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

VICTOR HUGO FERNANDES DE SOUSA

EVOLUÇÃO ARQUITETURAL DO SISTEMA HELIOS PARA A REALIDADE DO CENTRO ACADÊMICO DE CIÊNCIA DA COMPUTAÇÃO DA UFCG

CAMPINA GRANDE - PB

2019

VICTOR HUGO FERNANDES DE SOUSA

EVOLUÇÃO ARQUITETURAL DO SISTEMA HELIOS PARA A REALIDADE DO CENTRO ACADÊMICO DE CIÊNCIA DA COMPUTAÇÃO DA UFCG

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.

Orientadora: Professora Dra. Francilene Procópio Garcia.

CAMPINA GRANDE - PB

2019

S725e Sousa, Victor Hugo Fernandes de.
Evolução arquitetural do Sistema Helios para a realidade do Centro Acadêmico de Ciência da Computação da UFCG. / Victor Hugo Fernandes de Sousa. – 2019.

10 f.

Orientadora: Profa. Dra. Francilene Procópio Garcia.
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 Helios. 2. Evolução de arquitetura – sistema Helios. 3. Sistema de votação. 4. Centro Acadêmico de Ciência da Computação UFCG. 5. Aplicação Python. 6. API REST. I. Garcia, Francilene Procópio. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

VICTOR HUGO FERNANDES DE SOUSA

EVOLUÇÃO ARQUITETURAL DO SISTEMA HELIOS PARA A REALIDADE DO CENTRO ACADÊMICO DE CIÊNCIA DA COMPUTAÇÃO DA UFCG

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:

Professora Dra. Francilene Procópio Garcia
Orientadora – UASC/CEEI/UFCG

Professor Me. Pedro Sergio Nicolletti
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 25 de novembro de 2019.

CAMPINA GRANDE - PB

Evolução Arquitetural do Sistema Helios para a Realidade do Centro Acadêmico de Ciência da Computação da UFCG

Trabalho de Conclusão de Curso

Victor Hugo Fernandes de Sousa
victor.sousa@ccc.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba

RESUMO

Um sistema de votação para alunos que fazem ciência da computação deve ir além de um simples aplicativo, ele também deve oferecer uma interface que possibilite a comunidade estudantil aplicar os diversos conhecimentos adquiridos no curso sobre ela. Um sistema de código aberto e que possui uma grande comunidade como o Helios é uma opção ideal para o propósito de realizar eleições e consultas aos alunos de forma prática e segura, porém, sua arquitetura não permite realizar comunicações entre outros sistemas de forma simples. Esse trabalho tem o objetivo de evoluir a arquitetura do sistema Helios para que possa oferecer uma API REST como também desenvolver um cliente que utilize sua nova arquitetura.

1 INTRODUÇÃO

Como qualquer ambiente de software, integração de sistemas acontecem a todo tempo. Evolução de tecnologias e mudanças necessárias nos negócios são razões para melhorar ou substituir aplicações por outras melhores, rápidas e com ambiente de software atualizados[8] e é conhecido que o melhor padrão para o desenvolvimento de software que facilite a integração entre sistemas é aplicar a arquitetura orientada a serviços, conhecido como SOA (Service-Oriented Architecture), proposto por Roy Schulte e Yefim Natis em abril de 1996 [4] no artigo “Service Oriented"Architectures”. Ao implementar características de infraestrutura da Web ao SOA surgiu o REST (Representational State Transfer) na qual se tornou o mais popular estilo de arquitetura para o desenvolvimento de sistemas online.

Por padrão, projetos desenvolvidos com o framework Django não implementam soluções em REST, já que o framework foi projetado para desenvolvimento de aplicações web em MVT (Model-View-Template), sem integração ou estruturação de serviços. Esse é o caso do Helios, projeto de código aberto de votação e auditoria online e que possuiu uma comunidade assídua no github, contando no momento em que esse trabalho é escrito com mais de 150 forks. Uma de suas ramificações foi implementado pelo Instituto Federal de Santa Catarina (IFSC), que evoluiu o projeto para um sistema de escolha do conselho universitário da instituição.

