UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TIAGO LIMA PEREIRA

SISTEMA DE GESTÃO DO COMITÊ SANITÁRIO DE DEFESA POPULAR DE CAMPINA GRANDE

CAMPINA GRANDE - PB

2021

TIAGO LIMA PEREIRA

SISTEMA DE GESTÃO DO COMITÊ SANITÁRIO DE DEFESA POPULAR DE CAMPINA GRANDE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Tiago Lima Massoni.

CAMPINA GRANDE - PB

2021

P436s Pereira, Tiago Lima.
Sistema de gestão do Comitê Sanitário de Defesa Popular de Campina Grande. / Tiago Lima Pereira. – 2021.

11 f.

Orientador: Prof. Dr. Tiago Lima Massoni.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Sistema web de gestão. 2. Comitê Sanitário de Campina Grande - PB. 3. Pandemia. 4. Coronavírus. I. Massoni, Tiago Lima. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

TIAGO LIMA PEREIRA

SISTEMA DE GESTÃO DO COMITÊ SANITÁRIO DE DEFESA POPULAR DE CAMPINA GRANDE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Tiago Lima Massoni
Orientador – UASC/CEEI/UFCG

Professor Dr. Carlos Wilson Dantas Almeida
Examinadora – UASC/CEEI/UFCG

Trabalho aprovado em: 20 de outubro de 2021.

CAMPINA GRANDE - PB

ABSTRACT

With the emergence of the new coronavirus pandemic in 2020, in a country that has great social inequality like Brazil, it was necessary the union of the people as a tool to combat the health crisis that aggravated an already existing economic crisis, especially for the most vulnerable people residing in the periphery. In this scenario, in the entire country, the Health Committees for Popular Defense emerged, composed of academics and community leaders, as a means of popular organization to overcome the crisis. Therefore, as a proposal to add improvements in the management of the Sanitary Committee for Popular Defense of the city of Campina Grande - PB, the idea of creating a web management system arose. This system enables the registration of people and their demands, in order to optimize the results obtained by the committee helping the population.

Sistema de Gestão do Comitê Sanitário de Defesa Popular de Campina Grande

Tiago Lima Pereira
Universidade Federal de Campina Grande
tiago.pereira@ccc.ufcg.edu.br

Tiago Lima Massoni
Universidade Federal de Campina Grande
massoni@dsc.ufcg.edu.br

RESUMO

Com o surgimento da pandemia do novo coronavírus no ano de 2020, em um país que possui grande desigualdade social como o Brasil, fez-se necessário a união do povo como ferramenta de combate à crise sanitária que agravou uma crise econômica já existente, especialmente para as pessoas mais vulneráveis que residem em periferias. Nesse cenário, em todo o país, emergem os Comitês Sanitários de Defesa Popular, compostos por acadêmicos e líderes comunitários, como meio de organização popular para superar a crise. Diante disso, como uma proposta para agregar melhorias na gestão e ações do Comitê Sanitário de Defesa Popular da cidade de Campina Grande - PB, surgiu a ideia da criação de um sistema web de gestão, que possibilite o cadastro de pessoas e suas demandas, a fim de otimizar os resultados obtidos pelo comitê em auxiliar a população.

PALAVRAS-CHAVE

Comitê sanitário, coronavírus, pandemia, sistema de gestão, gerenciamento de recursos, doações.

LINKS ÚTEIS

https://github.com/tiago-lp/comite-sanitario

1. INTRODUÇÃO

A pandemia do novo coronavírus no ano de 2020 trouxe diversos desequilíbrios econômicos no mundo. O Brasil teve sua economia afetada e a população vivencia as consequências disso [1]. No contexto de um país que possui uma grande desigualdade social, como o Brasil, a pandemia apenas torna isso mais evidente. Diante disso, o movimento camponês intitulado “Liga dos Camponeses Pobres” fez um chamado através do jornal “A Nova Democracia”, expondo as contradições do Estado em relação à população, diante do cenário da pandemia, e sugerindo a organização popular como ferramenta de superação da crise [2].

Com a organização popular como forma de combater a crise do Coronavírus, diversas cidades aderiram ao movimento, dentre elas Campina Grande - PB, dando origem ao Comitê Sanitário de Defesa Popular de Campina Grande, criado em abril de 2020[3]. O CSDP-CG é composto por professores, alunos e líderes comunitários, e suas ações são organizadas em defesa dos direitos do povo à saúde, alimentação e transporte, no período da crise sanitária, especialmente em periferias, onde residem as populações mais vulneráveis.

Uma das dificuldades enfrentadas pelo Comitê Sanitário é a de ter um mapeamento adequado entre pessoas e as suas demandas, como por exemplo alimentos, roupas, dinheiro, remédios e material de limpeza. Atualmente o gerenciamento de pessoas e recursos é feito de forma manual. Diante dessa dificuldade, surge a ideia do Sistema de Gestão do Comitê Sanitário de Defesa Popular como uma proposta para melhorar ações dos membros do comitê e destinar doações com maior precisão para cada demanda específica. O CSDP-CG é uma ideia que pretende se estabelecer mesmo após a pandemia, como uma forma de organização popular perante os descasos do Estado para com a população. Dessa forma, a ideia do Sistema de Gestão também pode ser útil a longo prazo.

