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

ERI JOHNSON OLIVEIRA DA SILVA

UM SISTEMA PARA AUTOMATIZAR OS PROCESSOS GERENCIAIS DA OLIMPÍADA PARAIBANA DE INFORMÁTICA

CAMPINA GRANDE - PB

2019

ERI JOHNSON OLIVEIRA DA SILVA

UM SISTEMA PARA AUTOMATIZAR OS PROCESSOS GERENCIAIS DA OLIMPÍADA PARAIBANA DE INFORMÁTICA

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. Rohit Gheyi.

CAMPINA GRANDE - PB

2019

S586s Silva, Eri Johnson Oliveira da.
Um sistema para automatizar os processos gerenciais da Olimpíada Paraibana de Informática. / Eri Johnson Oliveira da Silva. – 2019.

9 f.

Orientador: Prof. Dr. Rohit Gheyi.
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. Olimpíada Paraibana de Informática. 2. Automação de processos gerenciais. 3. Sistema de informação. 4. Desenvolvimento de software. 5. Desenvolvimento de aplicação. 6. Framework Scrum. 7. Técnicas de Kaban. 8. Padrões de projeto. 9. Verificação de software. 10. Validação de software. I. Gheyi, Rohit. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

ERI JOHNSON OLIVEIRA DA SILVA

UM SISTEMA PARA AUTOMATIZAR OS PROCESSOS GERENCIAIS DA OLIMPÍADA PARAIBANA DE INFORMÁTICA

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. Rohit Gheyi
Orientador – UASC/CEEI/UFCG

Professor Dr. Franklin de Souza Ramalho
Examinador – UASC/CEEI/UFCG

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

Trabalho aprovado em: 25 de novembro 2019.

CAMPINA GRANDE - PB

Um sistema para automatizar os processos gerenciais da Olimpíada Paraibana de Informática

Trabalho de Conclusão de Curso

Eri Jonhson Oliveira da Silva
eri.silva@ccc.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil

RESUMO

As informações necessárias à organização da Olimpíada Paraibana de Informática (OPI) são processadas, em geral, via planilhas eletrônicas e documentos de texto. Esse tipo de atividade demanda tempo e é tendenciosa a erros, especialmente quando envolvem informações de centenas de participantes. Nesse sentido, este trabalho compreende o desenvolvimento dos módulos de geração de relatórios e envio de mensagens do OPI Interno: um sistema de informação para automatizar, tanto quanto possível, os processos gerenciais da OPI. Foram utilizadas técnicas para desenvolvimento de aplicações corporativas avançadas tais como entregar valor para os clientes; análise e projeto orientados a objetos; autogerencia ágil seguindo o framework scrum e técnicas kanban; padrões de projeto; reúso, verificação e validação de software. Através desse sistema os envolvidos com a OPI podem gerenciar escolas, competidores, ranking, seleção para outras fases da olimpíada e geração de certificados. O presente trabalho amplia o sistema para ser capaz de mostrar página de resultados, gerar relatórios e enviar mensagens via e-mail.

1 INTRODUÇÃO

A Olimpíada Paraibana de Informática (OPI) visa despertar nos alunos o interesse em computação, imprescindível na formação básica dos estudantes atualmente, através de atividades que envolvem desafios motivadores e competição saudável.

A edição de 2019 envolve cerca de 3.000 competidores, de 20 colégios diferentes, distribuídos em 5 cidades da Paraíba. Os desafios para gerenciar as atividades dessa edição contam com 35 pessoas na equipe, sendo 25 pessoas diretamente ligadas à geração e tratamento de informações relacionadas.

Produzir, processar, persistir, recuperar, e atualizar informações vinculadas com a OPI são atividades organizacionais mantidas por delegados dos colégios e organizadores da olimpíada. Essas atividades são realizadas por meio de considerável esforço manual para tratar documentos de textos e planilhas eletrônicas relacionadas e têm tendência a aumentar em número e complexidade.

Considere, por exemplo, o esforço em horas para o cadastro de alunos em nova competição, correção da prova de cada um deles, registro de notas em cada fase, definição de ranking, relação de classificados para 2ª fase da olimpíada, relação de premiados e participantes, divulgação e acompanhamento de cada uma dessas etapas, geração de certificados, e relatórios gerenciais.

