente, o que pode ser um incômodo para o administrador caso a instituição tenha um número elevado de novos registros a serem criados. Uma solução para isso seria a automatização desse processo por meio do suporte, na própria interface, para realizar a importação de dados provenientes de planilhas, por exemplo.

3.1.1 Requisitos funcionais e não funcionais

Sabendo quais são os principais microsserviços e como as suas funcionalidades se situam em uma arquitetura em camadas, é possível listar os seus requisitos.
Para os requisitos funcionais, temos:

- **Usuário pode realizar doações:** pertence ao microsserviço de doação. A doação pode ser feita por meio da API do PagSeguro [15] que será integrada e o usuário poderá preencher ou não o formulário de dados pessoas com nome, idade, profissão, motivo da doação, entre outros;
- **Administradores podem visualizar doações feitas:** pertence ao microsserviço de doação, mas na visão da ONG. Toda doação feita pelo

usuário será registrada e os administradores poderão consultar doações realizadas;

- **Administradores podem visualizar perfil dos doadores:** pertence ao microsserviço de doações. Juntamente com as doações realizadas, os administradores poderão ver dados do usuário, tais como nome, idade, profissão, motivo da doação, entre outros, seja de forma consolidada ou individual;
- **Administradores podem adicionar ou remover funcionários do sistema:** pertence ao microsserviço de gestão de funcionários. Ao adicionar um funcionário no sistema, administrador poderá informar dados como nome, CPF e idade do funcionário;
- **Administradores podem designar cargos e separar funcionários em times:** pertence ao microsserviço de gestão de funcionários;
- **Sistema lista os funcionários, mostrando suas informações e times:** pertence ao microsserviço de gestão de funcionários;
- **Administradores podem adicionar ou remover voluntários do sistema:** pertence ao microsserviço de gestão de voluntários. Ao adicionar um voluntário no sistema, administrador poderá informar dados como nome, CPF e idade do voluntário;
- **Administradores podem separar voluntários em áreas:** pertence ao microsserviço de gestão de funcionários;
- **Sistema lista os voluntários, mostrando suas informações e áreas:** pertence ao microsserviço de gestão de voluntários;
- **Administrador pode agendar projetos, com informações detalhadas:** pertence ao microsserviço de gerenciamento de projetos. Eventos realizados pela ONG também poderão ser categorizados como projetos. As informações pertinentes são: data de início e fim, descrição, times e áreas responsáveis, times e áreas participantes, descrição, gastos previstos, gastos realizados, arrecadação prevista, arrecadação alcançada, entre outros;
- **Sistema é capaz de exibir os projetos em ordem cronológica:** pertence ao microsserviço de gerenciamento de projetos. Além de listar os projetos em ordem cronológica, também deverá ser possível visualizar as suas informações, conforme listadas no item anterior. Configuração padrão é de exibir em ordem cronológica, mas também deverá ser possível filtrar e

ordenar os projetos por mão de obra necessária (considerando os times e áreas envolvidos), gastos e arrecadação;

- **Administrador pode verificar as entradas e saídas financeiras da organização:** pertence ao microsserviço de controle financeiro. Organização pode cadastrar as entradas e saídas financeiras e depois consultá-las;
- **Administrador pode definir metas financeiras:** pertence ao microsserviço de controle financeiro. Administrador poderá ver quanto falta para atingir a meta com base nas entradas e saídas recorrentes da organização;
- **Sistema deve ser capaz de consolidar informações financeiras:** pertence ao microsserviço de controle financeiro. As informações relativas às entradas, saídas e metas poderão ser consolidadas e exibidas para o administrador;
- **Administrador pode armazenar ideias e arquivos para postagem em mídias sociais:** pertence ao microsserviço de suporte para controle de marketing digital. Formulário a ser preenchido para armazenamento de ideias e arquivos para postagem terá campos para descrição, título e imagens;
- **Administrador pode buscar conteúdos para postagem já salvos:** pertence ao microsserviço de suporte para controle de marketing digital. Será possível listar ideias registradas e filtrar por data;
- **Administrador pode indicar para o sistema quais publicações foram feitas e qual o nível de engajamento delas:** pertence ao microsserviço de suporte para controle de marketing digital. Administrador poderá editar ideias já salvas para indicar que elas foram publicadas e qual foi o nível de engajamento delas;
- **Sistema lista um ranking de publicações mais engajantes:** pertence ao microsserviço de suporte para controle de marketing digital. Com base nas informações do requisito anterior, deverá ser possível ordenar as publicações já feitas por número de engajamento.
- **Sistema permite que usuários exportem dados e gráficos a respeito das despesas e entradas da organização:** pertence ao microsserviço de relatórios de transparência. A instituição poderá selecionar quais informações ficarão disponíveis ao público, mas a configuração padrão é que todas fiquem. Ficará visível para o usuário quais informações a instituição optou por não disponibilizar;

- **Sistema fornece informações sobre as notas fiscais doadas:** pertence ao microsserviço de estatísticas de notas fiscais doadas. Administrador poderá informar ao sistema as doações recebidas e depois consultá-las;

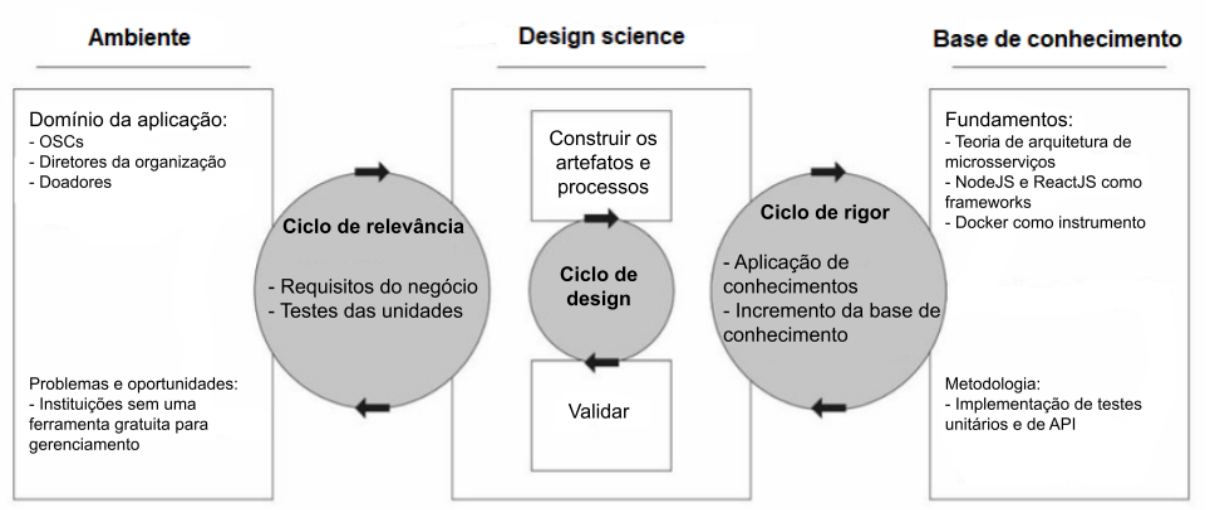
Para os requisitos não funcionais, temos:

- **Segurança:** o sistema deve garantir que informações internas da organização não sejam disponibilizadas para os usuários comuns nas interfaces de doação e de relatórios de transparência. Além disso, na plataforma disponibilizada para os administradores, deverá ter um sistema de login seguro.
- **Escalabilidade:** o sistema deverá permitir que novas funcionalidades sejam adicionadas com facilidade e que as funcionalidades existentes possam ser estendidas sem prejuízos.
- **Independência entre os componentes:** cada um dos 8 microsserviços devem ser independentes entre si e a remoção de um não deve ter impactos no outro, com exceção do microsserviços de relatórios de transparência, que depende do serviço de controle financeiro. Porém, embora haja uma dependência entre os dois, ela se dá apenas em relação aos dados utilizados e consumidos, o que permite que eles não dependam sincronicamente um do outro, bastando apenas que o microsserviço de relatórios de transparência tenha uma cópia dos dados que existem no serviço de controle financeiro.

3.2 Modelo conceitual

O trabalho foi desenvolvido utilizando o framework de Design Science, mencionado em 1.4. Um diagrama ilustrativo pode ser visto na imagem abaixo:

Figura 3.2.1 - Diagrama Design Science



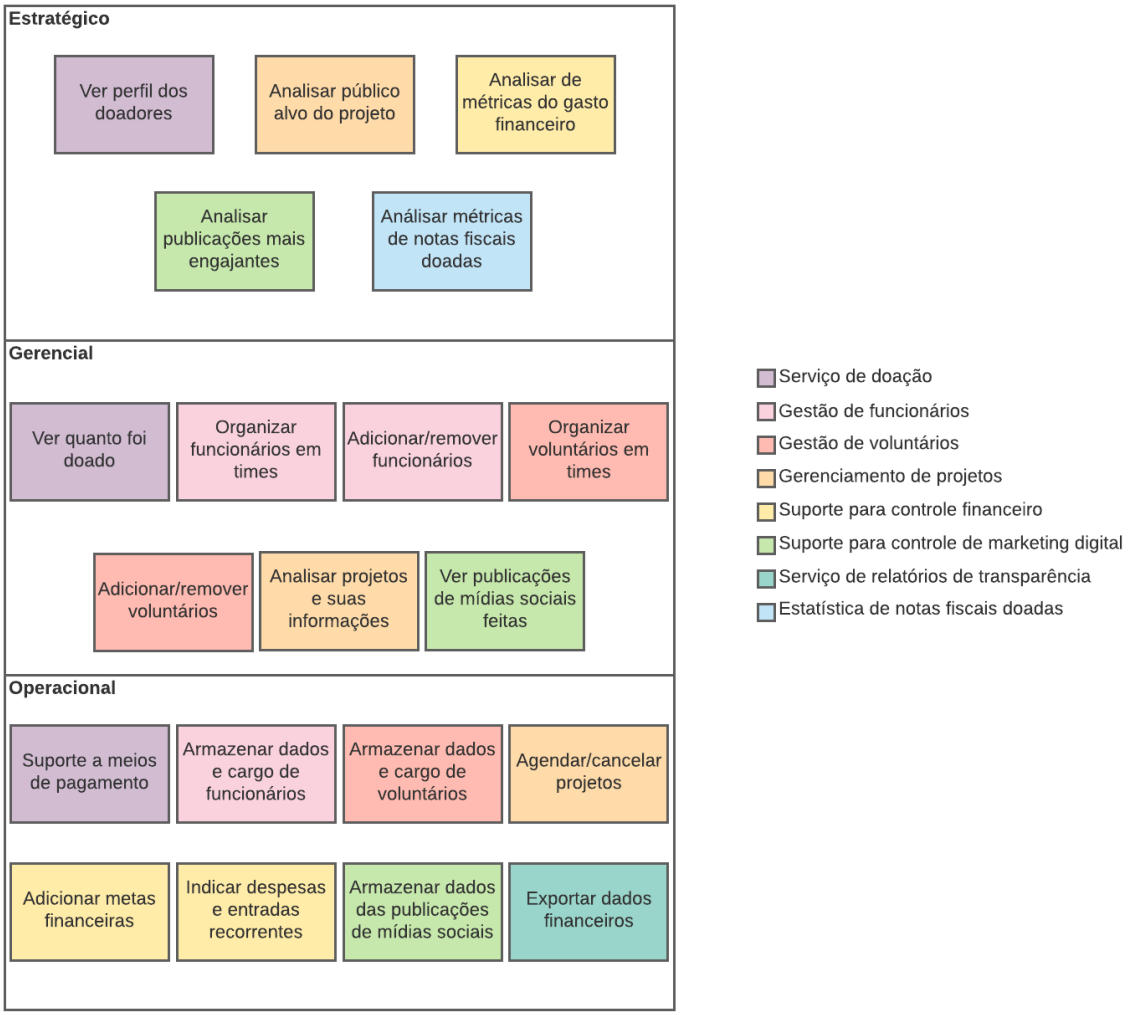
Fonte: Produção própria

3.3 Produto principal

3.3.1 Estrutura do produto

As funcionalidades de cada microsserviço podem ser organizadas em um modelo de camadas, separadas por nível operacional, gerencial e estratégico, formando a arquitetura computacional, conforme na imagem abaixo:

Figura 3.3.1.1 - Arquitetura computacional em camadas



Fonte: Produção própria

3.3.2 Ciclo de operação

Considerando o contexto da instituição do terceiro setor que irá utilizar a ferramenta, o produto desenvolvido se encaixa como uma ferramenta auxiliar para a gestão de seus processos internos. O código-fonte do software desenvolvido está aberto e disponível no GitHub da autora [16]. A ideia é que a própria organização baixe esse conteúdo e coloque para rodar cada microsserviço em um servidor na nuvem, como por exemplo, uma instância de da Amazon Elastic Compute Cloud [17] (também conhecida como EC2), onde é possível, em poucos minutos, colocar um site no ar. O ideal será, portanto, que a OSC tenha uma área com profissionais de tecnologia que possam realizar essa manutenção.

Além disso, caso a instituição deseje utilizar o microsserviço de doações, é necessário que esta tenha uma conta no PagSeguro [15] dado que este foi o serviço de pagamentos escolhido para ser integrado na solução.

Por fim, vale ressaltar que, como a instituição tem em mãos o código-fonte, é possível fazer qualquer alteração ou customização desejada diretamente no conteúdo baixado do Github ou, ainda, caso seja uma extensão mais genérica, incluí-la no repositório original em que está o código, pois trata-se de um produto open source.

3.3.3 Desenvolvimento do produto

Para o desenvolvimento do produto, foi adotada a metodologia Scrum e utilizado o conceito de Sprints, conforme mencionado anteriormente na seção 1.4, sendo que cada microsserviço foi implementado em uma Sprint de uma forma adaptada, pois não foi utilizado a definição de tempo de sprints. Para o bom andamento do projeto, se fez mais adequado ter o tempo das sprints de forma flexível, pois alguns serviços são mais complexos do que outros, porém não tanto a ponto de quebrá-los em 2 sprints separadas.

Dado que são, no total, 12 microsserviços - 8 para o backend das principais funcionalidades, 1 para a interface do administrador, 1 para a interface do usuário, 1 para gerenciar a autenticação e os administradores e 1 para configurações gerais - então foram necessárias 12 sprints, que foram executadas de abril a julho, conforme a tabela abaixo:

Tabela 3.3.3.1 - Cronograma de Sprints

		Mês			
		Abril	**Maio**	**Junho**	**Julho**
Microsserviço	**Doações**				
	Autenticação				
	Funcionários				
	Voluntários				
	Projetos				
	Financeiro				
	Marketing				

	Relatórios				
	Notas fiscais				
	Configurações				
	Interface Administrador				
	Interface Usuário				

Fonte: Produção própria

A lista dos *commits* realizados no desenvolvimento do projeto pode ser encontrada no repositório do Github [15]. Um *commit* representa a unidade mínima do desenvolvimento de uma funcionalidade. É importante quebrar as funcionalidades nessas unidades para que seja mais fácil retornar o código para alguma versão específica e também até para se ter uma documentação mais completa do projeto.

3.4 Implementação

3.4.1 Tecnologias utilizadas

Pensando nas tecnologias a serem utilizadas no projeto, foi levado em conta a adequabilidade de cada uma ao conceito de microsserviços e também a popularidade das mesmas, pois é necessário que o produto final seja o mais acessível possível. Assim, para a construção do back-end de todos os microsserviços, a tecnologia ideal para o uso é o JavaScript com a ferramenta Node.js [18], pois, segundo a sua própria documentação, permite a construção de aplicações escaláveis. Além disso, o Node.js é a ferramenta de desenvolvimento back-end web mais utilizada quando levamos em conta números como seguidores no Stackshare [19], número de questões no StackOverflow [20] e número de estrelas no Github [21], mostrando-se muito acessível a diversos desenvolvedores. Por fim, os pontos positivos de se utilizar Node.js para uma aplicação de microsserviços superam os pontos negativos [22] - inclusive, os contras não impactam tão fortemente o sistema a ser aqui desenvolvido em específico, como por exemplo, problemas de performance em tarefas que exigem alto nível de processamento - o que não há no nosso sistema.

Tabela 3.4.1.1 - Principais pontos positivos e negativos do Node.js.

Backend em Node.js	
Pontos Positivos	**Pontos Negativos**
Permite a construção de sistemas escaláveis	Performance pode ser um gargalo em tarefas com alto nível de processamento
+77k de seguidores no Stackshare (contra +18k do Django e +9k do Rails, outras duas ferramentas populares)	Linguagem dependente de callbacks, o que pode reduzir a qualidade do código
+374k de questões no StackOverflow (contra +259k do Django e +324k do Rails)	
+77k estrelas no Github (contra +55k do Django e +47k do Rails)	
Ecossistema rico, com mais de 800k de bibliotecas [22]	

Fonte: Produção própria

Quanto ao desenvolvimento da interface com o usuário, o front-end, foi decidido que também seria utilizado o JavaScript, devido a sua popularidade, mas com o framework React.js, que é o mais popular para a linguagem segundo dados de plataformas da comunidade de desenvolvedores [23]. O React é um framework simples de utilizar e que permite a facilidade de criar componentes de UI com agilidade, o que é perfeito para o desenvolvimento do sistema.

Já para o banco de dados, foi feita uma análise um pouco diferente, uma vez que precisamos de soluções diferentes para cada microsserviço. Para os microsserviços Serviço de doação, Serviço de relatórios de transparência e Estatísticas de notas fiscais doadas o mais adequado será a utilização de um banco de dados não relacional, pois não é necessário criar relacionamento entre os elementos utilizados por esses serviços e também porque o modelo não relacional permite mais escalabilidade e armazenamento massivo de dados [24], o que é ótimo uma vez que o serviço de Estatísticas de notas fiscais doadas, por exemplo, tende a crescer muito em termos de quantidade de registros no banco de dados. Entre os bancos de dados não relacionais existentes, foi decidido que seria utilizado o MongoDB [25], pois possui uma boa integração com o Node.js e é open source.

Para os demais serviços - Gestão de funcionários, Gestão de voluntários, Gerenciamento de projetos, Suporte para controle financeiro e Suporte para controle de marketing digital - será necessário um banco de dados relacional, pois é preciso construir relacionamentos entre entidades, além de que eles lidarão com um uso de dados que não crescerá tão rápido como os outros que vão utilizar MongoDB. Entre os bancos relacionais disponíveis, foi decidido que seria utilizado o Postgres [26], pois é open source, facilmente de ser expandido [27], acessível e funciona bem com Node.js, uma vez que o PostgreSQL é muito parecido com os padrões do SQL.

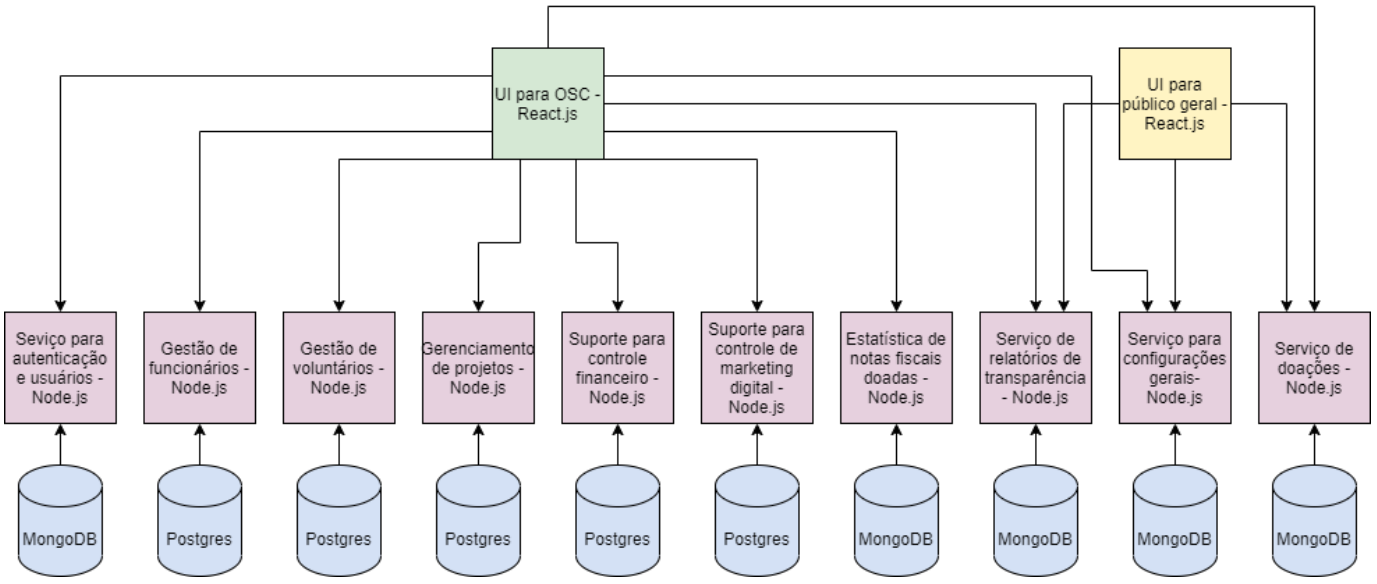
Tabela 3.2 - Pontos positivos destacados de cada banco de dados.

