Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

CPqs Abertas: Dando continuidade ao desenvolvimento de um sistema web modular voltado para a exposição das produções acadêmicas da USP

Mohamad Hussein Rkein, Rafael Rodrigues Vieira dos Santos

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisor: Prof. Dr. Alfredo Goldman Vel Lejbman

São Paulo
2022

Resumo

Mohamad Hussein Rkein, Rafael Rodrigues Vieira dos Santos. **CPqs Abertas: Dando continuidade ao desenvolvimento de um sistema web modular voltado para a exposição das produções acadêmicas da USP**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2022.

Este trabalho tem como objetivo apresentar o estado do projeto CPqs Abertas no início do ano de 2022 e exibir as atividades desenvolvidas para melhoria e extensão do código. O trabalho consistiu na criação de páginas novas para o projeto, por meio do lançamento do IME Aberto e do desenvolvimento do Hub CPqs Abertas - plataforma unificada para divulgação do projeto e inscrição de novos institutos -, e também em melhorias no Ciclo de Vida de Desenvolvimento de Software (CVDS), por meio da automatização de processos e da redução do tempo e esforço necessários entre a publicação de uma mudança de código e a visualização dessa mudança nos sistemas em produção. Dentre essas melhorias, houve a migração da hospedagem do CPqs Abertas para servidores sob posse da equipe, e foi implementado um sistema de *deploy* contínuo das mudanças para tais servidores, padrão seguido por todo projeto moderno de grandes empresas no ramo de software. Essa modernização do projeto tornou possível a rápida implementação de correções e de funcionalidades novas para todos os sistemas CPqs Abertas. É esperado que essas melhorias venham a ser aproveitadas por desenvolvedores futuros do projeto ao estendê-lo para o restante das unidades da USP interessadas, possibilitando um processo de criação eficiente e fluido para desenvolvedores e clientes.

Palavras-chave: CPqs Abertas. Ciclo de Vida de Desenvolvimento de Software. Deploy Contínuo. Automação. Produções Acadêmicas.

Abstract

Mohamad Hussein Rkein, Rafael Rodrigues Vieira dos Santos. **CPqs Abertas: Continuing the development of a modular web system aimed at exposing USP academic productions**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2022.

This paper aims to present the status of CPqs Abertas project at the beginning of 2022 and display the activities conducted for improvement and extension of the system. The work consisted of creating new pages for the project, through the launching of IME Aberto page and the development of Hub CPqs Abertas - unified platform for exposing the project and registering new institutes -, and also improvements in the Software Development Life Cycle (SLDC), through processes automation and reduction of necessary time and effort between publishing a code change and seeing that change in the production systems. Among these improvements, there was the migration of CPqs Abertas hosting to servers under possession of the team, and a system of continuous deploy was implemented for these servers, methodology followed by every modern project of large companies in the field of software development. This modernization of the project made it possible to quickly implement and see the results of corrections and new functionalities for all CPqs Abertas systems. It is expected that these improvements will be used by future developers of the project when extending it to the rest of the interested institutes of USP, allowing an efficient and fluid creation process for developers and customers.

Keywords: CPqs Abertas. Software Development Life Cycle. CI CD. Automation. Academic Productions.

Lista de abreviaturas

API	Interface de Programação de Aplicação (*Application Programming Interface*)
AWS	Amazon Web Services
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CPq	Comissão de Pesquisa
FAU	Faculdade de Arquitetura e Urbanismo
FEA-RP	Faculdade de Economia, Administração e Contabilidade de Ribeirão Preto
HTTP	Protocolo de Transferência de Hipertexto (*Hypertext Transfer Protocol*)
IME	Instituto de Matemática e Estatística
PRPI	Pró-Reitoria de Pesquisa e Inovação
SQL	Linguagem de Consulta Estruturada (*Structured Query Language*)
SSH	Secure Shell
TPS	Transações por segundo
URL	Localizador Uniforme de Recursos (*Uniform Resource Locator*)
USP	Universidade de São Paulo

Lista de figuras

Lista de programas

Sumário

Capítulo 1

Introdução

Os principais fins das universidades públicas são promover, desenvolver e estender o conhecimento à sociedade por meio do ensino, da pesquisa e da extensão. A ciência brasileira tem crescido sistematicamente, e as universidades são o principal motor a tracionar esse desenvolvimento, o que demonstra certa proximidade aos fins estabelecidos. Segundo relatório da empresa Clarivate Analytics Science Group (2019), o país ocupa a 13ª posição a nível mundial em número de trabalhos científicos publicados, mesmo após cortes constantes e expressivos no orçamento do Ministério da Ciência durante a última década.