Nesse sentido, utilizar um sistema de informação, feito sob medida para essas e outras atividades associadas, garante mais eficiência e simplicidade, economia de recursos humanos, menor margem a erros e possibilidade de envolver mais participantes, sejam competidores, delegados ou organizadores.

O sistema pode ser dividido em seus módulos fundamentais. O módulo de escolas permite que delegados cadastrem e mantenham informações sobre a escola que representam. O módulo de competição permite que organizadores e delegados cadastrem e atualizem informações dos competidores a partir de planilhas excel.

O módulo de ranking e seleção para outras fases permite que organizadores selecionem competidores para segunda fase da olimpíada automaticamente a partir de uma porcentagem escolhida. O módulo de geração de certificados permite que organizadores façam download de certificados de participação e premiação, tudo gerado automaticamente. Para processar planilhas *excel* e documentos *word* a API do *Apache POI* é utilizada.

O módulo de envio de mensagens permite que notícias importantes sejam encaminhadas e que lembretes sejam disparados automaticamente. Para compor este módulo o *Mailchimp* é utilizado. O módulo de geração de relatórios permite visualizações com análises exploratórias de dados sobre outras competições em informática, além da OPI. Para plotar as visualizações a ferramenta *Vega* é utilizada.

Nos projetos das disciplinas Desenvolvimento de Aplicações Corporativas Avançadas (DACA) e Projeto 1 (P1), o autor desenvolveu soluções parciais para alguns desses módulos, individualmente em DACA e em grupo em P1. O presente trabalho foca em aprimoramentos nos módulos desenvolvidos e no desenvolvimento dos módulos de resultados, envio de mensagens e geração de relatórios.

O presente documento está disposto da seguinte maneira. A seção *2 Trabalhos Relacionados* revela alguns trabalhos associados à OPI. A seção *3 Arquitetura e Projeto da Solução* se debruça sobre a arquitetura macro das soluções desenvolvidas neste trabalho e aproveita para esmiuçar suas partes, a saber, *3.1 Backend*, *3.2 Persistência*, *3.3 Envio de E-mails*, *3.4 Frontend*, *3.5 Visualização de Dados*, *3.6 Autenticação e Autorização com JWT*. Em seguida, na seção *4 Experiências e Lições Aprendidas*, há espaço para explanar as lições aprendidas, em alguns eixos, tais como, *4.1 Processo de desenvolvimento*, *4.2 Principais desafios e suas soluções*, *4.3 Limitações e Trabalhos Futuros*. Enfim, são feitos os *Agradecimentos* e as *Referências* são indicadas.

2 TRABALHOS RELACIONADOS

Tentar melhorar os processos gerenciais da OPI vai muito além do que testar capacidades de produzir software ou elaborar um trabalho de conclusão de curso. Significa tentar influenciar positivamente em uma cultura frutífera que envolve alunos em treinamento e competição saudável, além de acadêmicos em pesquisa, ensino e extensão.

O Projeto Olímpico é um projeto de extensão com objetivo de capacitar alunos, de escolas de ensino fundamental e médio, em programação. O autor teve oportunidade de participar deste projeto em escolas da rede pública de ensino e transmitir conceitos fundamentais de algoritmos a partir de jogos.

O trabalho de Marcelo Vitorino e Hugo Silva [3] acentua o impacto do Projeto Olímpico, quando mostra que uma fatia expressiva, mais de 40%, de alunos se prepara para ganhar olimpíadas na universidade/escola, sendo o Projeto Olímpico o principal encarregado por essa preparação.

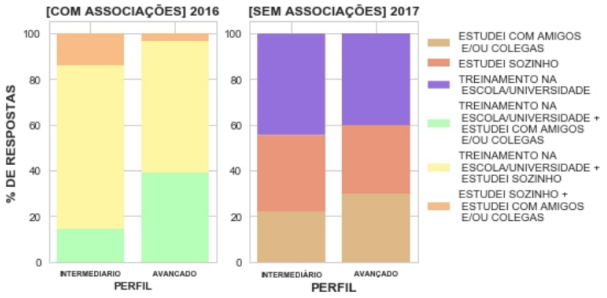

Figura 1: Formas como campeões tipicamente se preparam para Olimpíadas de Informática

Além disso, eles evidenciam que a sinergia natural dos atletas olímpicos (muito comparáveis aos Círculos Matemáticos Russos [2], onde alunos e professores estudavam matemática com mesmo entusiasmo que esportistas) é a principal força motriz que sustenta essa rica cultura de treinamento intenso e competição saudável entre a grande maioria dos participantes.