2. PROBLEMA

No dia 24 de abril de 2020, estudantes, professores e líderes comunitários de Campina Grande se reuniram para conformar o Comitê Sanitário de Defesa Popular - Campina Grande. O objetivo do comitê é o de oferecer alternativas sociais, sanitárias e científicas que estejam relacionadas com as necessidades de defesa do povo em meio à epidemia do Coronavírus no município de Campina Grande - PB, organizando ações de defesa popular contra a contaminação pela COVID-19 e em defesa dos direitos do povo à saúde, alimentação, transporte e moradia neste período, bem como fortalecendo os vínculos da comunidade acadêmica com a população.

Desde então os integrantes do CSDP-CG têm atuado junto às populações mais vulneráveis dos bairros periféricos da cidade, denunciando a crise econômica e o descaso do Estado, cujo teve o agravamento de suas condições com o surgimento da pandemia. Dentre as ações desenvolvidas pelo CSDP-CG estão incluídas: assessoria jurídica para as denúncias de moradores frente às promessas não cumpridas da Prefeitura; campanha de arrecadação de dinheiro e tecidos para produção e distribuição de máscaras; panfletagem, bate-papo sobre a situação sanitária e distribuição de artigos de higiene.

O problema principal das ações é que, apesar de existir uma organização, esta não possui um gerenciamento eficaz das medidas a serem tomadas de acordo com as demandas existentes. Isso acaba gerando uma dependência com determinadas pessoas do comitê que estão a par de informações específicas para tomarem decisões, ou até mesmo a falta de distribuição da informação de locais ou pessoas que estão com uma necessidade mais urgente. A proposta do Sistema de Gestão

é diminuir esse problema, provendo uma ferramenta para melhorar a organização e otimizar os resultados das ações destinadas à população, obtendo e fazendo uso dos dados sobre demandas, sejam jurídicas, financeiras ou de qualquer outra ordem. O sistema também pretende aumentar o espectro de união popular, dando a possibilidade de trazer mais pessoas para contribuírem com a organização, mesmo que esporadicamente, por meio da disponibilidade da informação das doações.

3. SOLUÇÃO

Este trabalho tem como objetivo final desenvolver uma solução para o problema citado, através de uma aplicação web que forneça aos membros do Comitê Sanitário de Campina Grande a capacidade de cadastrar dados de pessoas e suas necessidades.

A aplicação visa ser simples e responsiva, com capacidade de executar em qualquer navegador web através de desktops e smartphones.

Através da aplicação será possível agir em cima de alguma demanda de forma mais eficiente, onde será possível cadastrar doações e distribuir de acordo com as demandas de cada pessoa cadastrada no sistema, além da possibilidade de localizar essas pessoas e ter métricas gerais sobre as doações.

3.1 Funcionalidades

O usuário administrador inicial, delegado a alguém por padrão, poderá cadastrar outros administradores. Os usuários da aplicação serão apenas pessoas responsáveis pela organização do Comitê Sanitário, logo, todos serão administradores. Estes usuários poderão cadastrar pessoas na base de dados com suas informações socioeconômicas, localização geográfica, além das informações básicas como nome, idade, contato (se houver) e ocupação. Isto serve para se ter um acompanhamento de quem precisa de doações, e saber onde localizar essas pessoas. Além disso, será possível cadastrar doações, que podem ser criadas com o estado pendente ou já recebida, além de indicar o tipo de doação e o destinatário (pessoas cadastradas ou a própria organização). O sistema também oferece uma interface amigável, tanto pelo frontend com um template responsivo, quanto pela área de administração da API.

3.2 Arquitetura

A arquitetura utilizada para a aplicação web desenvolvida foi a Cliente-Servidor[4]. Esse modelo diminui a complexidade das aplicações dividindo a carga de processos, executando no navegador ou aplicativo do cliente, tendo a responsabilidade de fazer requisições e obter respostas do servidor, como mostrado na Figura 1. Nesse caso, o cliente é o Frontend, responsável por exibir uma interface amigável para o usuário com informações do servidor (API e Banco de dados). O servidor é o container que é composto pelo Banco de Dados e a API, responsável pelo gerenciamento dos dados e da comunicação com o cliente.

O diagrama de container seguindo o modelo C4[5] é mostrado na Figura 2, onde é ilustrado o container da API, que contém o banco de dados e a API REST, inserido dentro do container geral do sistema que possui o frontend se comunicando com a API. Um exemplo de como o Frontend escrito em JavaScript se comunica com o container da API é demonstrado na Figura 3, onde é chamado uma requisição *GET* dos modelos de doações na API. Já um exemplo de como a API REST escrita em Python se comunica com o banco de dados é ilustrado na Figura 4, onde apenas uma classe é definida para que framework Django possibilite automaticamente que os métodos GET, POST, PUT e DELETE do modelo de doações fiquem disponíveis para o Frontend.