MongoDB	Postgres
● Permite maior escalabilidade ● Lida facilmente com volume massivo de dados ● Integra bem com Node.js	● Permite relacionamento entre as entidades ● Pode ser facilmente expandido ● Tem suporte ao Node.js ● Linguagem similar ao SQL e seus padrões

Fonte: Produção própria

Assim definidas as tecnologias e serviços, temos o seguinte diagrama na figura abaixo:

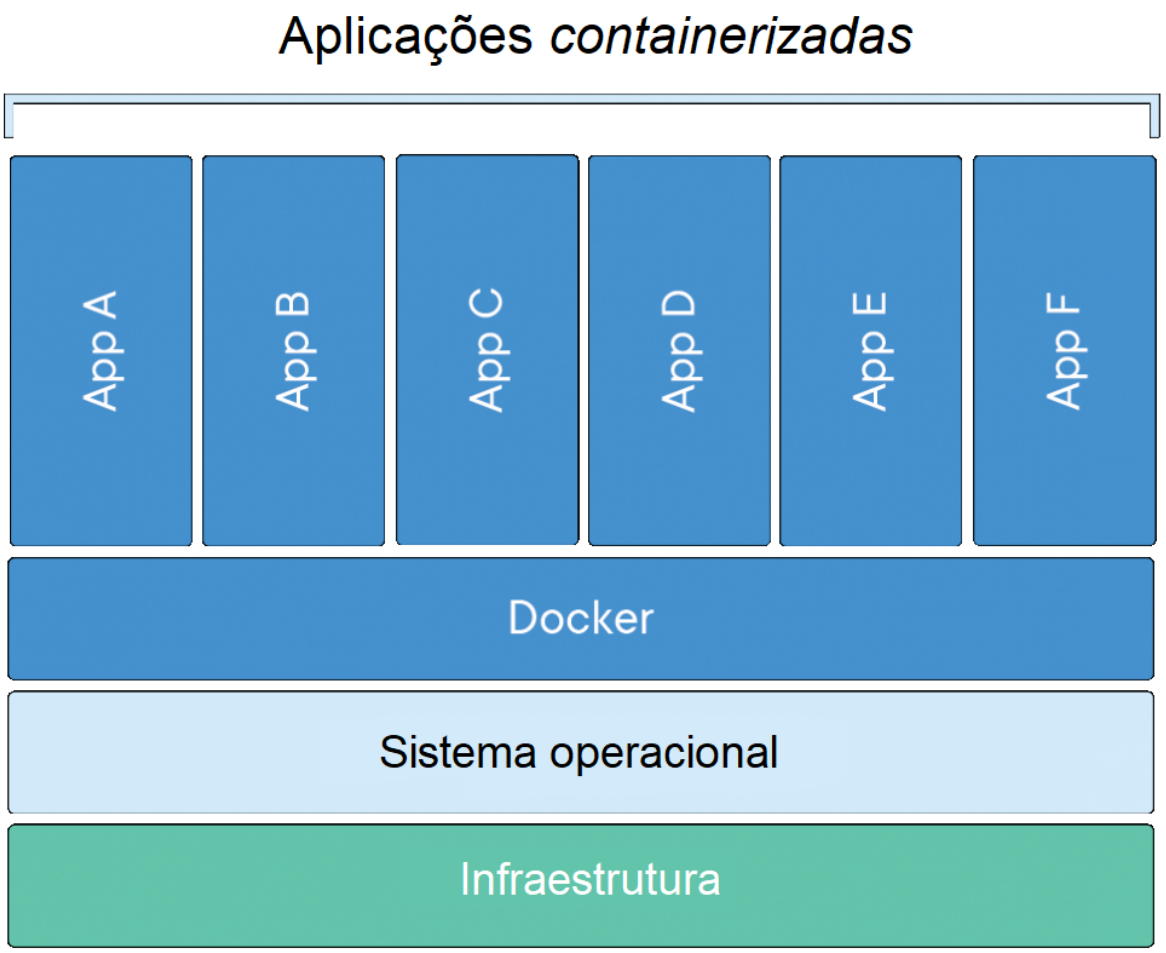
Figura 3.1 - Modelo de implementação



Fonte: Produção Própria

Cada microsserviço do trabalho será um módulo e ficará armazenado em um *container*. Para criar esses containers, será utilizado o Docker [28], que permite o empacotamento de serviços junto com suas dependências. Feito este empacotamento, é possível instalar a aplicação de forma rápida e segura em qualquer máquina que tenha suporte ao Docker. O Docker é capaz de criar um ambiente isolado para cada serviço, sem nenhum tipo de conflito entre eles, de forma que todos rodem na mesma máquina. Um esquema de seu funcionamento com os aplicativos containerizados é mostrado na figura abaixo:

Figura 3.2 - Serviços no Docker

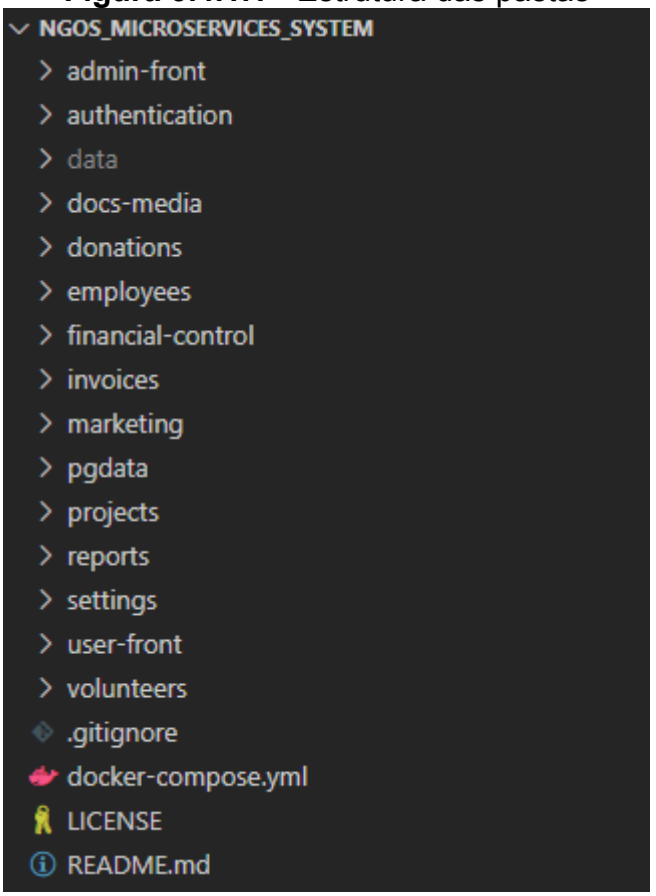


Fonte: Traduzida de Docker [28]

3.4.2 Descrição tecnológica do produto

A estrutura geral da pasta do projeto ficou da seguinte forma:

Figura 3.4.1.1 - Estrutura das pastas



Fonte: Produção própria

Dessa forma, cada serviço possui a sua própria pasta. Durante o desenvolvimento da aplicação, foi utilizado o Docker Compose [29], ferramenta para rodar múltiplos containers Docker, representado pelo arquivo “docker-compose.yml”, que pode ser visto na Figura 3.4.1.2. Essa ferramenta, além de instanciar todos os containers solicitados, também pode ser utilizada para criar serviços de banco de dados, o que é necessário para o produto, visto que ele utiliza o Postgres e o MongoDB como tecnologias para armazenamento de dados. Utilizando um terminal, é possível entrar na pasta do projeto e, digitando apenas “docker-compose up”, todos os containers são inicializados, conforme mostra figura abaixo:

Figura 3.4.1.2 - Inicialização dos containers

```
D:\wsl\tcc\ngos_microservices_system>docker-compose up
Starting NGO_ADMIN_FRONT        ... done
Starting NGO_INVOICES           ...
Starting NGO_REPORTS            ...
Starting NGO_USER_FRONT         ... done
Starting NGO_FINANCIAL_CONTROL  ... done
Starting NGO_VOLUNTEERS         ... done
Starting NGO_PROJECTS           ... done
Starting NGO_EMPLOYEES          ... done
Starting NGO_MARKETING          ... done
Starting NGO_SETTINGS           ... done
Starting NGO_DONATIONS          ... done
Starting NGO_AUTHENTICATION     ... done
Starting NGO_INVOICES           ... done
Starting NGO_REPORTS            ... done
```

Fonte: Produção própria

Para tudo funcionar conforme previsto, é necessário que cada uma das pastas dos serviços possua:

- **Um arquivo "Dockerfile":** responsável por descrever o container do serviço, indicando quais comandos executar para sua inicialização e em qual porta web o servidor estará disponível.