3 ARQUITETURA E PROJETO DA SOLUÇÃO

A arquitetura da solução é cliente-servidor. Nas soluções corporativas atuais, como explica Fiqueredo [4], tipicamente um servidor implementa as regras de negócio do produto e é responsável por persistir os dados importantes para o negócio. Em resumo, o servidor (implantado em uma máquina servidora) entende, manipula e extrai informações dos dados persistidos.

O cliente, por sua vez, comunica-se com o servidor para enviar ou receber dados necessários para os usuários do produto. Em resumo, o cliente (executando na máquina do usuário) é responsável por criar interfaces amigáveis para capturar informações dos usuários, comunicar-se com o servidor para entregar essas informações, bem como para resgatar informações quando os usuários solicitarem.

Segue um esquema visual dessas interações:

Na figura 2 é possível apreciar em detalhes a arquitetura da solução. Da esquerda para a direita temos a) os dispositivos dos

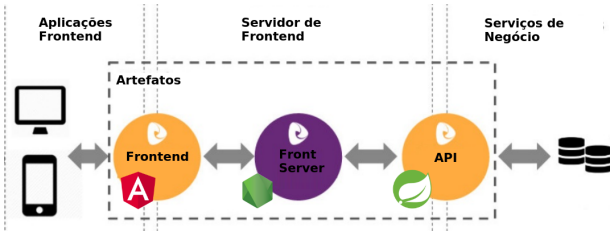

Figura 2: Arquitetura da Solução

usuários da solução; b) o frontend que serve de interface ao usuário; c) o servidor do frontend, que recebe, processa e responde às solicitações dos usuários; d) o servidor do backend (API/serviços), onde é implementada a lógica de negócio da solução; e e) a camada de persistência de dados, onde dados de todos os usuários estão armazenados.

No mais, a comunicação entre backend e frontend segue padrões REST [1]. Isso significa, resumidamente, acessar recursos a partir de Identificadores Únicos de Recursos (*Uniform Resource Identifiers - URIs*), utilizando HTTP, para garantir interface uniforme, desassociação de representação de conteúdo, para permitir acesso em vários formatos e comunicação sem estado, ou seja, não há gerência de sessão de usuários logados ou afins.

3.1 Backend

O backend é construído com componentes de software [8] com paradigma orientado a objetos [7], pois é o padrão para desenvolvimento de aplicações corporativas modernas. A granularidade de componentes, nesse caso, é definida de acordo com os diferentes módulos do OPI Interno, ou seja, escola, competidor, olimpíada e afins são representados como componentes separados do sistema.

A tecnologia escolhida para construir o backend é Spring Boot, utilizando a linguagem Java, pois facilita a construção rápida de apps com base em Spring e outros frameworks. O Spring Framework, por sua vez, fornece um modelo abrangente de programação e configuração para aplicações corporativas modernas para qualquer tipo de plataforma de implementação. A maturidade no mercado e a confiabilidade associada são os pontos mais relevantes.

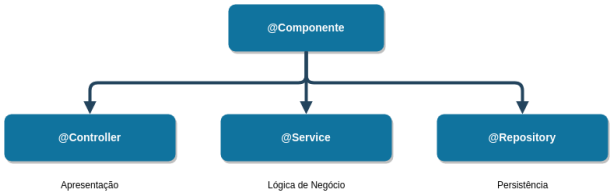

Figura 3: Típico componente do backend

A figura 3 mostra, da esquerda para direita, a) controller é a camada que expõe interface para manipulação de dados; b) service é camada de negócio, ou seja, onde as regras necessárias para tratar problemas dos clientes são implementadas; c) repository é a camada de persistência, ou seja, a camada responsável por manipular e garantir que os dados estão salvos e em perfeito estado.

https://spring.io/projects/spring-boot, acessado em 20 de Novembro de 2019

A figura 3 está simplificada, mas, na prática, ainda são usados a) models, que representa entidades do componente; b) exceções, que representam erros conhecidos que podem ocorrer; e c) objetos de transferência de dados (DTOs), que são usados para entrada e saída de dados no backend. Essa arquitetura é importante, pois preserva o princípio de responsabilidade única [9], garante solução fácil para os problemas, manutenibilidade e reúso.