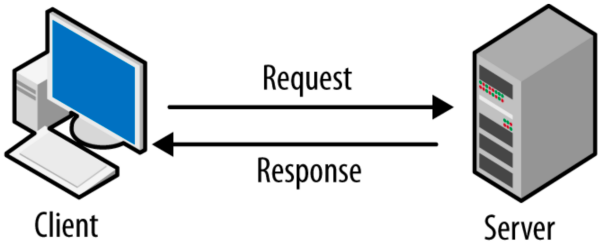

Figura 1 - Arquitetura Cliente-Servidor

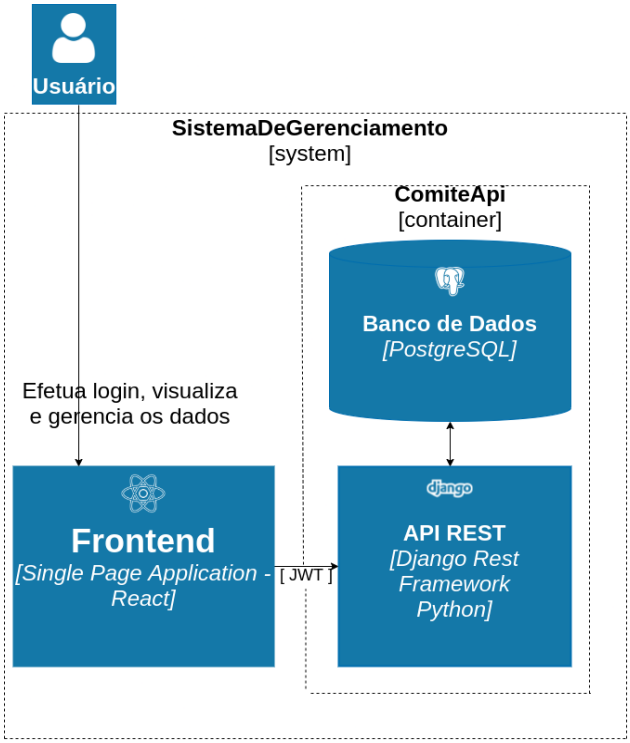

Figura 2 - Diagrama de Containers (C4)

```
const donationsAsyncRequestStarted = () => ({
  type: DONATIONS_ASYNC_REQUEST_STARTED
});

const getDonationsSuccess = (data) => ({
  type: GET_DONATIONS_SUCCESS,
  payload: data
});

const getDonationsFailed = () => ({
  type: GET_DONATIONS_FAILED,
});

export const getDonationsRequest = () => {
  return dispatch => {
    dispatch(donationsAsyncRequestStarted());
    Api.get('donations/')
    .then(response => {
      dispatch(getDonationsSuccess(response.data))
    })
    .catch(error => {
      dispatch(getDonationsFailed());
    })
  };
};
```

Figura 3 - Requisição GET do modelo de doações

```
class DonationViewSet(BaseViewSet):
    permission_classes = (IsAuthenticated,)

    queryset = models.Donation.objects.filter(deleted=False)
    serializer_class = serializers.DonationSerializer
```

Figura 4 - View do modelo de Doações

3.3 Tecnologias utilizadas no servidor

O servidor utiliza o framework web Django[6], que utiliza o padrão Model-Template-View. O motivo de optar pelo Django foi a familiaridade e facilidade que a ferramenta oferece, dando a possibilidade de criar APIs sem escrever soluções largamente usadas que já existem, como por exemplo criar modelos no Banco de Dados e os métodos para operar com esses modelos, comumente os métodos de Create, Read, Update e Delete (CRUD). Dessa forma o Django facilita a criação de modelos no Banco de Dados, apenas codificando uma classe em Python definindo os nomes dos campos e os tipos dos mesmos. Ao criar os modelos através de uma classe em Python que herda a classe Model do Django, o framework cria o CRUD do modelo automaticamente ao definir uma rota para esse modelo. A Figura 5 mostra todos os métodos criados para o modelo de Doações da aplicação. O Django também é muito adaptável, permitindo plugar e desplugar aplicações apenas modificando arquivos de configurações, como por exemplo adicionar autenticação e modificar a integração com qualquer banco de dados. Além disso, disponibiliza também uma área de administração que é acessível pela web, com autenticação e possibilidade de gerenciar os modelos. A parte de autenticação do sistema foi feita utilizando o Simple-JWT[7], que é comumente usado em aplicações web.

A persistência dos dados ficou a cargo do PostgreSQL[8] por ter uma boa garantia em relação à integralidade dos dados e ter uma consistência melhor dos modelos com o surgimento de novas funcionalidades e consequentemente novos modelos na base de dados.