Figura 3.4.1.3 - Exemplo de Dockerfile

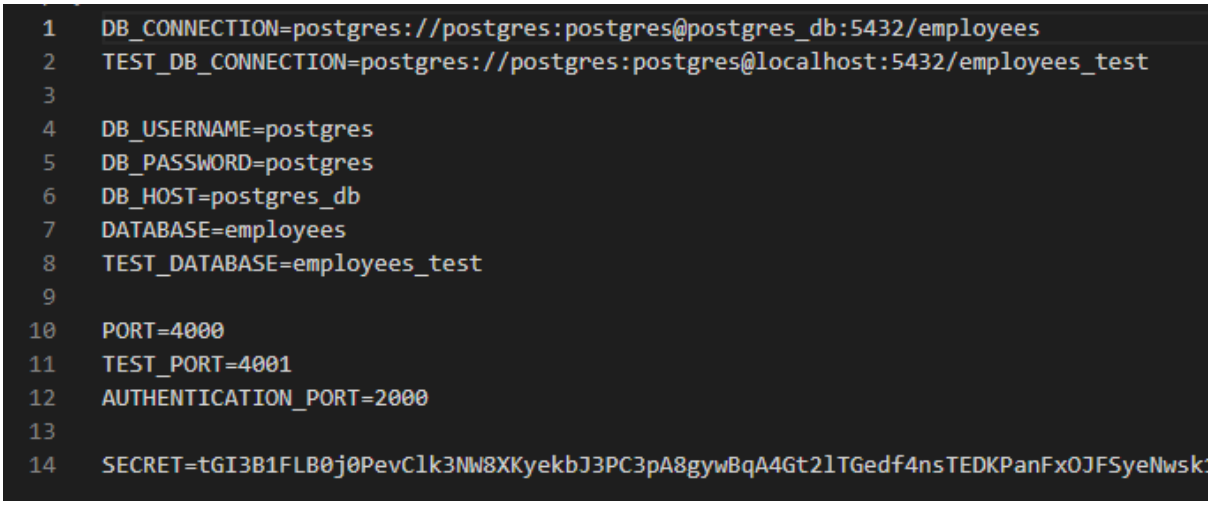
```
1   FROM node:alpine
2
3   WORKDIR /authentication
4
5   COPY package.json .
6
7   RUN npm install --quiet
8
9   RUN npm install nodemon -g --quiet
10
11  COPY . .
12
13  EXPOSE 2000
14
15  CMD nodemon -L --watch . index.js
```

Fonte: produção própria

- **Um arquivo ".env"**: necessário para indicar as variáveis de ambiente específicas de cada serviço. Para construção desse arquivo, que fica escondido da ferramenta de versionamento de código por motivos de

segurança, deve-se seguir o ".sample-env" que existe em cada uma das também.

Figura 3.4.1.4 - Exemplo de .env



Fonte: produção própria

- **A implementação em si do serviço**
- **Um arquivo README.md:** necessário para documentar cada um dos serviços.

3.4.2.1 Serviço de doações

Este serviço permite a realização de doações com cartão de crédito por meio da API 4.0 do PagSeguro [15]. Em sua construção foi utilizado o Node.js com o Express e MongoDB e recebe requisições tanto da interface do usuário quanto da interface do administrador. Cada doação realizada é registrada no serviço de controle de controle financeiro como uma nova transação de entrada. Com o serviço de doações, é possível:

- Realizar uma doação com cartão de crédito
- Listar todas as doações realizadas
- Listar todos os doadores

3.4.2.2 Serviço de autenticação

Este serviço é consumido por todos os outros do back-end para a validação do token de acesso (necessário em todos os *endpoints* de uso do administrador) e também pelo front-end do administrador para realizar o login na plataforma e

adicionar novos usuários. Ele foi construído utilizando Node.js com o Express e MongoDB. Com ele, é possível:

- Cadastrar um usuário pela primeira vez
- Realizar o login
- Redefinir senha
- Registrar um novo usuário
- Verificar se é possível se auto cadastrar

Para simplificar a implementação e garantir a segurança, o fluxo implementado para este serviço é: após o auto cadastro do primeiro usuário, não é mais possível se auto cadastrar, sendo necessário que um administrador que tenha acesso à plataforma registre os novos usuários que possam aparecer.

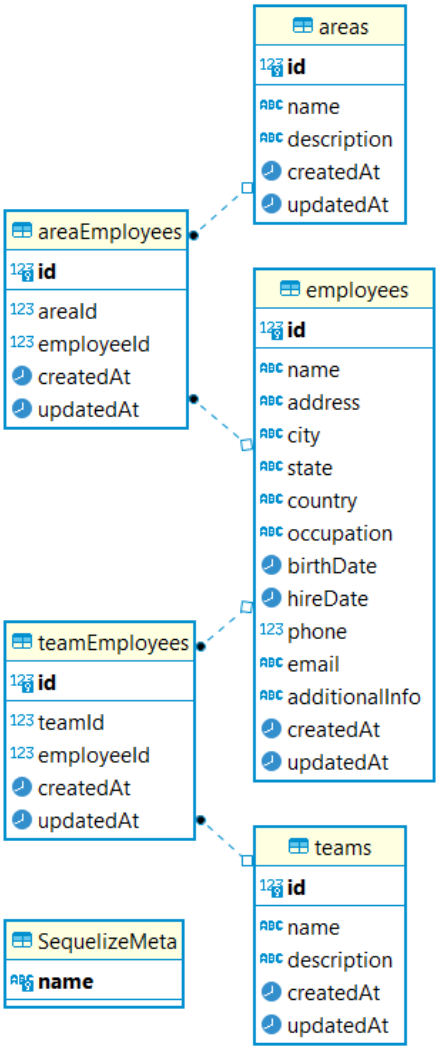
3.4.2.3 Serviço de funcionários

Serviço utilizado para gerenciar os usuários. Foi construído com Node.js com o Express e Postgres. Com ele, é possível:

- Listar todos os funcionários e filtrar
- Acessar dados de um único funcionário
- Criar novo funcionário
- Editar informações de um funcionário
- Remover um funcionário
- Listar todos as áreas e filtrar
- Acessar dados de uma única área
- Criar nova área
- Editar informações de uma área
- Remover uma área
- Listar todos os times e filtrar
- Acessar dados de um único time
- Criar novo time
- Editar informações de um time
- Remover um time

Ele possui a seguinte relação entre as tabelas de seu banco de dados:

Figura 3.4.2.3.1 - Diagrama do banco de funcionários



Fonte: Produção própria

3.4.2.4 Serviço de voluntários

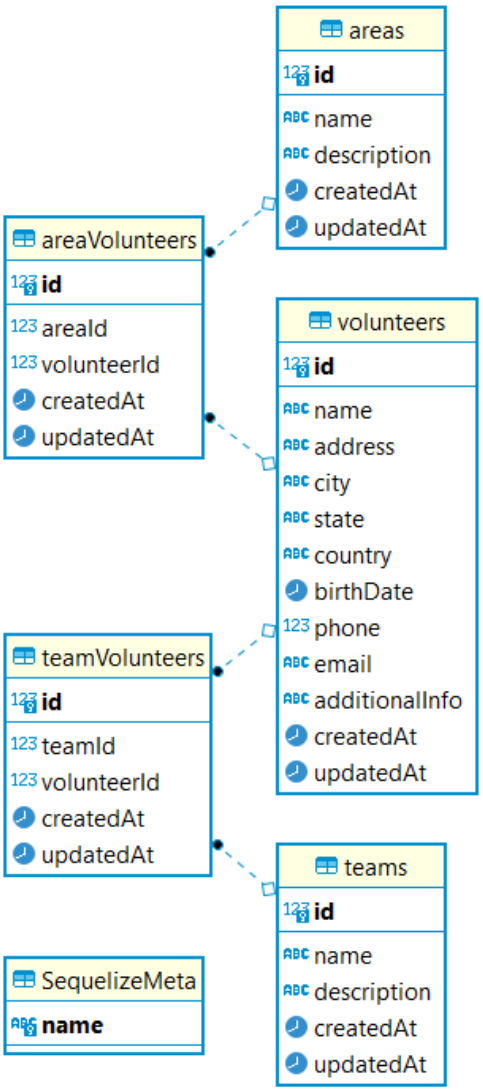
Serviço utilizado para gerenciar os voluntários. Foi construído com Node.js com o Express e Postgres. Com ele, é possível:

- Listar todos os voluntários e filtrar
- Acessar dados de um único voluntário
- Criar novo voluntário
- Editar informações de um voluntário
- Remover um voluntário
- Listar todos as áreas e filtrar
- Acessar dados de uma única área

- Criar nova área
- Editar informações de uma área
- Remover uma área
- Listar todos os times e filtrar
- Acessar dados de um único time
- Criar novo time
- Editar informações de um time
- Remover um time

Ele possui a seguinte relação entre as tabelas de seu banco de dados:

Figura 3.4.2.4.1 - Diagrama do banco de voluntários



Fonte: Produção própria

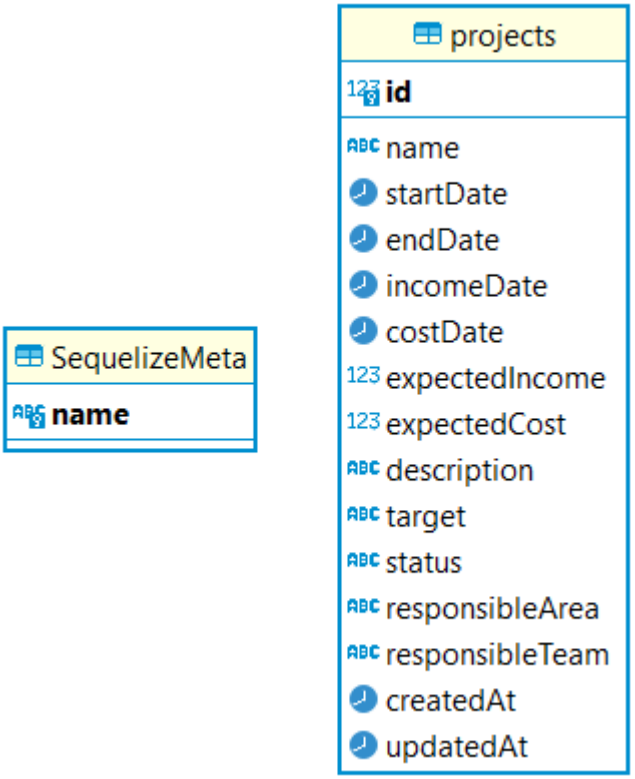
3.4.2.5 Serviço de projetos

Serviço que permite o gerenciamento de projetos - ou eventos - a serem realizados pela organização. Foi construído utilizando Node.js com o Express e Postgres. Com ele, é possível:

- Listar todos os projetos e filtrar
- Acessar dados de um único projeto
- Criar novo projeto
- Editar informações de um projeto
- Remover um projeto
- Consultar o total de entradas esperadas considerando os projetos cadastrados e os filtros aplicados
- Consultar o total de despesas esperadas considerando os projetos cadastrados e os filtros aplicados

Ele possui a seguinte relação entre as tabelas de seu banco de dados:

Figura 3.4.2.5.1 - Diagrama do banco de projetos



Fonte: Produção própria

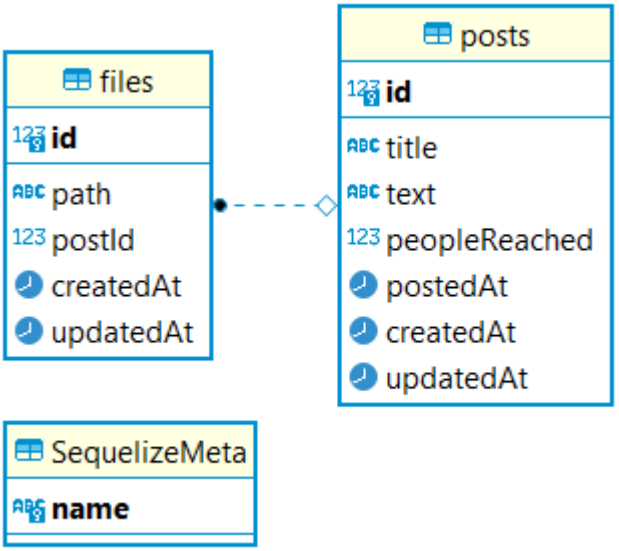
3.4.2.6 Serviço de marketing digital

Serviço para que o administrador possa registrar rascunhos e postagens realizadas em redes sociais. Ele foi construído usando Node.js com o Express e Postgres. Dentro de sua pasta, é necessária a criação de uma outra chamada “public-files” em que são armazenadas as imagens submetidas para cada postagem. Com esse serviço, é possível:

- Listar todas as postagens e filtrar
- Acessar dados de uma única postagem
- Criar nova postagem
- Editar informações de uma postagem
- Remover uma postagem
- Realizar o upload de uma foto
- Remover uma foto

Ele possui a seguinte relação entre as tabelas de seu banco de dados:

Figura 3.4.2.6.1 - Diagrama do banco de marketing digital



Fonte: Produção própria

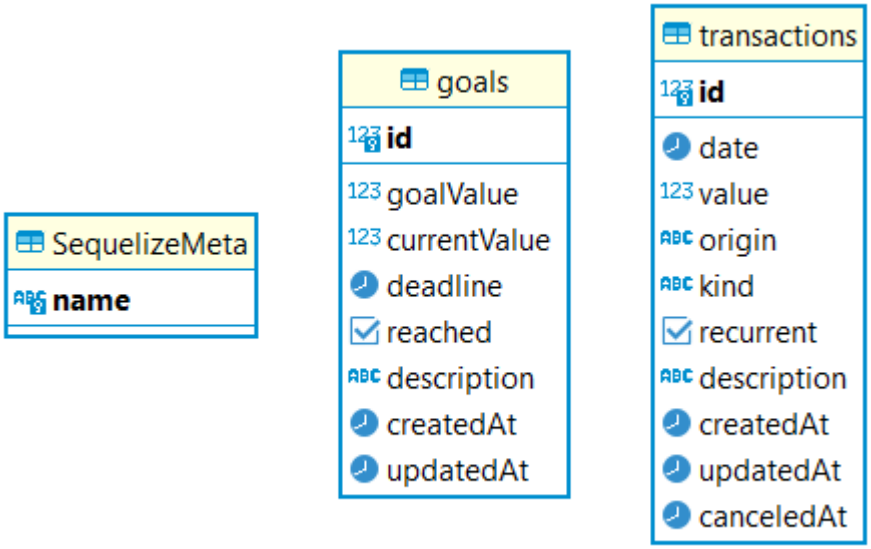
3.4.2.7 Serviço de controle financeiro

Serviço para que o administrador possa cadastrar entradas e saídas, além de metas financeiras. Ele também é usado no serviço do back-end de gerar relatórios de transparência, que são consumidos pelo usuário. Foi construído usando Node.js com o Express e Postgres. Com ele, é possível:

- Listar todas as transações e filtrar
- Acessar dados de uma única transação
- Criar nova transação
- Editar informações de uma transação
- Remover uma transação
- Listar todas as metas e filtrar por atingidas
- Acessar dados de uma única meta
- Criar nova meta
- Editar informações de uma meta
- Remover uma meta
- Visualizar as transações agrupadas por origem
- Visualizar o saldo atual, que é calculado utilizando todas as transações registradas
- Visualizar o valor recorrente mensal de entrada ou saída, calculado com base nas transações recorrentes registradas
- Visualizar todas as origens de entrada ou saída das transações

Ele possui a seguinte relação entre as tabelas de seu banco de dados:

Figura 3.4.2.7.1 - Diagrama do banco de controle financeiro



Fonte: Produção própria

3.4.2.8 Serviço de relatórios

Serviço que permite a visualização dos dados de transparência pelo usuário na forma de planilha ou de gráfico, caso o administrador tenha permitido a visualização. Ele foi construído utilizando Node.js com o Express e MongoDB. Com ele, é possível:

- Visualizar a configuração de visualização definida pelo administrador
- Alterar a configuração de visualização
- Ver os dados consolidados para a construção de um gráfico - as transações são agrupadas por origem
- Ver os dados em uma planilha

3.4.2.9 Serviço de notas fiscais

Serviço para a realização de cadastro de doações de notas fiscais. Ele foi construído utilizando Node.js com o Express e MongoDB. Com ele, é possível:

- Listar todas os cadastros e filtrar
- Acessar dados de um único cadastro
- Criar novo cadastro
- Editar informações de um cadastro
- Remover um cadastro

3.4.2.10 Serviço de configurações

Serviço para a definição de algumas configurações gerais da plataforma tais como nome da OSC e ativação e desativação dos serviços a serem utilizados. Foi construído utilizando Node.js com o Express e MongoDB. Com ele, é possível:

- Ver o nome definido para a OSC
- Alterar o nome definido
- Ver os serviços habilitados
- Alterar os serviços habilitados

3.4.2.11 Serviço de interface do administrador

Serviço responsável por renderizar a interface do administrador. Ele consome todos os serviços do back-end mencionados acima. Foi criado utilizando React.js

3.4.2.12 Serviço de interface do usuário

Serviço responsável por renderizar a interface do usuário, onde é possível ver a página institucional, realizar doações e verificar relatórios de transparência. Consome os seguintes serviços do back-end:

- Doações
- Relatórios
- Configurações

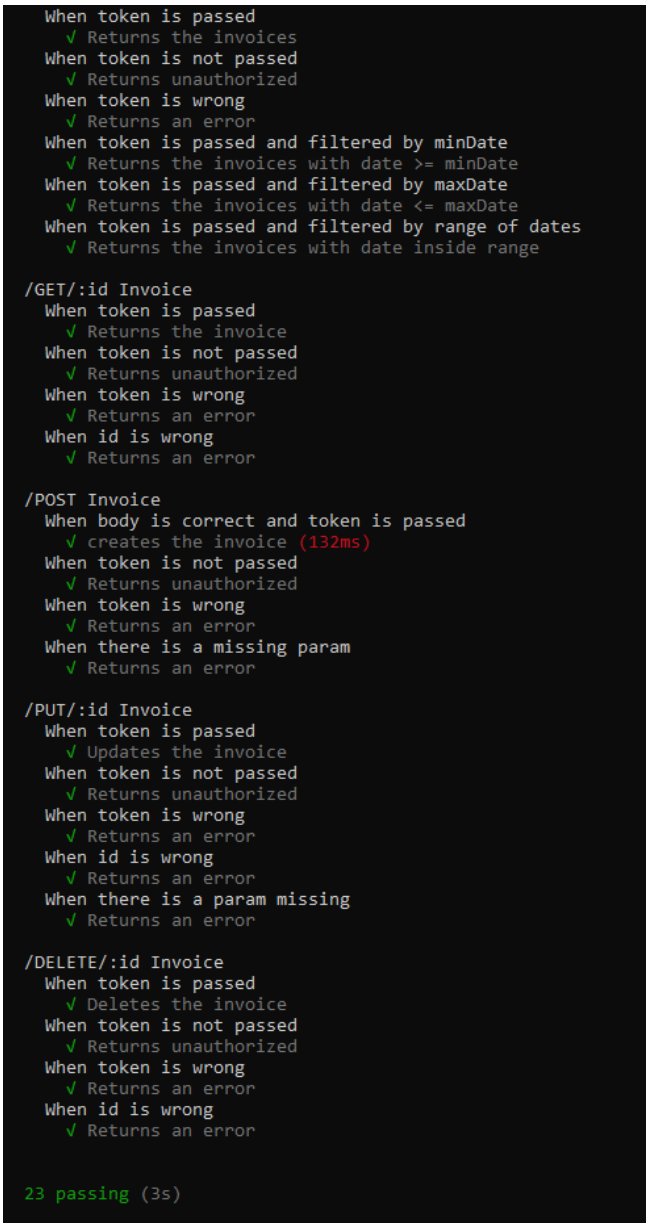
3.4.3 Testes

Para garantir o funcionamento adequado do sistema, todos os 10 serviços do back-end possuem tanto testes de integração quanto testes unitários implementados junto com suas funcionalidades. Cada uma das pastas, possui dentro uma pasta chamada “tests”, onde estão os testes automatizados criados para testar.