Por ser um sistema reconhecidamente seguro, utilizado na escolha do Conselho da Sociedade Brasileira de Computação (SBC) para o biênio de 2011/2013 [7], e em ambiente acadêmico como já descrito, escolhemos o Helios em sua ramificação criada pelo IFSC para ser implementado como o sistema de votação do centro acadêmico do curso de ciência da computação da Universidade Federal de Campina Grande. A comunidade estudantil desse curso é bastante diversificada, com 748 alunos matriculados em 2019.2, além do centro acadêmico do curso, conta com grupos organizados pelos próprios estudantes com ênfase em áreas específicas da computação: como o grupo Guardians que desenvolve soluções em adminstração de sistemas e o grupo OpenDev-UFCG que desenvolve e fomenta projetos em código aberto. Desta forma, para que um sistema de votação se adeque às necessidades dessa comunidade, é necessário que ela dê fácil suporte a possíveis integrações entre sistemas que utilizem as mais diversas tecnologias.

Para isso, propomos implementar uma solução em REST para o projeto Helios com o uso do Django REST Framework. Ao evoluir a arquitetura desse projeto, além de facilitar às integração com outros sistemas, pretendemos contribuir ao projeto original com a conteinerização de sua aplicação e criação de um aplicativo cliente desenvolvido em Angular 8 que utilize a API criada.

2 HELIOS VOTING

O Sistema de votação Helios permite a realização de eleições através da Internet e com auditoria aberta ao público (End-to-end voter verifiable – E2E), fazendo uso de criptografia homomórfica, é possível computar o resultado final de uma eleição sem que seja necessário ter acesso ao voto (descriptografar o voto) de cada eleitor [6]. Segundo o autor [Adida 2008], trata-se do primeiro sistema de votação on-line baseado na web, permitindo que o eleitor verifique que seu voto foi depositado, que todos os votos depositados sejam exibidos publicamente em forma criptografada e que qualquer um possa verificar que os votos depositados foram corretamente apurados [2]. Para esse trabalho foi utilizado o Helios em sua terceira versão.

2.1 Arquitetura

O sistema faz uso do modelo de rede cliente-servidor (Figura 1), na qual o servidor tem a responsabilidade de fazer o processamento das solicitações do cliente e se comunicar com o banco de dados, além de renderizar e entregar as páginas HTML solicitadas pelo cliente, que por sua vez faz o processamento da cifragem da cédula de votação que será enviado ao servidor.

O projeto Helios pode ser abstraído em quatro principais componentes:

- Montador de eleição: sistema web de criação e gerenciamento de eleições
- Cabina de votação: sistema web para que os eleitores façam suas escolhas e as cédulas de votação seja cifrada
- Servidor de votação de cédulas: o servidor para o qual as cédulas preenchidas são enviadas.

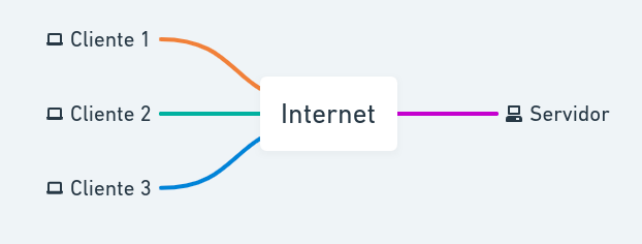


Figura 1: Arquitetura cliente-servidor

- Auditoria: local em que todos os dados são postados no final de uma eleição.

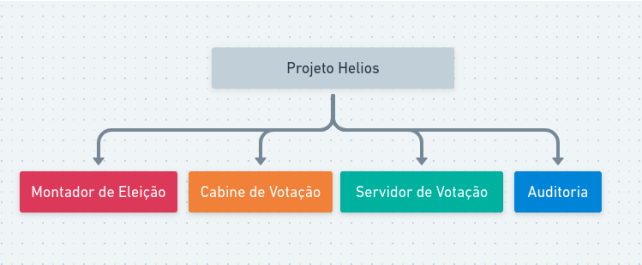


Figura 2: Componentes do Projeto Helios

2.1.1 Tecnologias do Servidor. A aplicação foi desenvolvida em Python 2.7 com o framework Django na versão 1.8, além do servidor de banco de dados em PostgreSQL para a persistência das informações. Django é um framework para desenvolvimento rápido para web, foi criado originalmente como sistema para gerenciar um site jornalístico na cidade de Lawrence, no Kansas, tornou-se um projeto de código aberto e foi publicado sob a licença BSD em 2005 [11]. O PostgreSQL, é um sistema de gerenciamento de banco de dados do tipo objeto-relacional com ênfase em extensibilidade. [5]

2.1.2 Estrutura do Servidor. Em Django, o termo aplicação descreve um pacote em Python que provê algumas funcionalidades, o sistema Helios é composto por cinco aplicações que implementam os quatro componentes já descritos, são elas:

- Helios
- Helios Auth
- Server UI
- Helios log
- Helios Institution

Para cada aplicação existe um arquivo "url.py" que é responsável por receber as solicitações http e direcioná-las para métodos que são declarados no arquivo "view.py", que por sua vez faz consultas ao banco de dados atráves do arquivo "models.py". A principal aplicação é o Helios, que é responsável pela criação, gerência e computação dos votos das eleições, já o Helios Auth é responsável pela autenticação e criação dos usuários e eleitores no sistema. As demais aplicações está relacionada com servir arquivos estáticos (Server UI), gerenciar e armazenar meta-informações do sistema (Helios log) e uma aplicação voltada para gerenciamento de usuários e telas de informações acerca do sistema (Helios Institution).

2.1.3 Tecnologias do Cliente. Como grande parte do cliente é processada pelo Django e apenas é entregue o HTML, fica a cargo do cliente o processamento da cifragem das cédulas de votação. Para isso, o Helios utiliza a linguagem JavaScript com o framework JQuery na versão 1.2.6. jQuery é uma biblioteca de funções JavaScript que interage com o HTML, desenvolvida para simplificar os scripts interpretados no navegador do cliente. [13]

2.1.4 Estrutura do Cliente. O cliente da cabine de votação tem duas classes principais que estão escritas nos arquivos "helios.js" e "elgamal.js". No arquivo helios.js está os métodos para gerenciar o processo de votação, além de abstrair o processo de cifragem. Já o arquivo "elgamal.js" é onde fica a computação da cifragem em si, ele faz uso da biblioteca "BigInt.js" para poder criar e manipular números de grande valor para fortalecer os cálculos matemáticos feitos no momento da cifragem.

3 PROCESSO DE VOTAÇÃO

Segundo o autor do projeto em sua publicação original [1], todo o protocolo do processo de votação utilizado pelo Helios pode ser descrito nos seguintes passos:

- Alice escolhe suas opções e verifica a escolhas feitas na sua cédula de votação, quando ela estiver satisfeita de sua escolha, Alice lança uma cédula criptografada, o que vai exigir que ela se autentique no sistema.
- O Helios pública o nome de Alice e a cédula criptografada. Qualquer pessoa, incluindo Alice, pode verificar o quadro de avisos e encontrar seu voto criptografado publicado
- Quando a eleição termina, Helios embaralha todas as cédulas criptografadas e produz uma prova não interativa do embaralhamento correto.
- Após um certo período para permitir que os auditores verifiquem o embaralhamento, Helios descriptografa todas as cédulas embaralhadas, fornece uma prova de descriptografia para cada uma e realiza uma contagem.
- Um auditor pode fazer o download de todos os dados das eleições e verificar o embaralhamento, descriptografia e registro. Se uma eleição é composta por mais de um turno, então cada turno é tratado como uma eleição separada: cada uma com seu próprio quadro de avisos, sua própria prova de embaralhamento e suas próprias descriptografia. Isso serve para limitar a possibilidade de re-identificar os eleitores que recebem cédulas longas, onde qualquer conjunto de respostas pode ser único no conjunto de cédulas expressas

4 SOLUÇÃO

Para descrever a solução, primeiramente apresentaremos as restrições e modificações implementadas. Em seguida, com o objetivo de facilitar a implantação, adaptamos o sistema para um sistema conteinerizado, essa tarefa foi realizada utilizando a tecnologia Docker, que se trata de um software que fornece uma camada de abstração e automação para virtualização do sistema operacional, criando contêineres independentes para executar dentro de uma única instância do sistema operacional, evitando a sobrecarga de manter máquinas virtuais [12]. Após conteireziar o sistema, realizamos o desenvolvimento da API REST utilizando os modelos já implementados pelo sistema Helios, e em paralelo foi desenvolvido

um cliente em Angular 8 para consumir os serviços oferecidos da API.

4.1 Restrições e Modificações

Para a realização deste trabalho foi definido que a API gerada terá, inicialmente, a eleição com apenas uma pergunta e terá como responsável pela apuração das eleições e pela forma como os usuários são autenticados, o próprio sistema Helios. No projeto original é possível realizar eleições com mais de uma pergunta e utilizar terceiros para fazer a apuração da eleição, além de se autenticar a partir de outros sistemas (LDap, Facebook, Gmail, etc). Essas funcionalidades poderão ser migradas no futuro seguindo este trabalho como base.