3.2 Persistência

A representação de entidades e a camada de persitência de dados merecem um tópico a parte, tendo em vista o esforço, em termos de software, para conseguir solução elegante e eficaz com um mínimo esforço de implementação, como a figura 4 evidencia.

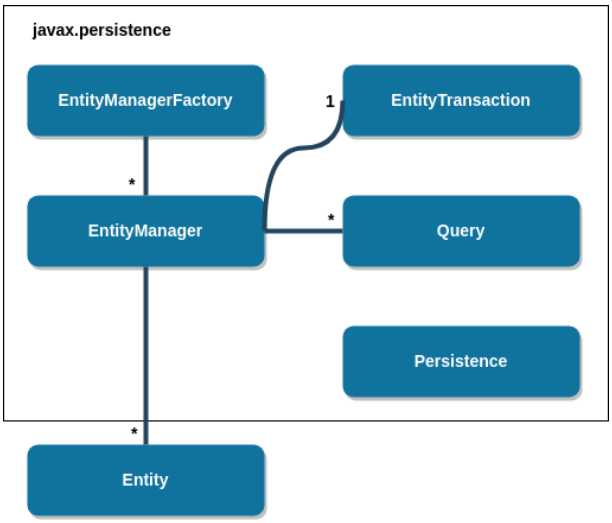

Figura 4: Típico componente de persistência de dados

Na figura 4, de baixo para cima, vê-se a entidade. Uma entidade é a representação primária de algum objeto do mundo real. Tipicamente é formada por atributos, que compõem seu domínio, e métodos, que manipulam esses atributos de forma coerente com as necessidades da solução.

No entanto, para resolver problemas reais em software, uma entidade deve ser criada, manipulada, persistida/transferida e destruída sempre que usuários requisitarem. Nesse caso, utilizou-se a Java Persistence API , pois ela provê construção, gerência/manipulação, busca e persistência de dados das entidades a partir de poucas configurações.

3.3 Envio de E-mails

O envio de e-mails, atividade geralmente simples, se mostrou um desafio em tanto. A começar pela escolha, visto três boas APIs para uso comercial elencadas, a saber, *SendGrid*, *Mailchimp* e *Amazon Simple Email Service (SES)*. Todas as opções têm nível gratuito suficiente para o presente trabalho.

Considerando quantidade de e-mails para uso gratuito, o *Mailchimp* se destaca com seus 10000 e-mails gratuitos por mês (contra 1000 do *SES* e 100 diários do *SendGrid*), além de contar com a melhor ferramenta para criação, verificação de métricas e envio de e-mails, seja pelo navegador ou via API externa. Por essas razões, *Mailchimp foi* a ferramenta escolhida para envio de e-mails no escopo atual.

A implementação dos serviços para envio de e-mails é feita através de chamadas à *Mailchimp API 3.0*, implementada de acordo com padrão REST, como mostra o esquema da figura 5.

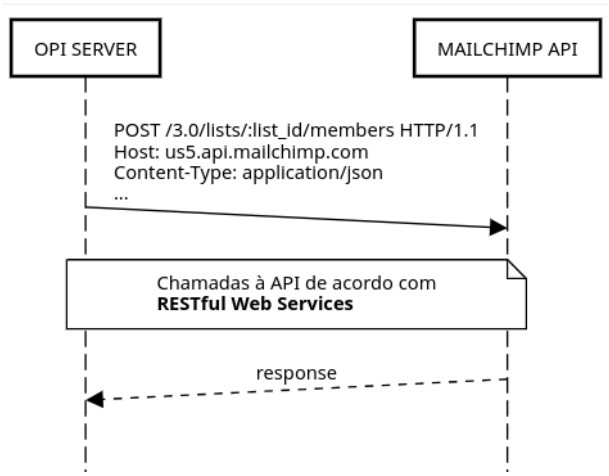

Figura 5: Típica interação entre servidor da OPI e *Mailchimp API* para envio de e-mails

A figura 5 mostra dois componentes, o servidor backend do OPI Interno e a API do *Mailchimp*. Tipicamente, a) o OPI interno requisita uso de algum serviço, tal como cadastrar novos membros à lista de e-mails, b) o *Mailchimp* avalia requisição a partir de tokens no cabeçalho da requisição (visto que não há persistência de estado entre as partes), processa e, se tudo der certo, devolve resposta favorável.