Os testes foram construídos utilizando as bibliotecas Mocha [30] e Chai [31] do Javascript, pois são dois frameworks que permitem uma forma bem rápida e simplificada de construir testes e são mais voltadas para aplicações web, que é o caso do produto realizado.

Para rodar os testes, é necessário ter os containers rodando e, então, basta entrar, utilizando um terminal, em cada uma das pastas e digitar “npm test”. O output no terminal deve ser conforme o exemplo da imagem abaixo:

Figura 3.4.3.1 - Execução de teste automatizado



Fonte: Produção própria

3.6 Resultados

Como resultado final do projeto, há um sistema completo para que uma instituição do terceiro setor possa usufruir gratuitamente.

Na interface do usuário, temos:

Figura 3.6.1 - Tela inicial da interface do usuário

Fonte: Produção própria

Figura 3.6.2 - Tela de doações da interface do usuário

Fonte: Produção própria

Figura 3.6.3 - Tela de doações da interface do usuário (cont.)

Fonte: Produção própria

Figura 3.6.4 - Tela de transparência da interface do usuário

Fonte: Produção própria

Figura 3.6.5 - Planilha exportada da tela de transparência

	A	B	C	D	E	F	G
1	Data	Valor (R$)	Origem	Tipo	É recorrente (mensal)?	Recorrência finalizada em	Descrição
2	2021-08-20	10	Doação	Entrada	Não	-	Doação feita pelo site
3	2021-07-17	12	Doação	Entrada	Não	-	Doação feita pelo site
4	2021-07-17	12	Doação	Entrada	Não	-	Doação feita pelo site
5	2021-07-17	12	Doação	Entrada	Não	-	Doação feita pelo site
6	2021-07-17	12	Doação	Entrada	Não	-	Doação feita pelo site
7	2021-07-17	12	Doação	Entrada	Não	-	Doação feita pelo site
8	2021-07-17	122	Doação	Entrada	Não	-	Doação feita pelo site
9	2021-07-17	122	Doação	Entrada	Não	-	Doação feita pelo site
10	2021-07-17	10	Doação	Entrada	Não	-	Doação feita pelo site
11	2021-07-17	12	Doação	Entrada	Não	-	Doação feita pelo site
12	2021-07-07	123	111	Entrada	Não	-	
13	2021-07-01	90	Contas	Saída	Sim	-	Despesas mensais
14	2021-05-29	10	Doação	Entrada	Não	-	Doação feita pelo site
15	2021-05-29	5	Doação	Entrada	Não	-	Doação feita pelo site
16	2021-04-01	20	Outras	Entrada	Sim	2021-07-01	Doação recorrente
17	2021-02-04	80	Doação recorrente	Entrada	Sim	-	
18							
19							

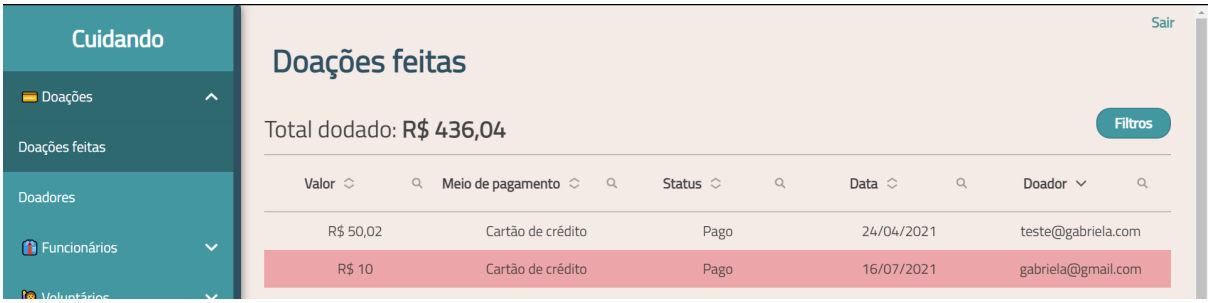
Fonte: Produção própria

Para a interface do administrador, temos:

Figura 3.6.6 - Tela inicial da interface do administrador

Fonte: Produção própria

Figura 3.6.7 - Tela de doações da interface do administrador



Fonte: Produção própria

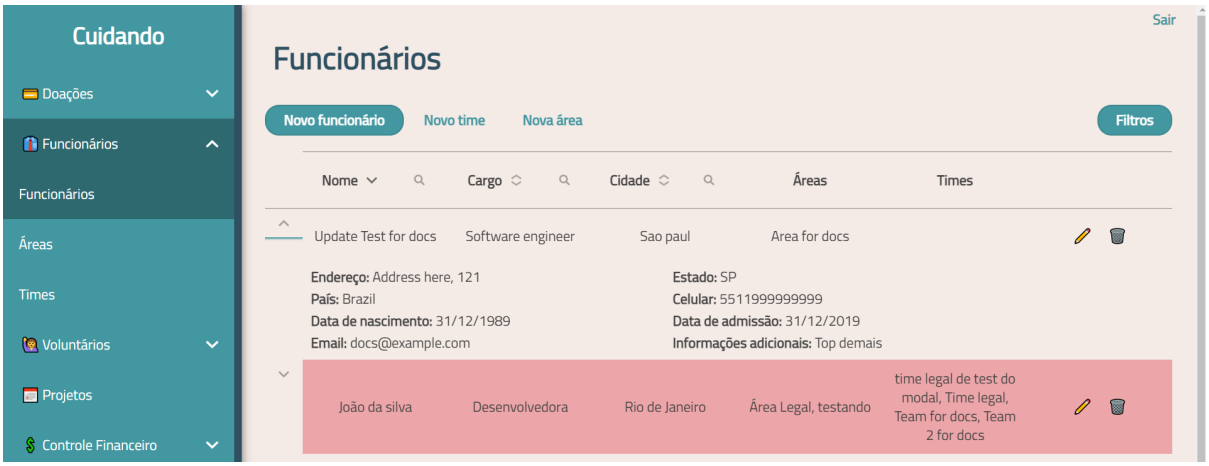
Figura 3.6.8 - Filtros de doações da interface do administrador

Fonte: Produção própria

Figura 3.6.9 - Tela de doadores da interface do administrador

Fonte: Produção própria

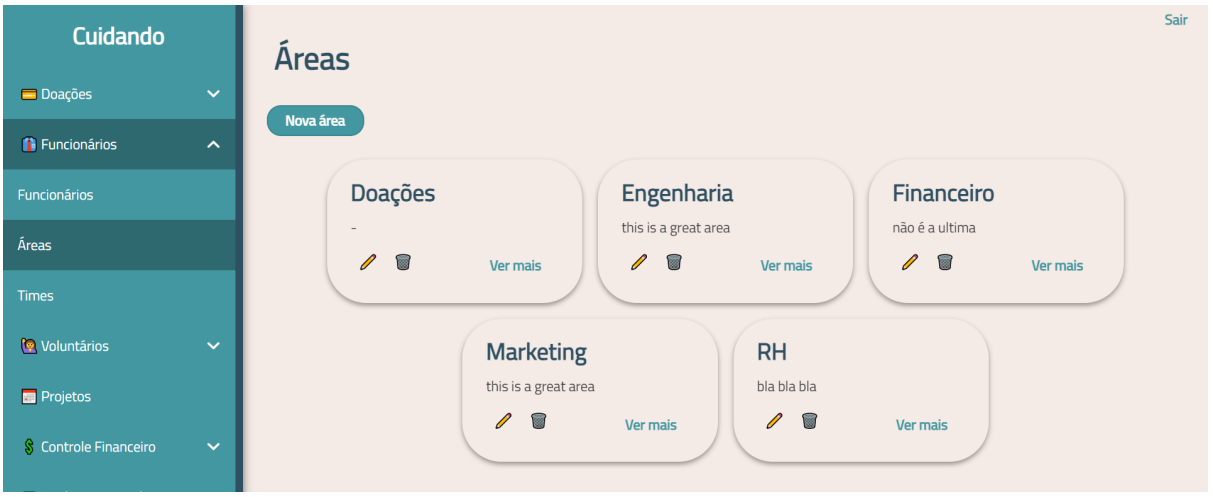
Figura 3.6.10 - Tela de funcionários da interface do administrador



Fonte: Produção própria

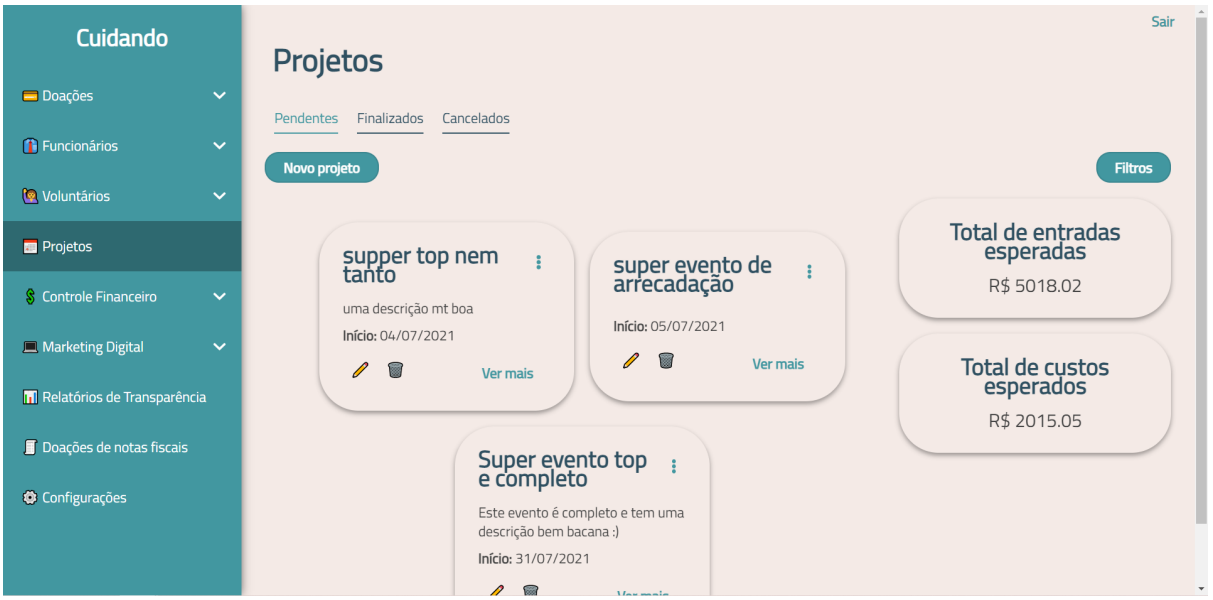
A tela de voluntários é muito similar à de funcionários. Para não haver uma informação repetitiva na monografia, somente a figura 3.6.10 acima será mantida. O mesmo ocorre na figura 3.6.11 abaixo, mostrando a tela de áreas, que por sua vez é muito similar a de times.