Foi necessário o desenvolvimento da funcionalidade de criação de usuários e eleitores a partir do próprio sistema, algo que estava para ser implementado pelo projeto Helios.

Também foi modificado a lógica entre eleitor e usuário. No projeto original um usuário do sistema e um eleitor eram tratados como entidades distintas, gerando assim duas formas de autenticação, agora o sistema entenderá que todo eleitor é também um usuário do sistema. Essa funcionalidade também estava pendente e foi implementado por satisfazer a realidade da solução proposta.

4.2 Conteinerização

Por ter tecnologias que não existe mais suporte, como a versão 1.8 do Django, e com o objetivo de isolar o sistema para evitar conflitos de pacotes, foi realizado a conteinerização do projeto Helios. Para essa tarefa, foi utilizado a tecnologia Docker e criado uma imagem com todos os requisitos do sistema para a criação dos containers, com isso garantimos que eventuais problemas que possam acontecer é necessariamente problema da aplicação e não interferência da máquina em que está sendo feito o desenvolvimento da aplicação, além de dar portabilidade ao projeto, podendo ser facilmente instalado em qualquer servidor que tenha Docker.

Para facilitar a iniciação do sistema Helios a partir de containers, foi utilizado a ferramenta docker-compose, que gerencia grupos de contêineres, podendo integrar sistemas de arquivos e redes entre eles. Para o projeto, foi necessário a utilização de cinco contêineres:

- Banco de dados
- Sistema Helios
- API
- Celery
- Redis

Foi decidido criar um container separado para a API para que os usuários possam escolher utilizar o sistema Helios com ou sem API Rest Os containers Celery e Redis são responsáveis por tratar partes da aplicação que utilize concorrência. O Celery é uma fila de tarefas assíncrona de código aberto, baseada na passagem de mensagens de forma distribuída e utiliza o Redis para a alocação dessas tarefas [10]. Redis é um armazenamento de estrutura de dados de chave-valor de código aberto que os armazenam na memória principal. [3]

A inicialização dos containers é de forma serial, primeiro é iniciado e configurado o banco de dados que utiliza a imagem oficial do postgres na versão 9.8.15. Em seguida é iniciado o projeto Helios que utiliza a imagem oficial do python na versão 2.7, instalado os pacotes necessários especificados no arquivo “requirements.txt”, criado um usuário administrador ao sistema Helios e iniciado a aplicação propriamente dita. Após o início do helios, é iniciado o container da API que têm seus pacotes descritos no arquivo “api-requirements.txt”, um desses pacotes é o Django Rest Framework. Por fim, são iniciados os containers responsáveis pela parte concorrente do sistema.

4.3 Arquitetura da API

A API é uma aplicação Python desenvolvida com o Django Rest Framework e que reaproveita a lógica já desenvolvida pelo projeto original ao adicionar o Helios e Helios Auth, fazendo com que os controladores da API façam chamadas aos modelos dessas aplicações, diminuindo a reescrita de código, assim como comportamentos fora do esperado pelo sistema original. Como é descrito na Figura 3, o desenvolvimento da API não teve acesso direto ao banco de dados, todas as solicitações são passadas para o sistema Helios, mesmo as modificações de autenticação descritas foram feitas nos modelos do sistema interno. Desta maneira, a API Rest é uma camada acima que abstrai o comportamento do projeto original.

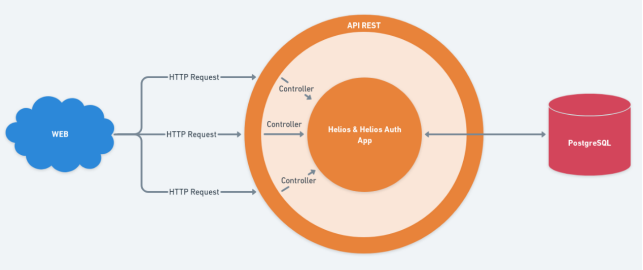


Figura 3: Arquitetura da API REST

A Figura 4 a seguir mostra como foi feito a organização da aplicação Python da API, o arquivo “urls.py” gerencia os endpoints do servidor e direciona qual classe irá capturar a requisição do cliente. Na pasta controllers, encontra-se os arquivos responsáveis pela lógica da API. No Django não existe o conceito de controller, deixado para que a View da arquitetura MVT (Model View Template) responsável por essa tarefa, com isso, o Django Rest Framework oferece várias modelos de View pré-configuradas para agilizar o processo de desenvolvimento. Utilizamos o modelo ViewAPI por deixar explícito quais métodos HTTP estará sendo utilizado nas lógicas, já que os métodos das classes do View API utiliza a mesma assinatura dos métodos HTTP e, desta forma, a aparência dos controllers fica parecidos com padrões de outras arquiteturas REST já conhecidas, como o do framework ExpressJS.