Na figura 6 é possível apreciar o resultado dessa ação com relação ao cadastro de novos Delegados ao sistema, por exemplo.

3.4 Frontend

Historicamente, construir interface de usuário para web consistia em gerar páginas web com html, css e javascript apenas [5]. No entanto, nos últimos anos, frameworks como Angular diminuem o esforço para produzir, testar e manter webapps, tanto que tornaram-se padrões da indústria.

Por essa razão Angular é a tecnologia escolhida para produzir o frontend do OPI Interno. Além disso, Angular, através de sua interface de linha de comando, provê definição automática de módulos, que são estruturados de acordo com a figura 7.

A figura 7 mostra, ao centro, a) componente, estrutura de lógica do frontend; b) template, estrutura de experiência do usuário. Juntos, um componente e um modelo definem uma visão angular. Um decorador, em uma classe de componente, adiciona metadados, incluindo um ponteiro para o modelo associado. Diretivas e

https://docs.spring.io/spring-data/jpa/docs/current/reference/html, acessado em 20 de Novembro de 2019

https://mailchimp.com/developer/guides/get-started-with-mailchimp-api-3, acessado em 20 de Novembro de 2019

https://angular.io/guide/architecture, acessado em 20 de Novembro de 2019

Figura 6: Exemplo de e-mail enviado pelo OPI Interno

marcações de ligação no modelo de um componente modificam as exibições com base nos dados e na lógica associada. Enfim, o injetor de dependência fornece serviços a um componente, como o serviço de roteador, que permite definir a navegação entre visualizações.

O Angular é usado para estrutura, tratamento de enventos, comunicação, roteamento e outras técnicas e padrões típicos de frontends modernos. No entanto, para compor a interface de usuário, ou seja, o que o usuário vê e como ele interage com elementos na tela, é preciso uma infraestrutura de componentes de interface e design. Para tanto, o OPI Interno conta com o Angular Material [1], o conjunto de princípios de bom design Google.

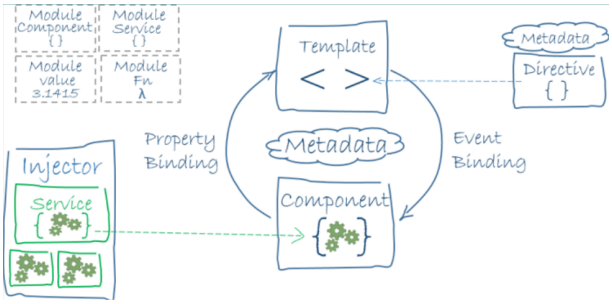

Figura 7: Típico componente do frontend

Na figura 8 é possível apreciar interface de usuário para a tarefa de resultados, anteriormente PDFs feitos à mão, produzida durante a execução deste projeto, que compõe o webapp do OPI Interno.

3.5 Visualização de Dados

A visualização de dados é a representação gráfica de informações e dados. Com elementos visuais, como tabelas, gráficos e mapas, as ferramentas de visualização de dados fornecem uma maneira acessível de ver e entender tendências, outliers e padrões nos dados.

[1] https://material.angular.io, acessado em 20 de Novembro de 2019

Figura 8: Tela de Resultados - anteriormente PDF feito à mão

Considerando a sólida cultura de grupos de estudo para resolução de problemas e otimização, seja pelo esforço do Projeto Olímpico ou por disciplinas como Algoritmos Avançados, aliada ao alto nível da OPI, é comum que os competidores da região conquistem prêmios em outras olimpíadas de programação no país e no mundo. Dessa forma, o OPI Interno dispõe uma seção para apresentar visualizações sobre essas outras conquistas.

Para tanto, é utilizada uma gramática de alto nível de gráficos interativos. A vega-lite fornece uma sintaxe JSON concisa para gerar visualizações rapidamente e, por essa razão, é utilizada no OPI Interno. Apesar de simples para configurar, a ferramenta é poderosa e provê plot de histogramas, gráficos de área, nuvem de bolhas, cartogramas, mapa de distribuição de pontos, redes e muito mais.

Na figura 9 é possível apreciar uma típica visualização da geração de relatórios, anteriormente planilhas excel, produzida durante a execução deste projeto, provida pelo OPI Interno.