PESQUISA FAPESP 20 ANOS

A expansão em números

Em duas décadas, parâmetros da ciência brasileira evoluíram de modo consistente

Fabrício Marques

Edição 284
out. 2019

Cientometria
Educação
Financiamento
Pol. Públicas

Desde 1999, quando *Pesquisa FAPESP* começou a circular, o perfil da ciência brasileira passou por uma grande transformação. A produção científica nacional cresceu mais do que cinco vezes: o número de artigos de pesquisadores do Brasil publicados em revistas indexadas na base Scopus, que estava na casa dos 13,5 mil no final dos anos 1990, alcançou 74 mil em 2018, levando o país do 18º para o 13º lugar entre as nações que mais geram conhecimento na forma de *papers*.

Figura 1.1: *Notícia FAPESP - Crescimento da ciência no Brasil.*

Entretanto, apesar do cenário positivo que abrange a produção de ciência, vive-se um panorama insatisfatório quanto à disseminação desse conhecimento: pouco do que se é produzido nas universidades alcança aqueles que vivem fora dos muros dessas instituições. Durante pesquisa do Instituto Nacional de Ciência e Tecnologia em Comunicação Pública

da Ciência e Tecnologia (INCT-CPCT, 2021), 87% dos jovens brasileiros não conseguiram citar uma única instituição nacional de pesquisa e 93% não souberam citar o nome de algum cientista brasileiro.

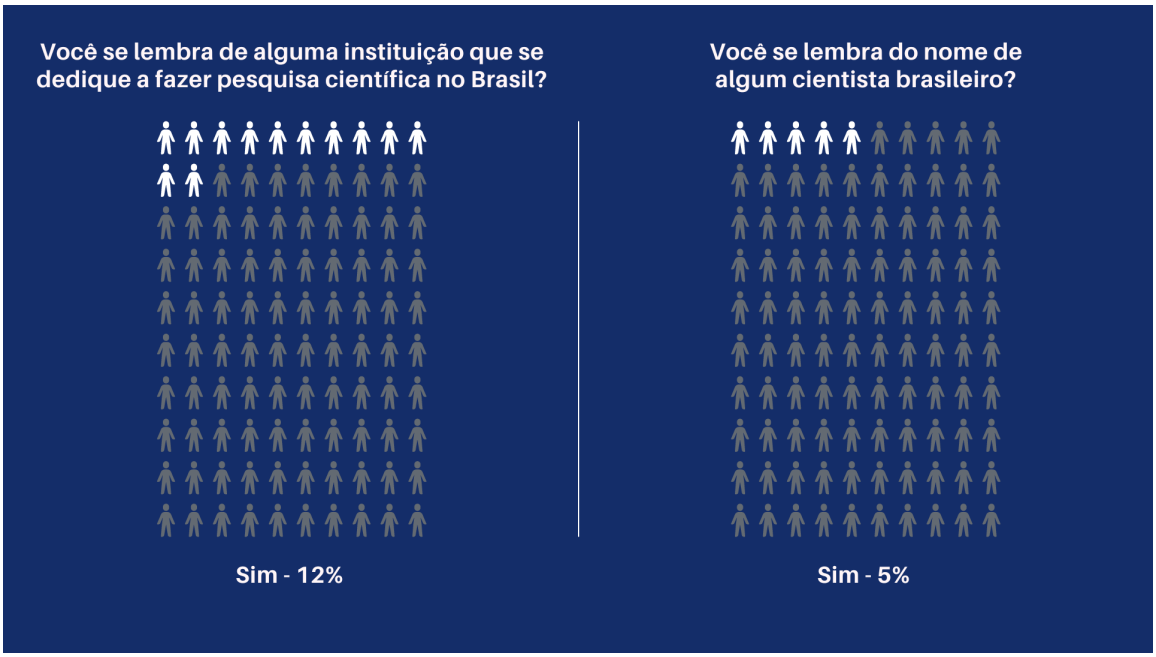

Figura 1.2: *O que os jovens brasileiros pensam da ciência e da tecnologia? - Pesquisa INCT-CPCT*