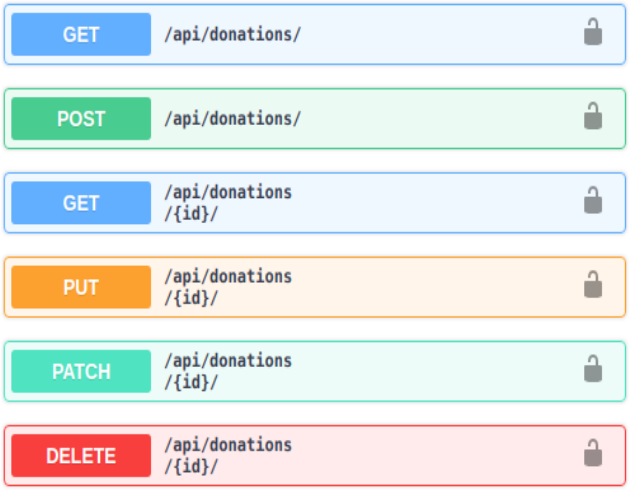

Figura 5 - CRUD de doações gerado pelo Django

3.4 Estrutura da API

O desenvolvimento da API foi feito utilizando as recomendações do Django, portanto, a estrutura do código foi gerada pela inicialização de um projeto padrão utilizando a própria biblioteca instalada localmente. Na pasta *comite_api* existem os arquivos responsáveis pela execução do servidor e o arquivo *settings.py* que é responsável pela definição da conexão com o banco de dados, middlewares e todos os recursos disponíveis no Django. A modificação necessária para a estruturação da API foi feita com a adição de um diretório nomeado *core*, onde estão localizados os arquivos principais da aplicação. Na pasta *core* temos a pasta com as migrações do banco de dados, que torna possível integrar a aplicação com qualquer outro banco de dados, caso seja necessário. Os arquivos de migração registram o histórico de cada criação de modelo ou mudança nos modelos já existentes. Ainda na pasta *core* temos os arquivos das aplicações que funcionam junto com o Django, os modelos, os serializers e as views. Na pasta *requirements* também existem os arquivos de dependências de bibliotecas externas e na raíz da API estão os arquivos de infraestrutura utilizados para executar a aplicação através do Docker[9] tanto localmente como para o ambiente de homologação, como o arquivo *manage.py* que contém os scripts para execução da API e o arquivo *cloudmigrate.yaml* para definição do deploy. Toda a estrutura citada é mostrada na Figura 6.

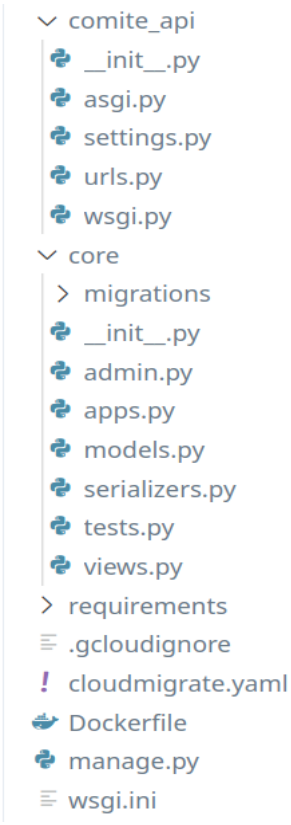

Figura 6 - Estrutura do backend

3.5 Tecnologias utilizadas no Frontend

O lado do cliente foi implementado utilizando a biblioteca ReactJs[10], escrita em JavaScript e mantida pelo Facebook. A escolha dessa biblioteca foi direcionada pela experiência do autor em projetos utilizando ReactJs e pela estrutura baseada em componentes, que facilita o desenvolvimento e expansão da aplicação. As principais bibliotecas JavaScript utilizadas no projeto do frontend foram react-redux[11] para gerenciamento de estados da aplicação, google-map-react[12] para renderização do Google Maps[13] na aplicação e react-places-autocomplete[14] para buscar endereços pela API do Google Maps. Além disso, foi utilizado um template gratuito chamado minimal[15], escrito utilizando o Material-UI[16]. A Figura 7 mostra o uso do template com as informações mostradas no Dashboard: total de dinheiro recebido em doações, número de doações recebidas, número de doações pendentes e número de pessoas cadastradas no sistema. A Figura 8 mostra o uso do Google Maps integrado ao sistema, na tela de edição do cadastro de uma pessoa onde é possível pesquisar um endereço no campo de *localização* do formulário. Ao pesquisar um endereço, sugestões são mostradas, e ao selecionar alguma das sugestões a biblioteca renderiza um mapa com a localização do endereço selecionado. O mapa também aparece no formulário de cadastro de pessoa no sistema e na visualização de detalhes do cadastro de uma pessoa.