Figura 3.6.11 - Tela de áreas de funcionários da interface do administrador



Fonte: Produção própria

Figura 3.6.12 - Tela de projetos da interface do administrador

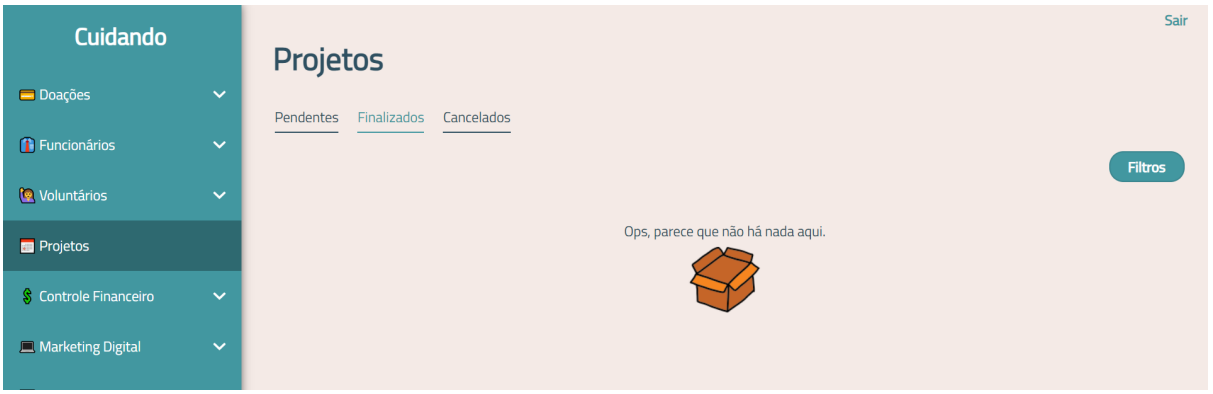


Fonte: Produção própria

Figura 3.6.13 - Descrição de um projeto

Fonte: Produção própria

Figura 3.6.14 - Tela de projetos finalizados da interface do administrador



Fonte: Produção própria

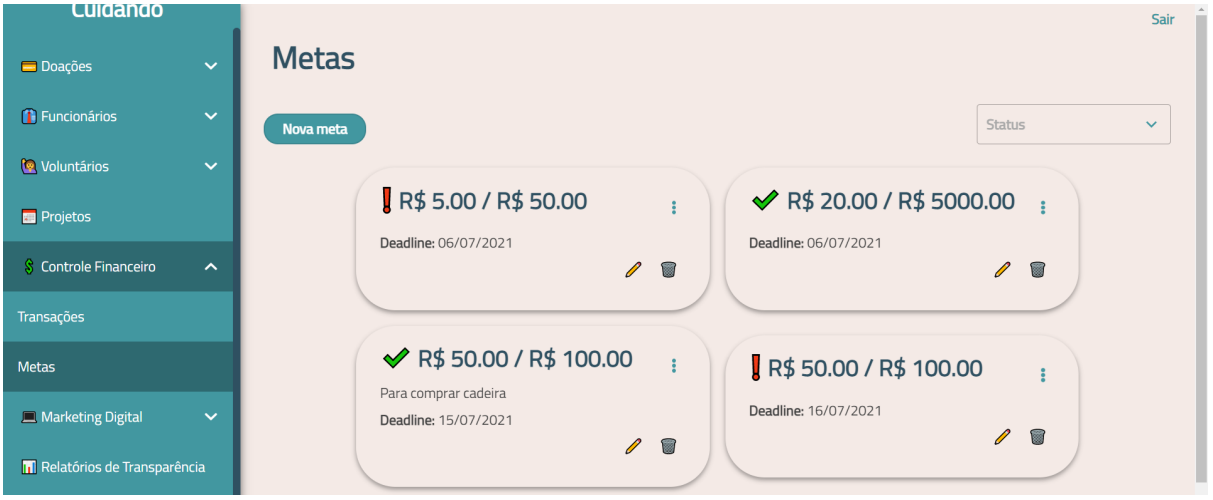
Figura 3.6.15 - Tela de transações da interface do administrador

Fonte: Produção própria

Figura 3.6.16 - Tela de transações agrupadas da interface do administrador

Fonte: Produção própria

Figura 3.6.17 - Tela de metas da interface do administrador



Fonte: Produção própria

Figura 3.6.18 - Tela de rascunhos de postagens da interface do administrador

Fonte: Produção própria

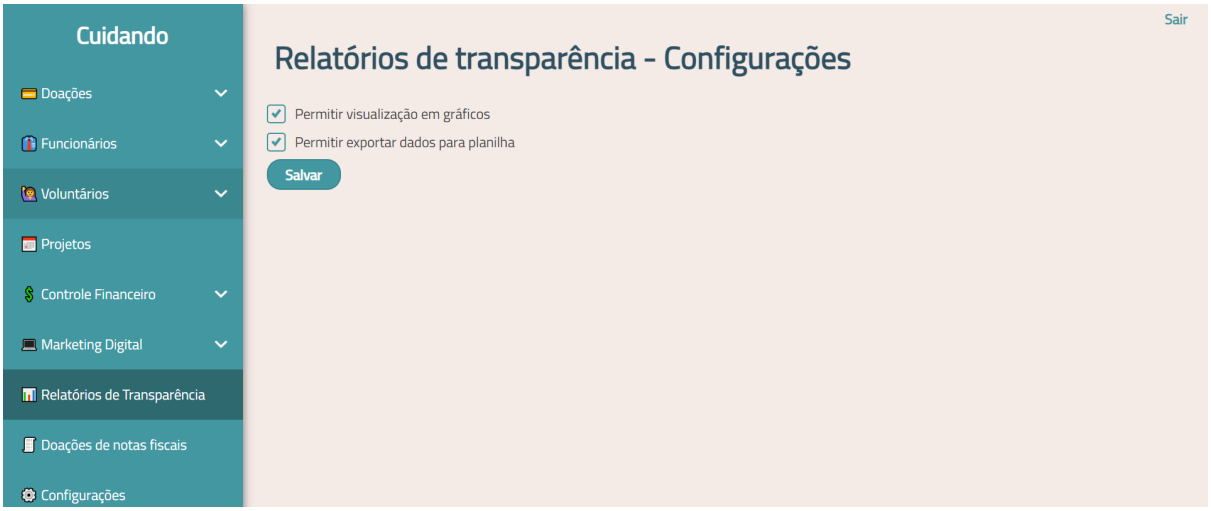
Figura 3.6.19 - Tela de edição de postagem da interface do administrador

Fonte: Produção própria

Figura 3.6.20 - Tela de publicações postadas da interface do administrador

Fonte: Produção própria

Figura 3.6.21 - Tela de configuração dos relatórios da interface do administrador