No andar dessa esteira, o projeto CPqs Abertas surge como uma alternativa moderna com grande potencial de apoiar e intensificar a divulgação do trabalho acadêmico desenvolvido na Universidade de São Paulo.

1.1 O que é o CPqs Abertas?

O projeto FAU Aberta, desenvolvido por meio de uma parceria de docentes da Faculdade de Arquitetura e Urbanismo com a disciplina MAC 5716 - Laboratório de Programação Extrema, coordenada pelo professor Alfredo Goldman no IME, é o ponto de partida e inspiração inicial do CPqs Abertas. No segundo semestre de 2020, após o lançamento da página web do FAU Aberta e sua apresentação para a Pró-reitoria de Pesquisa e Inovação decidiu-se estender a iniciativa para outras unidades da USP.

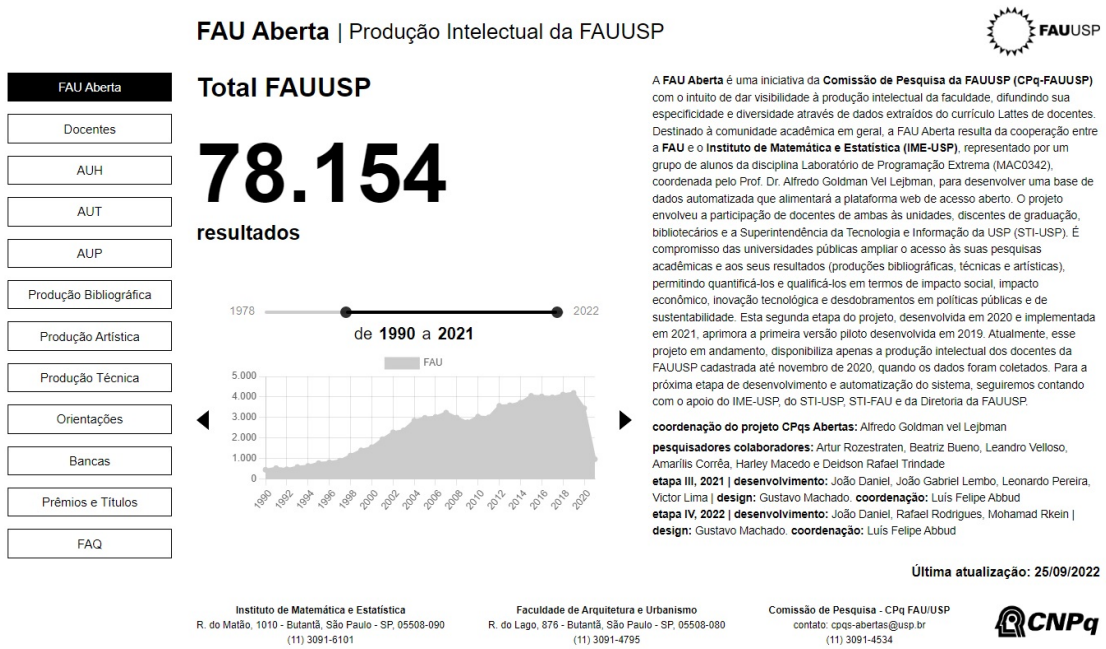

Figura 1.3: *FAU Aberta*

A partir daí, um trabalho conjunto entre docentes e discentes do IME e da FAU foi sendo desenvolvido. Em 2019, participaram do desenvolvimento do software por trás do projeto os alunos César Fernandes, Leonardo Aguilar, Larissa Sala, Mateus dos Anjos, Matheus Cunha, Nathalia Borin, Pedro Santos, Victor Batistella. Em 2020, participaram do desenvolvimento os alunos Kaique Komata, Jean Pereira, Luciana Marques, Priscila Lima e o responsável pelo design foi Luís Felipe Abbud. Em 2021, João Daniel, João Gabriel Lembo, Leonardo Pereira, Victor Lima contribuíram no desenvolvimento e Gustavo Machado no design, com coordenação de Luís Felipe Abbud. O professor Alfredo manteve-se como coordenador do projeto.