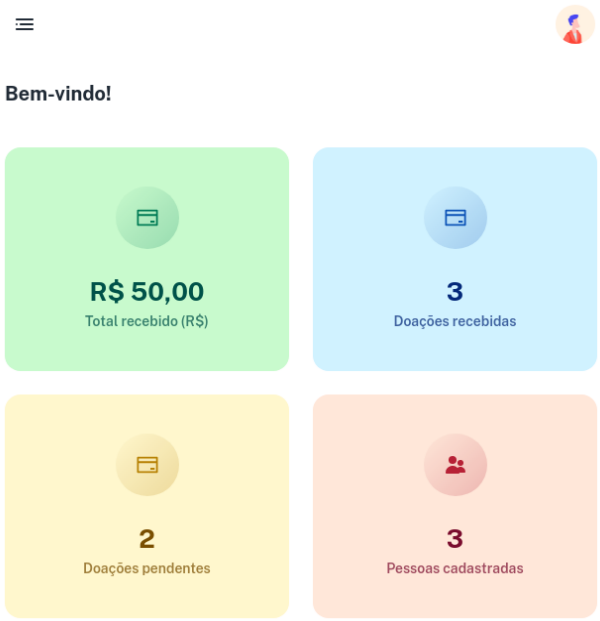

Figura 7 - Dashboard utilizando o template minimal

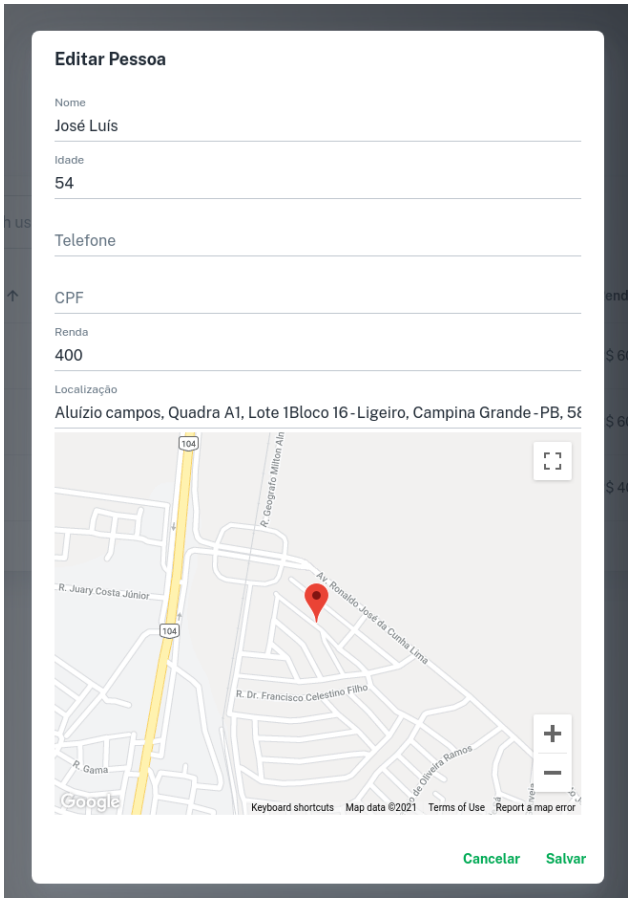

Figura 8 - Google Maps integrado com a aplicação

3.6 Tecnologias do sistema em uso

Todo o ambiente de homologação está hospedado no Google Cloud Platform[17] que oferece soluções na nuvem para diversas demandas. A plataforma fornece um voucher de 300 dólares por 3 meses para experimentar a plataforma. As principais APIs e serviços ativados para uso foram o Cloud Run, Cloud SQL, App Engine, Secrets Manager, Maps JavaScript e Geocoding.

O serviço Cloud Run foi utilizado para executar a aplicação Django de forma escalável. Além disso, o serviço Cloud SQL foi utilizado para criar uma instância do Banco de Dados PostgreSQL na nuvem, com 4 cpus, 26GB de memória RAM e 100GB de armazenamento SSD, a qual a aplicação Django executando no Cloud Run se comunica.

O App Engine foi utilizado para o deploy do frontend feito em ReactJs. O App Engine possui uma funcionalidade muito útil de versionamento, onde é possível deixar no ar uma versão específica desde a versão do primeiro deploy até o mais recente.

O serviço Secrets Manager foi utilizado para armazenar informações sensíveis que os serviços precisam acessar e não deve estar definido no próprio código, tais quais usuário do banco de dados, senha, host e algumas chaves de API do Google Maps.

As APIs Maps JavaScript e Geocoding foram utilizadas, respectivamente, para renderização de mapas no frontend e para conseguir resgatar endereços através de pesquisas dentro da aplicação durante o preenchimento da localização no cadastro de pessoas. Essas APIs permitem o funcionamento das bibliotecas google-map-react e react-places-autocomplete, utilizadas no frontend.