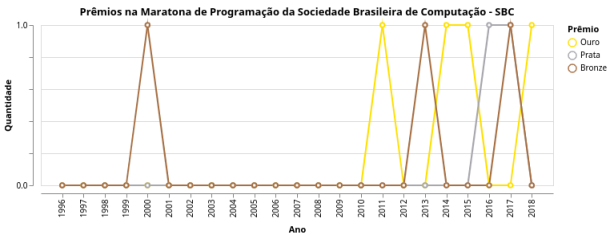

Figura 9: Visualização - anteriormente excel feito à mão

3.6 Autenticação e Autorização com JWT

Historicamente implementar segurança para usuário logado era feito pela técnica de sessão, ou seja, desde o login até o logout um objeto era persistido no servidor e acessado a cada interação do usuário para validá-lo. Atualmente, tipicamente a interface de usuário tem servidor separado do servidor de serviços do webapp, como é o caso do OPI Interno, vide figura 2. Neste caso, autorização e validação devem ser feitas a partir de outras técnicas.

[2] https://vega.github.io/vega-lite, acessado em 20 de Novembro de 2019

Então, para o OPI Interno, utilizou-se o JSON Web Token (JWT). Tecnicamente JWT é o uso do método RFC 7519, definido e revisado pela Força-Tarefa de Engenharia da Internet (*Internet Engineering Task Force - IETF*), ou seja, é um padrão aberto da industria para representar reivindicações de forma segura entre duas partes na web, como visto na figura 10.

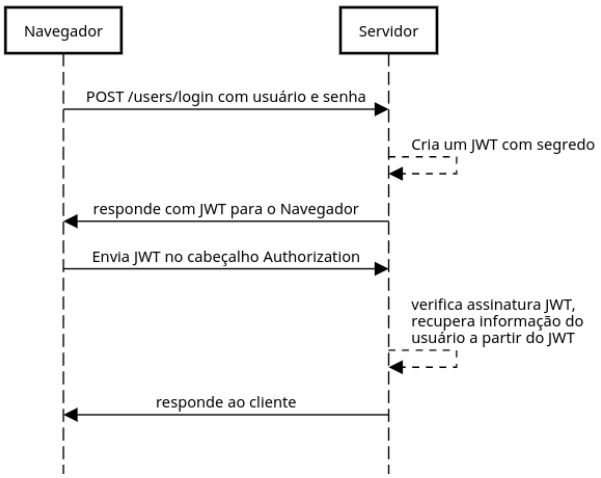

Figura 10: Uso de JWT para comunicação entre as partes

A figura 10 é um esquema das reinvidicações típicas entre as partes utilizando JWT: 1) cliente faz login através de seu nome de usuário e senha; 2) servidor cria JWT com chave secreta; 3) JWT retorna como resposta e é armazenada no dispositivo do cliente; 4) usuário fará novas requisições enviando JWT no cabeçalho da requisição; 5) servidor autoriza usuário a partir de informações do JWT; e 6) servidor responde adequadamente ao cliente.

Como o OPI Interno é construído com Spring, então utiliza-se Spring Security para blindar a solução e tratar requisições a partir de filtro, ou seja, todas as requisições são escrutinadas. Tipicamente, o JWT é decodificado, informações pertinentes são avaliadas e verifica-se a autoridade do cliente para usar determinado serviço de interesse, visto que cada serviço é unicamente identificado. Resumindo, há controle fino sobre quem pode acessar cada serviço do backend.

4 EXPERIÊNCIA E LIÇÕES APRENDIDAS

4.1 Processo de desenvolvimento

A primeira, e talvez mais importante, lição aprendida foi escolher a delimitação do tema deste trabalho. Como o OPI Interno é complexo o suficiente para praticar vários aspectos do conhecimento geral e específico do curso, um desafio foi escolhido: correção automática de gabaritos, necessários nas categorias Iniciação 1 e 2 da OPI. Na literatura de percepção computacional esse problema é conhecido como Reconhecimento Óptico de Marcas (*Optical Mark Recognition - OMR*) [10].

O problema ocorrido diz respeito a falta de instrução do autor em percepção computacional. Mesmo com orientação, essa não é tarefa trivial para ser implementada, integrada em uma API e utilizada em um sistema de informação em tempo. Nesse sentido, cerca de um terço do semestre foi utilizado para tentar entregar algo plausível, o que não foi possível.