Enquanto na aplicação Helios as classes dos serviços são todas declaradas por padrão no arquivo models.py, na aplicação da API as classes foram declaradas nos arquivos da pasta “controllers” e dividido de acordo com o serviço oferecido, assim facilitando a manutenção do código no futuro. Foi desenvolvido dez serviços para a aplicação, cada um podendo contar com uma ou mais classes referentes a endpoints distintos, porém, com a mesma categoria de serviço.

Figura 4: Estrutura do código da API

Com os serviços desenvolvidos é possível criar eleições de uma única questão, podendo ela ser aberta para qualquer usuário conectado ao sistema apto a votação ou um grupo específico de eleitores descrito em um arquivo CSV, também é possível gerenciar apuradores, eleitores, usuários, realizar a computação de votos e divulgar resultados. Todos esses serviços são gerenciados por métodos HTTP, desta maneira, a integração com outros sistemas que faça uso da internet e do protocolo HTTP, como uma aplicação de votação que funcione em um dispositivo móvel, pode ser facilmente integralizado a API REST do Helios. Além de clientes, também é possível integrar outros servidores de maneira mais simples ao sistema, ajudando na evolução do código do Helios para uma versão mais recente do framework utilizado, por exemplo.

4.4 Cliente em Angular

Também foi desenvolvido um cliente em Angular em sua oitava versão e que faz uso da API REST do Helios. Angular é uma plataforma de aplicações web de código-fonte aberto e front-end baseado em TypeScript liderado pela equipe Angular e por uma comunidade de indivíduos e corporações [9]. Foi escolhido esse framework para ser o primeiro cliente da API por ser uma ferramenta popular, com um grande suporte, boa documentação e por ter sua lógica escrita em TypeScript completamente separada do código HTML, facilitando também manutenções futuras.

Para o cliente em Angular, foi necessário utilizar as bibliotecas javascript BigInt e de criptografia SHA256 e SHA1 que estava no repositório do Helios, para garantir que os cálculo matemáticos do cliente em JavaScript do Helios seriam mantidos na transição para o cliente em Angular. A Figura 5 a seguir mostra como foi feito a organização do cliente Angular, as classes escritas em JavaScript para o processo de cifragem das cédulas de votação foram traduzidas para TypeScript e se encontram na pasta "models", enquanto as telas e suas lógicas ficam armazenadas na pasta "components", a pasta "interceptor é responsável por adicionar o token de autenticação para todas as solicitações para a API, caso o usuário realize o login, e a pasta "services" é onde se encontra os métodos de solicitações de recursos da API divididas pelo tipo de serviço.

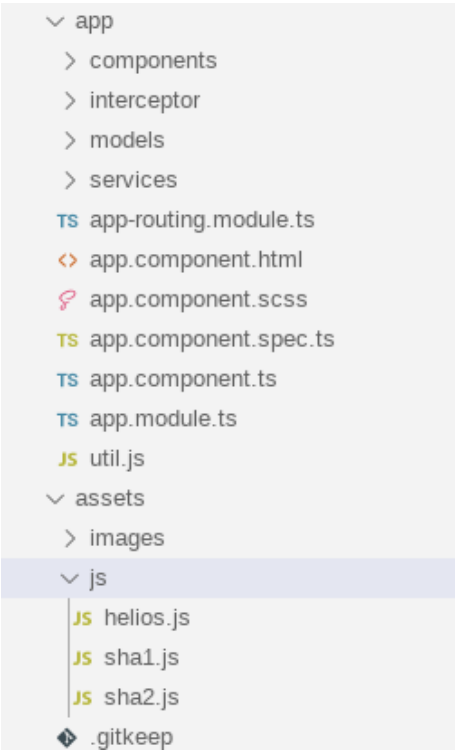
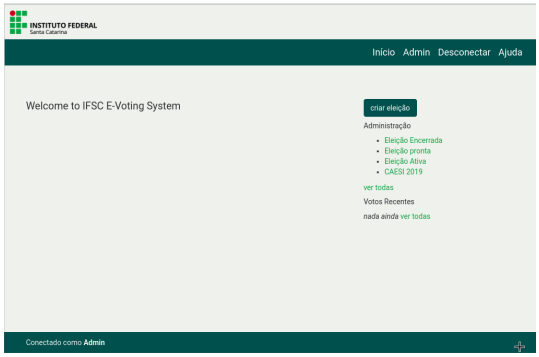