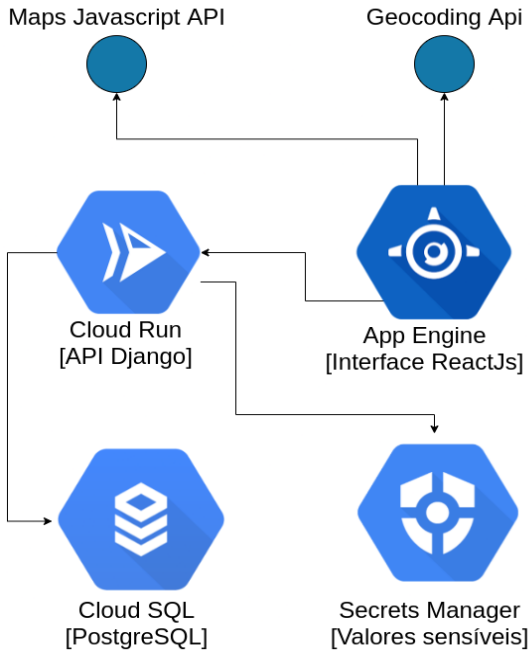

Figura 9 - Uso do Google Cloud Platform

3.7 Sistema em uso

A aplicação foi disponibilizada por uma semana, inicialmente apenas com o Django, que por si já disponibiliza uma interface para os usuários. Com essa aplicação, dois membros do comitê sanitário passaram a testar cadastrando algumas pessoas e doações, com a limitação de não ter integração com Google Maps. Posteriormente, após a disponibilização do frontend por uma semana, mais três pessoas passaram a fazer uso do sistema, totalizando cinco pessoas usando o sistema. Agora todos com a possibilidade de utilizar o Google Maps integrado ao sistema e também sendo possível ver o resumo das informações gerais de doações no Dashboard.

Após as duas semanas totais de uso entre os dias 15/09/2021 e 29/09/2021, foi disponibilizado um formulário do Google, do dia 29/09/2021 até 30/09/2021, para que os cinco usuários pudessem responder algumas perguntas a respeito do sistema. As principais perguntas foram "O Login funciona corretamente?", "O google maps dentro da aplicação funciona corretamente?", "Teve algum problema com o uso do sistema? (tanto pelo Django Admin quanto pelo Frontend)" e "O quanto você acha que o sistema irá contribuir positivamente com as atividades do comitê sanitário?".

Os resultados obtidos foram satisfatórios, como demonstrado na Figura 10. A respeito da funcionalidade do Login e da integração com o Google Maps, todos responderam que está funcionando corretamente. Apenas uma pessoa relatou um problema que ocorreu algumas vezes ao cadastrar doações pelo frontend e posteriormente foi identificado um pequeno problema no gerenciamento de estados, que foi corrigido em seguida. A respeito da pergunta sobre a contribuição positiva nas atividades do comitê sanitário, todos responderam com nota 5 numa escala de 1 a 5, onde 1 é nada a contribuir e 5 é muito a contribuir.

O Login funcionacorretamente?	Sim	Sim	Sim	Sim	Sim
O google maps dentro da aplicação funciona corretamente?	Sim	Sim	Sim	Sim	Sim
Teve algum problema com o uso do sistema? (tanto pelo Django Admir quanto pelo Frontend)	Não	Sim	Não	Não	Não
Se sua resposta foi sim para a pergunta anterior, qual foi o problema?		As vezes ao cadastrar doações os formulários já estão preenchidos.			
O quanto você acha que o sistema irá contribuir positivamente com as atividades do comitê sanitário?	5	5	5	5	5

Figura 10 - Respostas do formulário

4. EXPERIÊNCIA

Nesta seção serão apresentadas as experiências a respeito do processo de desenvolvimento, desafios, limitações e possíveis trabalhos futuros.

4.1 Processo de desenvolvimento

O desenvolvimento da ideia em si surgiu durante a pandemia da COVID-19, ainda no ano de 2020, como uma ideia

pessoal e colaborativa. Posteriormente apresentei a ideia a um membro do comitê sanitário de Campina Grande, onde surgiram alguns requisitos e melhores definições para o contexto do uso real. Primeiramente, foi feita uma reunião com duas pessoas que participam das atividades do comitê sanitário para discutir soluções e quais seriam as funcionalidades necessárias a serem implementadas. Após isso, foram decididas novas tecnologias para o processo de desenvolvimento (Django e ReactJs), em substituição ao desenvolvimento inicial feito de forma individual utilizando a linguagem Rust[18] para o backend e VueJs[19] para o frontend. Desde o início, a definição dos modelos do banco de dados já estavam prontas, com isso foi implementado toda a API utilizando o Django com PostgreSQL. Após isso foi criado um backlog das funcionalidades para que o desenvolvimento acontecesse de forma mais concisa, seguindo uma cadência de entregas. Seguindo o backlog foi implementado toda a integração com o frontend e o deploy para o ambiente de homologação utilizando o Google Cloud Platform, que foi disponibilizado para os usuários. O versionamento do código foi feito utilizando o próprio Github.