Por outro lado, a trilha de desenvolvimento do curso envolve disciplinas obrigatórias e optativas, tais como Programação 1 e 2, Projeto de Software, Análise de Sistemas, Projeto 1, Engenharia de Software, Desenvolvimento de Aplicações Corporativas Avançadas e outras. Além de cursar a maioria delas, o autor participou de projetos de capacitação relacionados com desenvolvimento de sistemas de informação, então o foco foi ajustado para produzir algo de acordo com essas competências.

Dessa forma, o processo de desenvolvimento, de forma geral, seguiu orientações do curso e dos projetos de capacitação, a saber, levantamento de requisitos, análise e projeto da solução, implementação, verificação e validação de artefatos de software e afins. No detalhe, a correta organização dessas fases gera produtos de software, tais como o webapp OPI Interno.

No entanto, tendo em vista experiência em grupo, é comum definir interfaces robustas e heterogêneas, construir artefatos separadamente e, apenas depois, integrá-los. Não há nada de errado com essa abordagem, aliás, as equipes que utilizam metodologia ágil desenvolvem de acordo com essa estratégia de dividir para conquistar, e isso garante entregas rápidas, completas e com qualidade.

Nesse sentido, tarefas do backend e do frontend foram implementadas com esse *mindset*. O problema dessa abordagem, principalmente em trabalhos individuais, é que a integração geralmente é o ponto focal de problemas no desenvolvimento de software. Em equipe, há muitas mãos e muitos cérebros ajustando pequenos problemas corriqueiros. Com apenas um indivíduo desenvolvendo, há mais chances para erros e estouro de prazos.

Nas palavras do coorientador deste trabalho, *"... isto é seguir na contramão das recentes descobertas da engenharia de software. Geralmente escolhemos uma fina, mas profunda, questão e trabalhamos nela em todas as frentes, até à entrega. Isso é uma abordagem vertical. Trabalhar nas histórias principais em paralelo e deixar a integração por último é uma abordagem horizontal sujeita a mais riscos"*.

Em resumo, outra lição importante aprendida é sobre ter mais cuidado ao trabalhar individualmente em tarefas envolvendo vários artefatos, principalmente os que necessitam de integração. Nesses casos, é preciso dividir com mais zelo as histórias, produzir cada uma delas de ponta-a-ponta e, então, repetir o processo para as demais.

4.2 Principais desafios e suas soluções

O principal desafio técnico apresentado foi combinar envio de e-mails automaticamente, serviço de e-mail gratuito do Gmail e serviço de implatação de sistema gratuito do Heroku. Mesmo após muita pesquisa e teste, seja em documentações ofiais, ou guias da comunidade, tornou-se impossível organizar esses artefatos de software para operarem em conjunto.

Mais precisamente, para que a conta pudesse ser utilizada a partir do OPI Interno, as cofigurações de segurança do Gmail precisaram

https://tools.ietf.org/html/rfc7519, acessado em 20 de Novembro de 2019
https://jwt.io, acessado em 20 de Novembro de 2019
https://spring.io/projects/spring-security, acessado em 20 de Novembro de 2019

https://www.heroku.com/pricing, acessado em 20 de Novembro de 2019

ser alteradas. O ápice desse questão foi perceber que, com cerca de 3 semanas sem fazer uso do SMTP do Gmail, a conta volta às configurações originais. Isso gera problema de indisponibilidade de serviço até que a configuração para acesso seja refeita.

Dessa forma, o problema escalou para implementar o próprio *SMTP* ou utilizar outro serviço além do Gmail para esse fim. Nesse último caso, há algumas opções interessantes, tais como *SendGrid*, *Mailchimp* ou *Amazon Simple Email Service (SES)*. Todas as opções têm nível gratuito suficiente para o presente trabalho.

Considerando os recursos limitados da solução gratuita do *Heroku* e a necessidade de utilizar outro serviço de e-mails, a solução mais interessante para implantação e serviço de e-mails do OPI Interno foi utilizar os 12 meses de uso gratuito do serviço de computação da AWS, o *Amazon Elastic Compute Cloud (Amazon EC2)*, e o *Mailchimp* por oferecer até 10000 e-mails por mês e 2000 membros na lista gratuitamente.