1.2 Objetivos

O objetivo deste trabalho é de dar continuidade ao que foi iniciado com o FAU Aberta e o CPqs Abertas, expandindo esse meio de divulgação para mais institutos da USP; criar uma plataforma unificada para o acesso às páginas do projeto e para inscrição de novos institutos; aplicar ao projeto tecnologias modernas de integração e *deploy*[1] contínuo para melhorar a experiência do desenvolvedor e acelerar o ciclo de vida de desenvolvimento de software; interagir com os usuários das páginas e implementar novas funcionalidades e corrigir *bugs*[2] de maneira eficaz.

[1] Ver **Apêndice A**.

[2] Ver **Apêndice A**.

1.3 Metodologia

A equipe de 2022 com participação ativa no projeto foi composta pelo coordenador do projeto Alfredo Goldman, pelo estudante de design Gustavo Machado, ainda sob supervisão do doutorando Luís Felipe Abbud, e pelos dois autores deste trabalho, responsáveis pelo desenvolvimento do projeto. A intenção inicial dos desenvolvedores foi de seguir a metodologia aplicada pelos mantenedores anteriores, seguindo as práticas da metodologia ágil, desenvolvendo e lançando o código em pequenas etapas. O repositório do projeto pode ser encontrado em https://gitlab.com/cpqs-abertas/cpqs-abertas.

Inicialmente, foram feitas reuniões com a equipe anterior para entender o estado do projeto, suas limitações, como as mudanças estavam sendo implementadas e lançadas, e quais eram as sugestões quanto aos próximos passos a serem seguidos. Dessa forma, pôde-se obter certa familiaridade com a estrutura inicial do projeto para iniciar as contribuições.

Em seguida, uma análise foi feita em cima de todo o código disponível no repositório, além dos últimos *commits*[3] realizados, buscando entender por conta dos autores essa estrutura. Feito isso, começou-se o desenvolvimento propriamente dito, cujas atividades serão descritas a fundo nas seções a seguir.

A metodologia de escrita de código utilizada foi diferente das tarefas síncronas realizadas pela equipe anterior. Baseado nos procedimentos seguidos por grandes empresas do ramo de desenvolvimento de software, as mudanças de código eram publicadas em *branches*[4] com poucas mudanças de código, as quais eram revisadas pelo outro membro em um *pull request*[5] aberto na plataforma do Gitlab. Dessa forma, tem-se um histórico bem estruturado das mudanças feitas no código, juntamente com descrições e motivações das mudanças, além dos comentários e discussões entre os membros, podendo ser posteriormente analisados por novos membros do projeto.

Quanto ao planejamento, a escolha das tarefas foi feita juntamente com o orientador, com limitada participação da equipe de design, e atualizações eram feitas mensalmente entre toda a equipe por meio de discussões de e-mail utilizando um grupo de discussão (cpqs-abertas@usp.br).

[3] Ver **Apêndice A**.

[4] Ver **Apêndice A**.

[5] Ver **Apêndice A**.

Capítulo 2

Estado anterior do projeto

O CPqs Abertas é um *fork*[1] do FAU Aberta, projeto que contava com uma arquitetura bastante diferente do que foi entregue pela primeira equipe de desenvolvedores nesse *fork*. A fim de simplificar este texto, será explicado apenas o software existente após o término do trabalho da primeira equipe, evitando entrar em detalhes de código existente apenas no FAU Aberta.