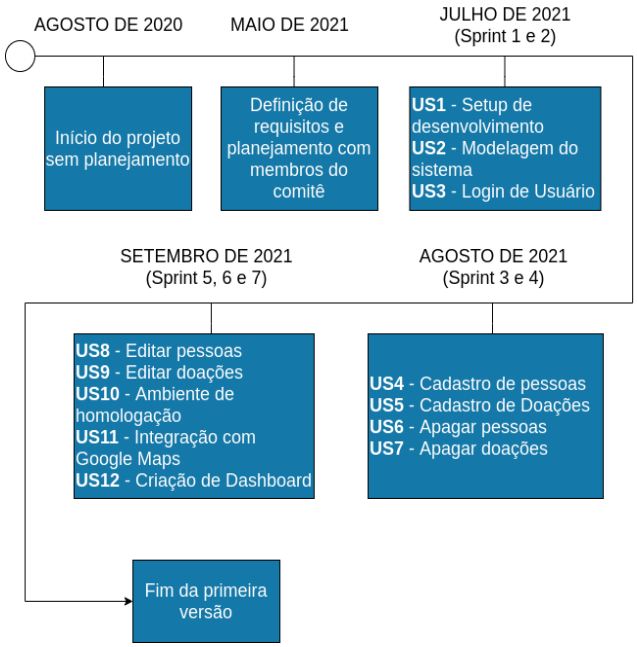

Figura 11 - Processo de desenvolvimento

4.2 Desafios

Os principais desafios estão relacionados ao processo de desenvolvimento, que por falta de experiência foi iniciado de forma avulsa, como uma ideia pessoal, antes de vir a ser um projeto que seria realmente posto em uso. Só depois de um tempo que foi feito um backlog para acompanhamento de funcionalidades, com prazos seguindo sprints do método SCRUM[20]. Também foi dificultoso dar andamento no projeto inicialmente criado com a linguagem de programação Rust, por esse motivo foi redefinido para ser usado Python.

Outro desafio foi em relação a disponibilizar o sistema para homologação dos usuários, que inicialmente foi pensado em utilizar uma arquitetura utilizando o Kubernetes Engine do Google ou o Heroku, mas que por fim foi mais amigável implementar esse ambiente utilizando os serviços do Google Cloud Platform citados na seção 3.6, que mesmo dispondo de ótimas documentações para deploy de uma aplicação Django, ainda assim existiam alguns erros na sequência de passos a serem feitos, notavelmente por falta de atualização de algumas documentações do Google, mas que usando o que foi aprendido no processo foi possível implantar o ambiente corretamente.

4.3 Limitações

Uma das limitações encontradas é na disponibilidade do sistema, que inicialmente terá um prazo de três meses, disponibilizado pelo teste gratuito com o voucher de 300 dólares fornecido pelo Google. Posteriormente pode vir a ser cobrado, dependendo da quantidade de recursos utilizados. A priori, o que será utilizado não chegará a vir no faturamento mensal, e caso venha será um valor simbólico que os voluntários do comitê conseguem arcar tranquilamente, mas que de toda forma isso pode vir a ser uma limitação.

4.4 Trabalhos futuros

Para trabalhos futuros, existe a possibilidade de novas funcionalidades que surgirem de acordo com o uso dos membros do comitê através de ideias e demandas que possam aparecer, bem como a expansão do sistema para uso pelo comitê sanitário de outras cidades, com uma réplica do sistema completo inicializado para uma determinada cidade.

AGRADECIMENTOS

Ao meu orientador Tiago Massoni, por sua excelência profissional, nesta e em outras disciplinas do curso que eu tive a oportunidade de tê-lo como professor, além de ser uma pessoa com muita empatia e bom humor. Aos amigos que entraram no curso comigo, Arthur Sampaio e Mariana Mendes e a todos os outros que me tornei amigo ao longo do curso e levo até hoje na minha vida, Agnaldo, Ariann, Damião, Dayvson, Itamar, João Pedro, João Maurício, Luan, Matheus Melo, Nilton, Ruan, Rubens, Samuel, Victor, Wesley e Yovany. Aprendi muito com todos, em cada nuance do dia a dia nessa jornada. Agradeço a todos por terem tornado cada dia mais leve e alegre dentro do ambiente acadêmico. Aos meus amigos mais antigos, Alison, Emanuel Joivo, Gabriel Lobão, Raul, Sabrina e Thiago David que permaneceram na minha vida mesmo durante a minha ausência demandada pelo curso. A Cilas e Milena, que foram como uma família para mim ao longo de todos esses anos de curso, que me ajudaram em situações difíceis e que me trouxeram tantas coisas boas. A Thaís Torquato por ter despertado a ideia deste trabalho ao me colocar em contato com o Comitê Sanitário de Campina Grande, que até então era desconhecido para mim. A diversos outros amigos que não será possível listar todos aqui, mas que foram igualmente importantes nesse processo.