Essa decisão, aliás, foi mais interessante para praticar outros conhecimentos do curso, considerando que o EC2 é uma máquina virtual na nuvem que deve ser completamente configurada para suportar os artefatos de software do sistema. Ou seja, foi preciso definir sistema operacional linux; instalar dependências, tais como java, maven, node, npm, postgresql; ajustar requisitos de rede, tais como abrir portas para acesso externo; utilizar SSH para acessar e configurar a máquina remotamente e afins.

4.3 Limitações e Trabalhos Futuros

É evidente que o OPI Interno precisa melhorar. A lista de competidores de determinada escola, por exemplo, pode ser mais funcional, pois, no momento, as operações são feitas exclusivamente pelo envio de planilhas, mas é importante também contar com o modo convencional de criar, editar e atualizar competidores via páginas no frontend.

Outro requisito do sistema ainda não implementado é mostrar uma página, ou modal, com resumo de um competidor específico, que deve aparecer ao clicar em seu nome na página de resultados, por exemplo. Nesse caso, será possível apreciar todas as conquistas individuais de um competidor de interesse.

Além de funcionalidades que os organizadores reportaram com o uso, é importante aumentar cobertura de testes de unidade e integração para, no mínimo, 75% de cobertura, pois é cobertura miníma esperada com base em projetos anteriores e tendência de cobertura dos artefatos do Google [6], por exemplo.

Caso o produto mínimo viável sugerido até aqui seja aceito para figurar como solução para o OPI Interno, é importante fazer testes alpha, com foco em escutar os usuários para melhorar suas experiências. Então, o trabalho de melhoria contínua focará em corrigir esses bugs reportados e implementar mais funcionalidades.

Por último e mais importante, será preciso migrar maioria dos dados das escolas e competidores a partir das planilhas, documentos de textos, PDFs e afins necessários à OPI. O parser e adequação ao OPI Interno certamente será o maior desafio dos listados até então.

https://sendgrid.com/pricing, acessado em 20 de Novembro de 2019
https://mailchimp.com/pricing, acessado em 20 de Novembro de 2019
https://aws.amazon.com/pt/ses/pricing, acessado em 20 de Novembro de 2019

AGRADECIMENTOS

Aos meus pais e familiares, que me apoiam desde sempre. Aos funcionários, professores e colegas do curso de Ciência da Computação da UFCG. Em especial, ao professor Rohit, por atender às insistências dos alunos e, assim, fazer do Projeto Olímpico e da OPI o que são hoje.

REFERÊNCIAS

[1] Oracle and/or its affiliates. 2013. What Are RESTful Web Services? Retrieved 19 de Novembro de 2019 from https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html
[2] Sergey Genkin Dmitri Fomin and Ilia Itenberg. 2010. *Círculos Matemáticos* (1 ed.). IMPA, Rio de Janeiro.
[3] Marcelo Vitorino e Hugo Silva e Livia Sampaio e Rohit Gheyi. 2018. Perfil dos premiados em Olimpíadas de Informática e sua influência sobre a Educação em Computação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)* 29, 1 (2018), 228. https://doi.org/10.5753/cbie.sbie.2018.228
[4] Eduardo Figueiredo. 2019. Padrões Arquiteturais. Retrieved 20 de Novembro de 2019 from https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/padroes-arquiteturais-sm_v01.pdf
[5] Dave Gavigan. 2018. The History of Angular. Retrieved 19 de Novembro de 2019 from https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7
[6] Marko Ivankovic. 2019. Code Coverage at Google. https://doi.org/10.1145/3338906.3340459
[7] Craig Larman. 2004. *Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientados a Objetos e ao Processo Unificado* (2 ed.). Bookman, Porto Alegre.
[8] Renato Figueiró Maia. 2004. *Um Framework para Adaptação Dinâmica de Sistemas Baseados em Componentes Distribuídos.* Master's thesis. Pontificia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brasil.
[9] Ester Andrade Ribeiro and Fernando Tiosso. 2019. PRINCPIO DA RESPONSABILIDADE NICA APLICADA AO DESENVOLVIMENTO DE APLICAES. *Revista Interface Tecnolgica* 16, 1 (jun. 2019), 278–290. https://revista.fatectq.edu.br/index.php/interfacetecnologica/article/view/625
[10] Shri Paramhans Education Trust (Ed.). 2015. Auto Evaluation of OMR Answer Sheets Using Mobile Application. *IRJMSI* 6, 3 (2015).