Figura 5: Estrutura do código do cliente Angular

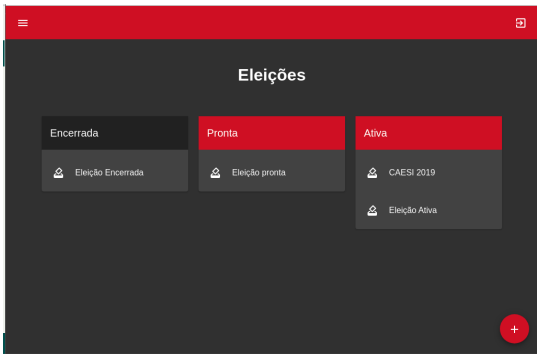
4.5 Comparação com a aplicação do IFSC

Ao desenvolver a aplicação que utiliza a API REST desenvolvida, surgiu a oportunidade de modificar o layout da aplicação de votação para que facilite a experiência do usuário, que no nosso caso, é o estudante do curso de ciência da computação da UFCG. A tela inicial da aplicação do IFSC (Figura 6a) ao se conectar como administrador do sistema, é possível visualizar as eleições já criadas e clicar no botão de criar uma nova eleição é visível, utilizando o mesmo princípio, nossa solução (Figura 6b) também cria três listas categorizando o estado em que a eleição se encontra:

- Ativa, eleição em andamento
- Pronta, eleição criada, porém ainda não foi lançada para que os usuários possam votar
- Encerrada, eleição em que a computação dos votos se encerrou

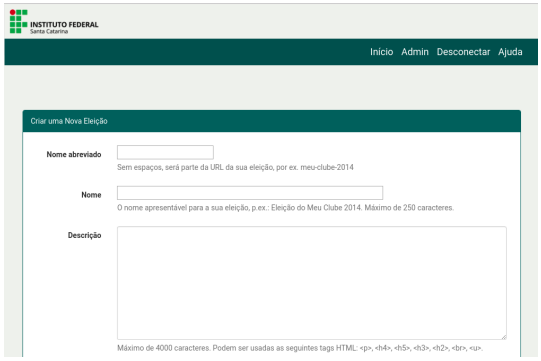


(a) IFSC

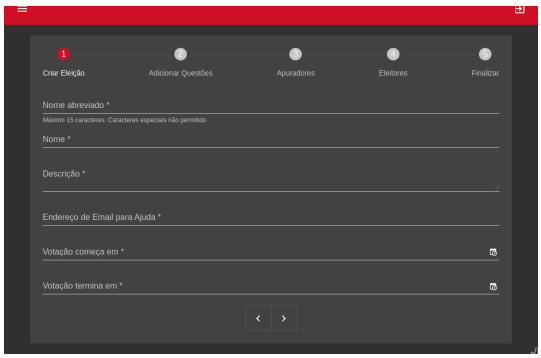


(b) Cliente Angular

Figura 6: Tela - Inicio



(a) IFSC



(b) Cliente Angular

Figura 7: Tela - Criar Eleição

Nas telas de criação de eleição, nossa solução (Figura 7b) procurou condensar todo o processo em cinco etapas:

- Criação da eleição, definição do nome, descrição e prazos da eleição
- Adicionar questões, definição da pergunta e das opções possíveis
- Apuradores, escolha se a aplicação utilizará o sistema helios ou alguma outra pessoa como apurador (no momento, o sistema só dá suporte para o sistema helios ser o apurador)
- Eleitores, etapa em que é definido se a eleição será acessível para qualquer pessoa que se conecta ao sistema ou será uma eleição privada, acessível apenas para uma lista de usuários disponibilizado por um arquivo csv.

Na aplicação do IFSC (Figura 7a) cada dessas etapas é tratado em telas distintas e o sistema informa ao usuário qual o próximo passo é necessário fazer para que a eleição seja de fato criada.

Na tela de visualização da eleição, optamos em dividir em quadros cada conteúdo de importância da eleição (Figura 8b), diferenciando por cores, para que o usuário identifique facilmente as informações que tenham mais prioridade, caso a eleição esteja em curso, adicionamos um cronômetro para que os eleitores saibam quanto tempo resta para o fim do período de votação da eleição. As mudanças nas telas da cabine de eleições e apuração de votos foram apenas estéticas, essencialmente, tem o mesmo comportamento da aplicação desenvolvida pelo IFSC.