Os mais importantes agradecimentos são ao meu tio-avô Carlos Alberto Lima, que me deu suporte e me inspirou em todos os aspectos da vida, desde a minha infância, como exemplo de um ser humano grandioso e lotado de virtudes. Aos meus pais a quem eu desejo despertar orgulho com essa conquista, Josenildo e Lenilda, que também me inspiram bondade e aos meus irmãos de sangue, Daniel, Daniella e Rafael que fizeram parte da constituição do que sou hoje.

Tudo isso só foi possível graças a vocês e a força divina que nos anima.

5. REFERÊNCIAS

[1] MENDONÇA, Heloísa. PIB tem queda histórica de 9,7% no segundo trimestre e pandemia arrasta o Brasil para recessão. **EL PAÍS**, 2020. Disponível em: <https://brasil.elpais.com/economia/2020-09-01/pib-tem-queda-historica-de-97-no-segundo-trimestre-e-pandemia-arrasta-o-brasil-para-recessao.html>. Acesso em: 23 de Setembro de 2021.

[2] Liga dos Camponeses Pobres: 'A guerra do governo não é contra o vírus! É contra o povo!'. **A Nova Democracia**, 2020. Disponível em: <https://anovademocracia.com.br/noticias/13224-liga-dos-camponeses-pobres-a-guerra-do-governo-nao-e-contra-o-virus-e-contra-o-povo>. Acesso em: 23 de Setembro de 2021.

[3] PB: Estudantes, professores e líderes comunitários criam Comitê Sanitário em Campina Grande. **A Nova Democracia**, 2020. Disponível em: <https://anovademocracia.com.br/noticias/13423-pb-estudantes-professores-e-lideres-comunitarios-criam-comite-sanitario-em-campina-grande>. Acesso em: 23 de Setembro de 2021.

[4] Arquitetura cliente-servidor. **Canalti**, 2018. <https://www.canalti.com.br/arquitetura-de-computadores/arquitetura-cliente-servidor>. Acesso em 24 de Setembro de 2021

[5] The C4 model for visualising software architecture. **Simon Brown.** Disponível em: <https://c4model.com/>. Acesso em 24 de Setembro de 2021.

[6] Django The web framework for perfectionists with deadlines. **Django Project**. Disponível em: <https://www.djangoproject.com/>. Acesso em 24 de Setembro de 2021.

[7] Simple JWT. **Django Rest Framework**. Disponível em: <https://django-rest-framework-simplejwt.readthedocs.io/en/latest/>. Acesso em 24 de Setembro de 2021.

[8] PostgreSQL: The World's Most Advanced Open Source Relational Database. **Site oficial do PostgreSQL**, 2021. Disponível em: <https://www.postgresql.org/>. Acesso em 26 de Setembro de 2021.

[9] O que é Docker? **Red Hat Forum**, 2021. Disponível em: <https://www.redhat.com/pt-br/topics/containers/what-is-docker>. Acesso em 27 de Setembro de 2021.

[10] Uma biblioteca JavaScript para criar interfaces de usuário. **Site Oficial do ReactJS**. Disponível em: <https://pt-br.reactjs.org/>. Acesso em 27 de Setembro de 2021.

[11] Official React bindings for Redux. **Site oficial do Redux**. Disponível em: <https://react-redux.js.org/>. Acesso em 29 de Setembro de 2021.

[12] google-map-react. **Github**. Disponível em: <https://github.com/google-map-react/google-map-react>. Acesso em 29 de Setembro de 2021.

[13] Google Maps. **Wikipédia, a enciclopédia livre**. Disponível em: <https://pt.wikipedia.org/wiki/Google_Maps>. Acesso em 29 de Setembro de 2021.

[14] react-places-autocomplete. **Github**. Disponível em: <https://github.com/hibiken/react-places-autocomplete#readme>. Acesso em 30 de Setembro de 2021.

[15] Minimal Free - Client & Admin Dashboard. **Site oficial**. Disponível em: <https://material-ui.com/store/items/minimal-dashboard-free/>. Acesso em 30 de Setembro de 2021.

[16] We're on a mission to make building UIs more accessible. Disponível em: <https://mui.com/pt/about/>. Acesso em 30 de Setembro de 2021.

[17] Por que o Google Cloud? Disponível em: <https://cloud.google.com/why-google-cloud>. Acesso em 01 de Outubro de 2021.

[18] Uma linguagem empoderando todos a construir softwares confiáveis e eficientes. Disponível em: <https://www.rust-lang.org/pt-BR>. Acesso em 01 de Outubro de 2021.

[19] The Progressive JavaScript Framework. Disponível em: <https://vuejs.org/>. Acesso em 01 de Outubro de 2021.

[20] SUTHERLAND, Jeff. **SCRUM: A arte de fazer o dobro do trabalho na metade do tempo**. 2 ed. São Paulo: Leya, 2016.