2.1 Estrutura Geral

O sistema encontrava-se dividido em três partes: o *front-end*[2], o *back-end*[3] e o banco de dados. Para o *front-end*, tinha-se código escrito em Javascript, utilizando React para renderizar os componentes. Este código ficava contido em um contêiner durante a execução do sistema. Já o *back-end* contava com código em Python, utilizando o *framework*[4] Django para receber e responder às requisições do *front-end*, e também para interagir com o banco de dados principal. Este código é executado em outro contêiner. Por fim, na parte do banco de dados, existem dois diferentes serviços de bancos de dados: o PostgreSQL e o Mongo DB, cada um destes sendo executado em um diferente contêiner.

A figura a seguir exibe essa arquitetura de forma simplificada:

[1] Ver **Apêndice A**.

[2] Ver **Apêndice A**.

[3] Ver **Apêndice A**.

[4] Ver **Apêndice A**.

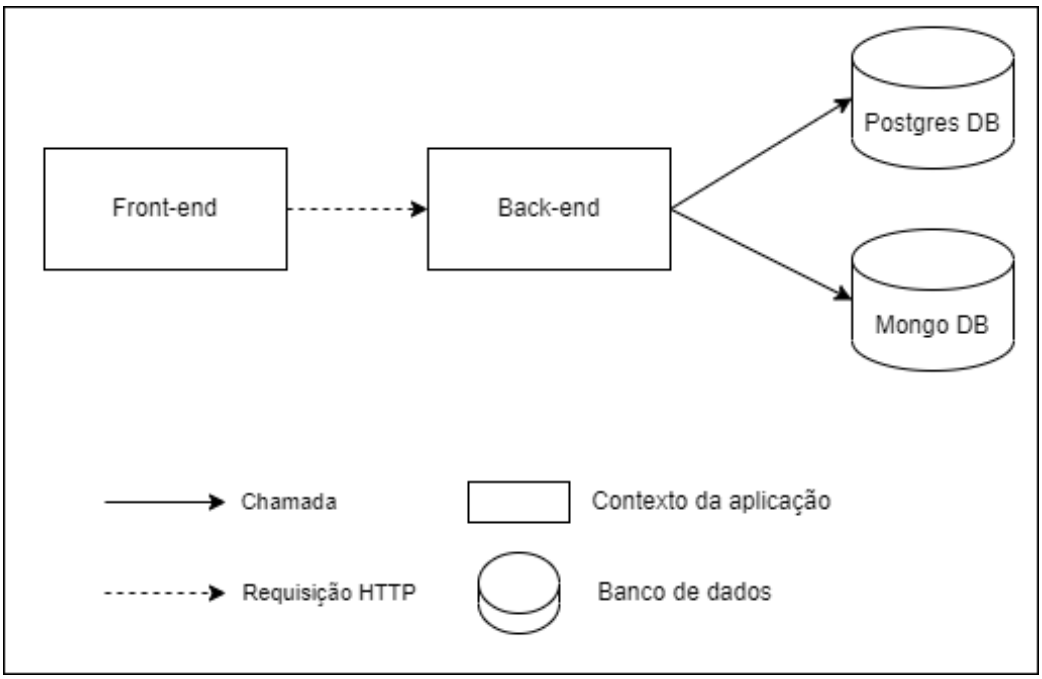

Figura 2.1: *Estrutura geral de uma página do CPqs Abertas*

2.2 Plataforma

Para manter cada uma das partes do sistema separada e possibilitar sua criação e manutenção separadamente, o projeto CPqs Abertas utiliza o serviço Docker para a criação de imagens, as quais funcionam como um conjunto de instruções para a inicialização dos contêineres de fato.

Como dito anteriormente, uma página do CPqs Abertas possui quatro contêineres criados a partir de quatro imagens diferentes, e estes quatro devem estar em execução simultânea para o funcionamento da página. Além disso, integrações devem ser feitas entre os contêineres, além da criação e utilização de volumes para uso de disco dos mesmos. Para facilitar estas etapas, foi utilizada a ferramenta Docker Compose, cuja finalidade é lidar com aplicações que manipulam múltiplos contêineres. A partir de um arquivo Compose (dois no CPqs Abertas: um para desenvolvimento, outro para produção), escrito no formato YAML, essas dependências e comunicações entre os contêineres pode ser explicitamente declarada. Com isso, ao invés de utilizar comandos separados para gerar cada uma das imagens e então executar cada um dos contêineres, passa a ser necessário apenas um comando para realizar todas as atividades descritas.