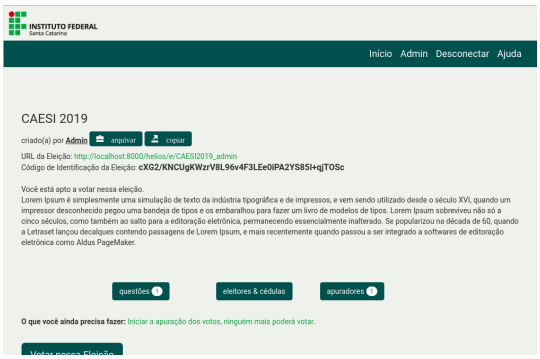
Ao término de uma eleição é adicionado mais dois quadros na tela de visualização da eleição: o quadro da escolha vencedora e o quadro da quantidade de cédulas (eleitores) que participou do pleito. (Figura 9b)

5 EXPERIÊNCIA

5.1 Processo de desenvolvimento

Antes de iniciar o processo de desenvolvimento, foi realizado a conteinerização da aplicação do Helios, além dos motivos descritos, a utilização de containers agiliza o processo de desenvolvimento de aplicações, já que a infraestrutura está toda descrita e configurada, uma vez a conteinerização feita, poderia desenvolver a aplicação em qualquer máquina que tenha o Docker instalado e com a certeza que o comportamento do sistema seria o mesmo. Após a conteinerização, foi feito o estudo do sistema Helios, para que entendêssemos a lógica de criação e manutenção de uma eleição, como também foi feito o estudo dos frameworks Django e Django REST Framework. Paralelamente aos estudos, foi feito o mapeamento das tabelas do banco de dados para que seja possível entender a relação entre os dados e como eram armazenados, essa etapa foi importante para que pudéssemos corrigir o armazenamento de senhas em formato texto para usuário cadastrados pelo tipo “password”.

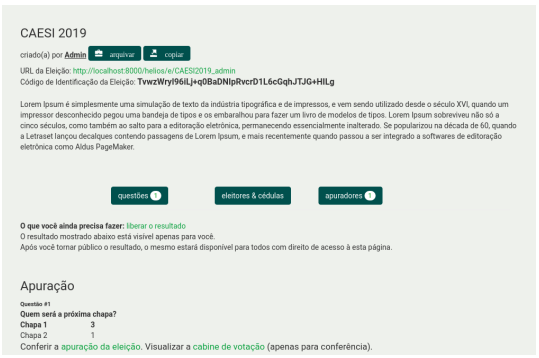
Em seguida, foi realizado o desenvolvimento dos serviços baseado no comportamento da aplicação do IFSC, ou seja, os serviços



(a) IFSC

(b) Cliente Angular

Figura 8: Tela - Visualizar Eleição



(a) IFSC

(b) Cliente Angular

Figura 9: Tela - Resultado da Eleição

foram, a princípio, desenvolvidos tendo como objetivo as ações previstas do sistema. Nesse processo, era analisado tanto comportamento as chamadas dos métodos no próprio código, como a experiência do usuário ao utilizar o sistema. Nesse processo, o uso do programa visual code studio aumentou a velocidade de analisar o código, já que essa ferramenta proporciona o uso de extensões que se conecta na virtualização dos containers, ajudando no processo de debug.

5.2 Desafios

Como esse trabalho é a evolução arquitetural de um projeto já existente, não foi possível escolher tecnologias que dominamos fazendo com que esse fator fosse fundamental nas dificuldades encontradas no desenvolvimento da solução. Além disso, o uso de versões antigas dessas tecnologias dificultou o acesso a documentações e integrações com outros recursos, como a escolha do middleware para a configuração do CORS (compartilhamento de recursos de origem cruzada), já que não foi possível usar a tecnologia mais recomendada nos dias atuais, assim como não foi possível utilizar a versão mais atual do Redis para as atividades de concorrência. Para superar essas dificuldades foi necessário fazer um trabalho de pesquisa de quais tecnologias são recomendadas usando como base a época que as versões das tecnologias que o Helios utiliza foram publicadas.