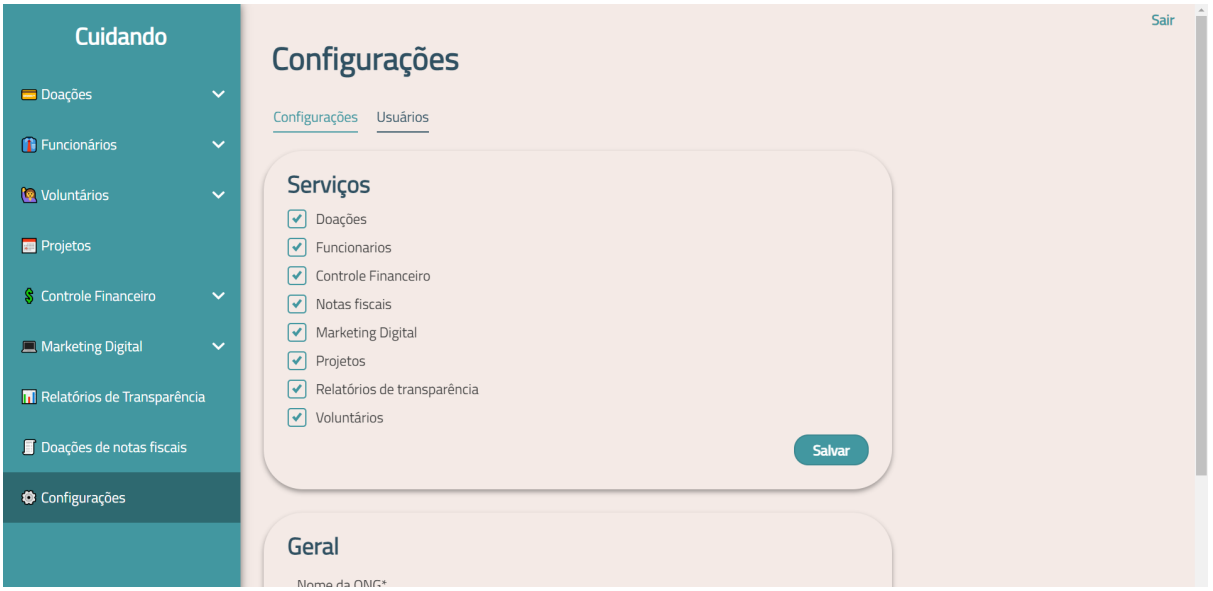


Fonte: Produção própria

Figura 3.6.22 - Tela de notas fiscais doadas da interface do administrador

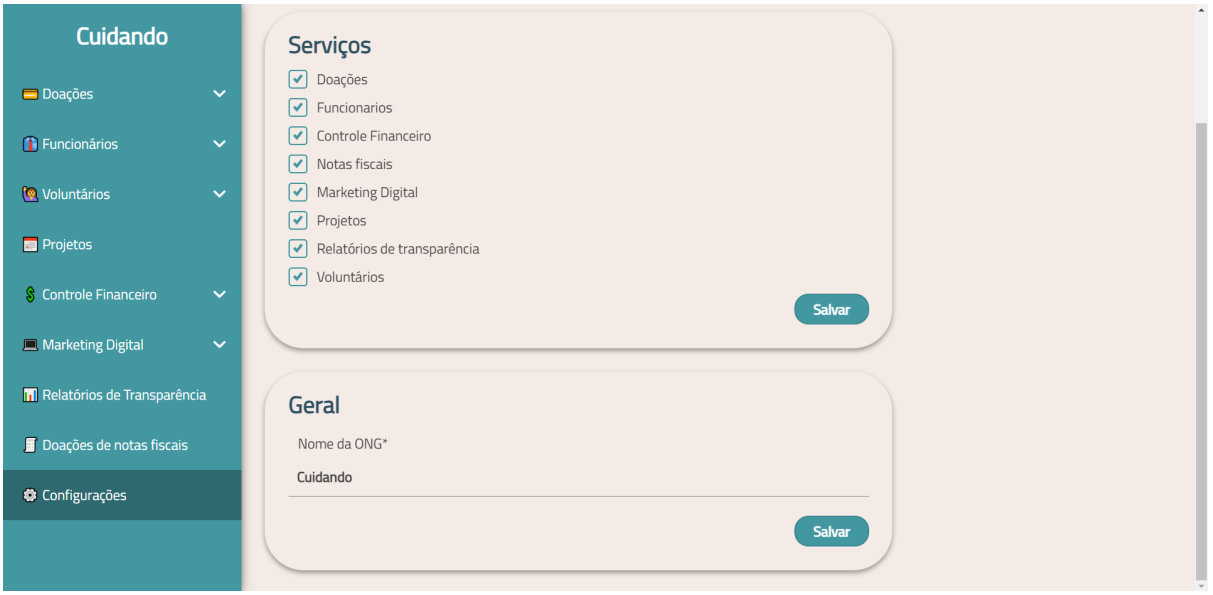
Fonte: Produção própria

Figura 3.6.23 - Tela de configurações da interface do administrador



Fonte: Produção própria

Figura 3.6.24 - Tela de configurações da interface do administrador (cont.)



Fonte: Produção própria

Figura 3.6.25 - Tela de configurações da interface do administrador (cont.)

Fonte: Produção própria

4. Considerações finais

4.1 Conclusões

O terceiro setor brasileiro é representativo no país, mas ainda possui uma série de dificuldades, como, por exemplo, a falta de uma plataforma acessível que seja capaz de auxiliar a instituição em suas atividades diárias. Pensando em solucionar essa dor, foi desenvolvido o projeto aqui descrito, baseando sua arquitetura em um sistema de microsserviços.

Em conjunto com o uso da tecnologia Docker, a arquitetura em microsserviços é a ideal para solucionar esse tipo de problema, uma vez que permite uma maior flexibilidade e modularização do software criado e, assim, a instituição que for utilizar a ferramenta, tem a total liberdade para escolher quais serviços deseja utilizar, além de também ser fácil a adição de novas funcionalidades e outras possíveis customizações. Uma das premissas do projeto era a sua acessibilidade e, pensando nisso, o código fonte da aplicação é open-source e foi feito com tecnologias que possuem um alto número de usuários, como Node.js e React.

O resultado final é uma plataforma completa e pronta para ser usada, mas que está aberta a alterações desejadas. Cada um dos microsserviços possui uma ampla cobertura de testes unitários e de integração, o que garante que o trabalho esteja funcionando da forma esperada.

4.2 Evoluções da monografia

O trabalho conta com funcionalidades básicas em cada um de seus serviços. A ideia era dar uma primeira base de produtos essenciais para o auxílio nas instituições usuárias da plataforma e, portanto, há alguns incrementos que podem ser realizados para a continuidade do projeto. Algumas das evoluções possíveis seriam:

- **Aprimoramento das funcionalidades existentes:** em conjunto com alguma organização, identificar outras necessidades que não foram contempladas no produto e implementá-las.
- **Melhorar a escalabilidade da infraestrutura:** a escalabilidade atual do sistema varia de acordo com o servidor na qual a aplicação está rodando. Pode ser interessante pensar em uma arquitetura que não possua essa dependência, o que inclusive seria mais fácil para as organizações comecem a utilizar a plataforma.

- **Integrações com outros sistemas:** para facilitar na inclusão de dados na plataforma, pode ser útil que a plataforma possua uma integração com outros sistemas ou, pelo menos, algum suporte para incluir registros a partir de uma planilha ou arquivo CSV.

REFERÊNCIAS

[1] INSTITUTO DE PESQUISA ECONÔMICA APLICADA. **Perfil das organizações da sociedade civil no Brasil**. Organização de Felix Garcia Lopez. Brasília, 2018. 18 p.

[2] INSTITUTO DE PESQUISA ECONÔMICA APLICADA. **Perfil das organizações da sociedade civil no Brasil**. Organização de Felix Garcia Lopez. Brasília, 2018. 24 p.

[3] INSTITUTO DE PESQUISA ECONÔMICA APLICADA. **Perfil das organizações da sociedade civil no Brasil**. Organização de Felix Garcia Lopez. Brasília, 2018. 85 p.

[4] DRAGONI, Nicola; LANESE, Ivan; LARSEN, Stephan; MAZZARA, Manuel; MUSTAFIN, Ruslan. **Microservices: How To Make Your Application Scale**. Moscow, Russia: A.P. Ershov Informatics Conference (the PSI Conference Series, 11th edition), 2017. Disponível em <https://hal.inria.fr/hal-01636132/file/microservices-make-application.pdf>. Acesso em 26/01/2021.

[5] **Portal OngFácil.** Disponível em <http://www.portalongfacil.com.br/>. Acesso em 26/01/2021.

[6] **Kad ONG.** Disponível em <https://www.enkad.com.br/software-ong.html>. Acesso em 26/01/2021.

[7] UEHARA, Mauricio T.; NAKAMASHI, Rogerio. **Microsserviços como abordagem arquitetônica aplicada para viabilizar a transformação digital de instituição do terceiro setor brasileiro.** São Paulo, 2020. Disponível em <https://pcs.usp.br/pcspf/wp-content/uploads/sites/8/2020/12/Monografia_PCS3860_COOP_2020_Grupo_C13.pdf>. Acesso em 07/03/2021

[8] **Instituto Adiante.** Disponível em <http://www.institutoadiante.org.br/>. Acesso em 07/03/2021.

[9] WIERINGA, Roel J. **Design Science Methodology for Information Systems and Software Engineering**. New York: Springer, 2014.

[9] **RedHat - Monolithic vs Microservices.** Disponível em <https://www.redhat.com/cms/managed-files/monolithic-vs-microservices.png>. Acesso em 07/03/2021.

[10] **Agile Manifesto**. Disponível em <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em 26/01/2021

[11] NADAREISHVILI, Irakli; MITRA, Ronnie; MCLARTY, Matt; AMUNDSEN, Mike. **Microservice Architecture: Aligning Principles, Practices, and Culture**. United States of America: O'Reilly Media, Inc, 2006.

[12] Martin, R. C. **Clean Architecture.** United States of America: Pearson Education, Inc, 2018.

[13] **PagSeguro.** Disponível em <https://pagseguro.uol.com.br/>. Acesso em 07/03/2021.

[14] TANG, Weilun; WANG, Li; XUE, Guangtao. **Design of Information System Architecture of Garment Enterprises Based on Microservices**. Journal of Physics: Conference Series, 2019. Disponível em <https://iopscience.iop.org/article/10.1088/1742-6596/1168/3/032128>. Acesso em 13/02/2021

[15] **Documentação API 4.0 PagSeguro.** Disponível em <https://dev.pagseguro.uol.com.br/reference>. Acesso em 23/10/2021.

[16] **Respositório do Produto.** Disponível em <https://github.com/GabrielaGuedes/ngos_microservices_system>. Acesso em 23/10/2021.

[17] **Amazon Elastic Compute Cloud.** Disponível em <https://aws.amazon.com/pt/ec2/>. Acesso em 23/10/2021.

[18] **Node.js.** Disponível em <https://nodejs.org/en/>. Acesso em 13/02/2021.

[19] **Stackshare.io.** Disponível em <https://stackshare.io>. Acesso em 13/02/2021.

[20] **StackOverflow.** Disponível em <https://stackoverflow.com/>. Acesso em 13/02/2021.

[21] **GitHub.** Disponível em <https://github.com/>. Acesso em 13/02/2021.

[22] **The Good and the Bad of Node.js Web App Development.** Disponível em <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/>. Acesso em 13/02/2021.

[23] **Front-end frameworks popularity (React, Vue and Angular).** Disponível em <https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190>. Acesso em 13/02/2021.

[24] **7 Pros and Cons of NoSQL.** Disponível em <https://greengarageblog.org/7-pros-and-cons-of-nosql>. Acesso em 13/02/2021.

[25] **MongoDB.** Disponível em <https://www.mongodb.com/>. Acesso em 13/02/2021.

[26] **PostgreSQL.** Disponível em <https://www.postgresql.org/>. Acesso em 13/02/2021.

[27] **PostgreSQL: a closer look at the object-relational database management system.** Disponível em <https://www.ionos.com/digitalguide/server/know-how/postgresql/>. Acesso em 13/02/2021.

[28] **Docker - What is a Container?**. Disponível em <https://www.docker.com/resources/what-container>. Acesso em 07/03/2021

[29] **Docker Compose**. Disponível em <https://docs.docker.com/compose/>. Acesso em 23/10/2021

[30] **Mocha.** Disponível em <https://mochajs.org/>. Acesso em 14/11/2021

[31] **Chai.** Disponível em <https://www.chaijs.com/>. Acesso em 14/11/2021