Assim, o CPqs Abertas continha dois arquivos Dockerfile, um para o *front-end* e outro para o *back-end*, os quais determinavam a imagem base para a criação destas imagens, assim como alguns comandos de inicialização do serviço, e dois arquivos Docker Compose, que, além de realizar tudo que foi descrito anteriormente, também definia as imagens base e algumas propriedades como portas de rede a serem abertas e volumes a serem utilizados pelos contêineres dos bancos de dados (MongoDB, PostgreSQL).

2.3 Banco de dados

Os bancos de dados são responsáveis por armazenar os dados que serão renderizados no *front-end*. Estes dados são, para as páginas do CPqs Abertas, informações sobre os docentes dos institutos, além de todas as suas produções acadêmicas registradas na plataforma Lattes.

Originalmente, no projeto FAU Aberta, apenas o banco de dados gerenciado pelo sistema PostgreSQL, um sistema gerenciador de banco de dados do tipo relacional, existia, e este armazenava todas os dados existentes na plataforma. Porém, com o desenvolvimento do CPqs Abertas, a equipe implementou uma otimização na busca de dados agregados para disponibilização no *front-end*. Anteriormente, estes dados eram buscados e agregados a cada chamada para o banco de dados. Para a otimização citada, a equipe implementou outro sistema de banco de dados, o MongoDB, que gerencia bancos de dados NoSQL, isto é, não relacionais. Neste novo banco de dados, a agregação dos dados era realizada durante a inicialização do sistema, e estas agregações eram armazenadas neste novo serviço, mantendo assim dois distintos bancos de dados.

2.4 *Back-end*

No CPqs Abertas, o *back-end* é responsável pela comunicação com os bancos de dados, além de disponibilizar uma API para envio de e-mails. Ambas as funcionalidades são utilizadas pelas páginas do *front-end*, mantendo a lógica de busca e a maior parte da manipulação dos dados no *back-end*, o qual servia estes prontos para renderização no lado do cliente por meio do *front-end*.

Como dito anteriormente, a ferramenta utilizada nesta parte do sistema foi o Django, um *framework* para desenvolvimento web feito para a linguagem de programação Python. Esse *framework* fornece a criação e manutenção de um serviço web por meio da definição de uma API utilizada para conectar rotas implementadas dentro do *back-end* e URLs, utilizadas para acessar essas rotas por meio, nesse caso, do *front-end*.

Além disso, outra ferramenta poderosa do Django utilizada no CPqs Abertas foi o Mapeamento Objeto-Relacional (ORM), cuja função é mapear tabelas de um banco de dados relacional em classes do Python. Desta forma, a criação e manipulação de dados torna-se menos complexa, simplificando o código-fonte no que diz respeito a interação com o banco de dados.

Por fim, para definir as rotas que realizavam determinadas ações do sistema, um conjunto de aplicações foram criados. Cada aplicação contém dentro de si a definição das rotas disponíveis e as ações que cada rota realiza quando acessada.

A estrutura do *back-end* pode ser vista na imagem a seguir:

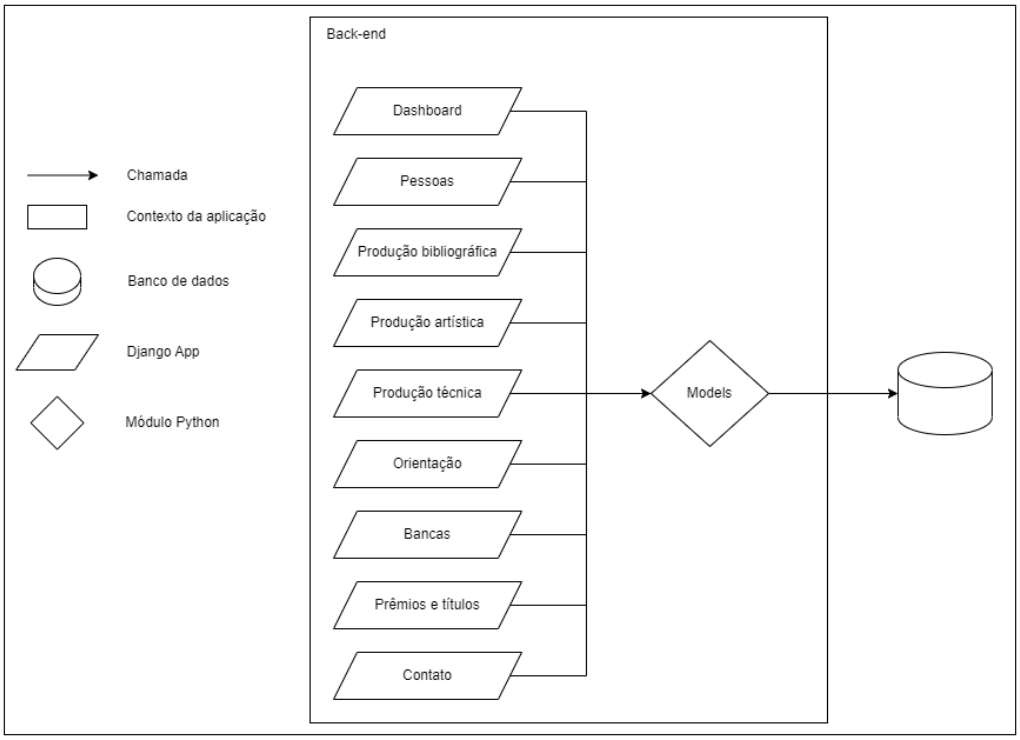

Figura 2.2: *Estrutura do back-end de uma página do CPqs Abertas*

2.5 *Front-end*

O *front-end* é a parte do sistema que é vista pelos usuários. Ele é responsável por solicitar dados ao *back-end* e entregar estes dados de forma significativa ao usuário, renderizando componentes gráficos e criando a experiência de navegação do mesmo na página.

Esse serviço disponibiliza uma página padronizada para os diferentes institutos do CPqs Abertas, utilizando uma estrutura de página singular na qual o usuário navega, alterando apenas seu conteúdo interno.

O *front-end* utiliza a linguagem Javascript, bastante comum no mercado para desenvolvimento de páginas web interativas. A renderização das páginas é feita no navegador do cliente (renderização *client-side*[5]) com o envio do código em Javascript ao requisitar a página, e utilizando este mesmo Javascript para toda a navegação no site, requisitando ao *back-end* novos dados quando necessário.

Este código em Javascript utiliza o React como *framework* de desenvolvimento *front-end*, um dos mais utilizados para este fim. O React é focado em criar interfaces de usuário por meio do uso de componentes, usados para representar desde páginas inteiras a objetos dentro delas, como botões ou parágrafos de texto. Estes componentes têm a característica de serem dinâmicos, podendo mudar de acordo com a interação com o usuário, e fazendo requisições para o *back-end* para atualizar os dados apresentados. No sistema do CPqs

[5] Ver **Apêndice A**.

Abertas, um componente central App é responsável por determinar as quais rotas do serviço exibem cada componente.

2.6 Elementos complementares

Além dos componentes centrais que foram citados, o sistema conta com alguns *scripts*[6] utilizados para configurar o ambiente e popular os bancos de dados.

Os dados dos professores e de suas produções acadêmicas são disponibilizados pela plataforma Lattes do CNPq por meio dos currículos Lattes. Estes currículos são obtidos no formato XML, e o projeto CPqs Abertas conta com um *parser*[7] para realizar a interpretação destes, gerando a partir deles entidades que podem ser inseridas nos bancos de dados. Anteriormente, no projeto FAU Aberta, existia um *script* que interagia com a página web do Lattes para obter estes currículos de forma automatizada. Porém, com a introdução de *captchas*[8] na página, este programa foi descontinuado, obrigando a equipe a atualizar manualmente o banco por meio de solicitações feitas aos institutos participantes para receber cópias internas destes currículos.

[6] Ver **Apêndice A**.

[7] Ver **Apêndice A**.

[8] Ver **Apêndice A**.