No desenvolvimento da aplicação em Angular 8 foi decidido fazer a tradução dos arquivos JavaScript do processo de cifragem das cédulas de votação para TypeScript, para que fosse possível modularizar e integrar esses métodos mais facilmente ao cliente da API, além de facilitar a leitura e manutenção do código no futuro. Porém, as bibliotecas que manuseiam BigInt disponíveis atualmente realiza cálculos distintos utilizados pela biblioteca do Helios, foi detectado que nas bibliotecas atuais o método “mod” (retorna o resto da divisão) pode retornar números negativos, enquanto que na biblioteca utilizada no Helios é retornado o módulo do resultado, ou seja, sempre retorna um valor positivo, e essa característica é explorada pelo Helios no cálculo da cifragem. Como não sabemos, a princípio, se o tratamento de números negativos do método “mod” foi uma implementação do próprio projeto Helios na biblioteca original e como essa diferença entre bibliotecas fez surgir a possibilidade de existir outros métodos que se comportem de forma distinta, foi decidido utilizar toda a biblioteca do Helios sobre BigInt e dos métodos de cifragem para o cliente em Angular, deixando a tradução para TypeScript apenas os métodos e processos que utilizam dessas bibliotecas de mais baixo nível da aplicação.

5.3 Trabalhos futuros

Os próximos passos desse trabalho é a abertura do projeto para que os estudantes do curso de ciência da computação da UFCG possam

contribuir e utilizar a API para futuras aplicações. Para esse processo de abertura é necessário a criação das seguintes funcionalidades:

- Documentação da API
- Documentação do processo eleitoral para a realidade do centro acadêmico do curso
- Divulgar a solução para os alunos do curso

Além da abertura, é necessário criar serviços para que a API REST do Helios dê suporte para eleições com mais de uma questão como também utilizar outras pessoas como apuradoras das eleições. Em relação ao cliente em Angular, será necessário refatorar as telas para melhorar a experiência do usuário.

Após a abertura para os estudantes poderem contribuir e ser feito os últimos ajustes da API para o funcionamento completo do Helios utilizando REST, poderá ser realizado a evolução das tecnologias da aplicação, como a utilização do framework Django em sua versão mais atual, como também poderá ser feito a evolução da proposta da aplicação em si, deixando de ser apenas uma ferramenta para votação e consulta online mas também para um ambiente que estimule a participação democrática da comunidade.

REFERÊNCIAS

[1] Ben Adida. 2008. Helios: Web-based Open-Audit Voting. Retrieved November 14, 2019 from https://www.usenix.org/legacy/events/sec08/tech/full_papers/adida/adida.pdf

[2] Ben Adida. 2019. Helios v3 Verification Specs. Retrieved November 14, 2019 from https://documentation.heliosvoting.org/verification-specs/helios-v3-verification-specs

[3] Amazon. 2019. O que é o Redis? Retrieved November 14, 2019 from https://aws.amazon.com/pt/elasticache/what-is-redis/

[4] Eduardo Barra Cordeiro. 2012. SOA – Arquitetura Orientada a Serviços. Retrieved November 14, 2019 from http://blog.iprocess.com.br/2012/10/soa-arquitetura-orientada-a-servicos/

[5] Edson. 2015. PostgreSQL Tutorial. Retrieved November 14, 2019 from https://www.devmedia.com.br/postgresql-tutorial/33025

[6] IFSC. 2019. Helios – Sistema de Votação On-line. Retrieved November 14, 2019 from https://dtic.ifsc.edu.br/sistemas/sistema-de-votacao-on-line-helios/

[7] Emerson Ribeiro de Mello Shirlei Aparecida de Chaves. 2014. O uso de um sistema de votação on-line para escolha do conselho universitário. Retrieved November 14, 2019 from https://dtic.ifsc.edu.br/files/chaves-sbseg14.pdf

[8] Daniel Szepielak. 2007. REST-Based Service Oriented Architecture for Dynamically Integrated Information Systems. Retrieved November 14, 2019 from https://pdfs.semanticscholar.org/5638/dc03c4341ca256e59c4c70d2df4082f10a0a.pdf

[9] Wikipedia. 2019. Angular (framework). Retrieved November 14, 2019 from https://pt.wikipedia.org/wiki/Angular_(framework)

[10] Wikipedia. 2019. Celery (software). Retrieved November 14, 2019 from https://en.wikipedia.org/wiki/Celery_(software)

[11] Wikipedia. 2019. Django (framework web). Retrieved November 14, 2019 from https://pt.wikipedia.org/wiki/Django_(framework_web)

[12] Wikipedia. 2019. Docker (software). Retrieved November 14, 2019 from https://pt.wikipedia.org/wiki/Docker_(software)

[13] Wikipedia. 2019. jQuery. Retrieved November 14, 2019 from https://pt.wikipedia.org/wiki/JQuery