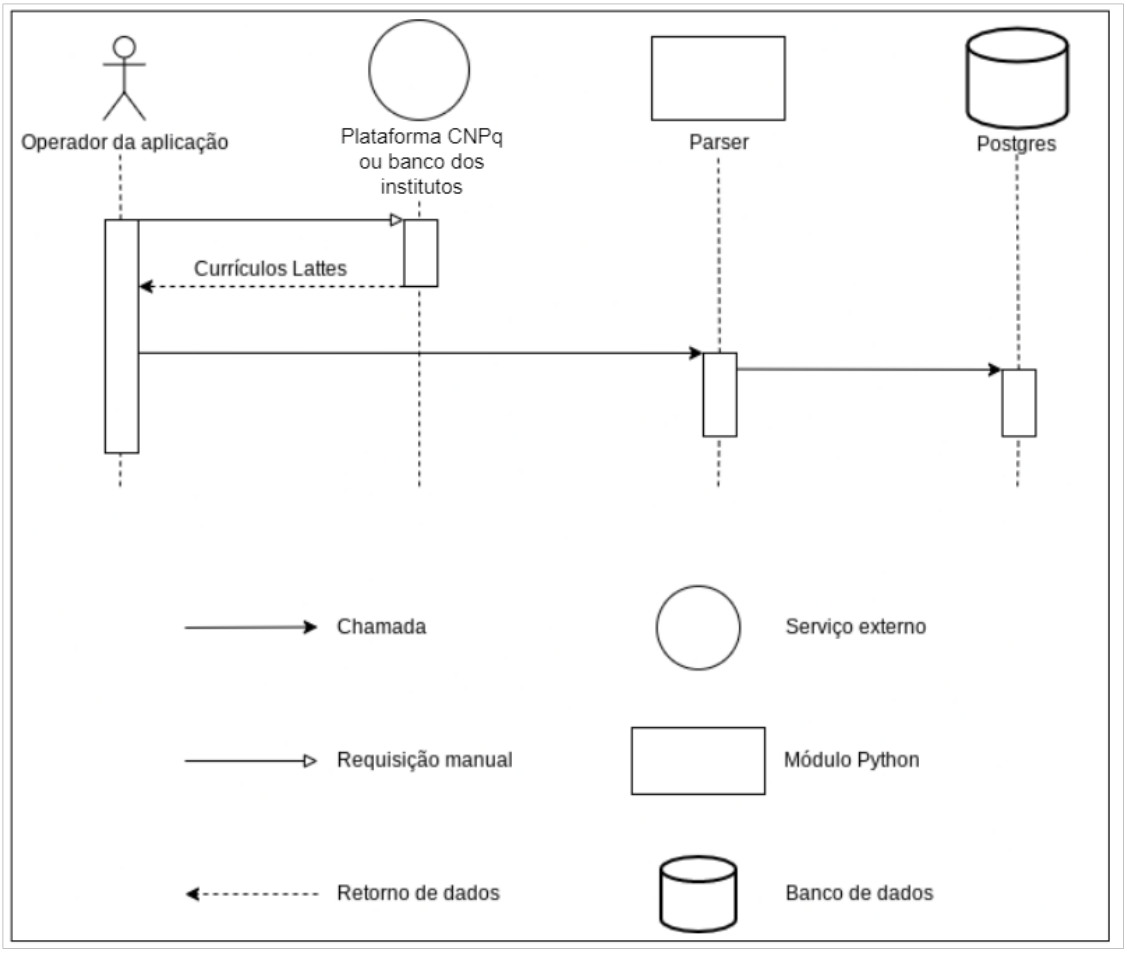

Figura 2.3: *Povoamento dos dados pelo operador da aplicação*

2.7 Infraestrutura

Quanto à infraestrutura do projeto, o código do CPqs Abertas fica armazenado em um repositório no Gitlab na seguinte URL: https://gitlab.com/cpqs-abertas/cpqs-abertas. O projeto utiliza da *pipeline* do Gitlab para realizar testes automatizados no *back-end* a cada *push*[9] feito para o repositório.

Quanto aos servidores, a página FAU Aberta encontra-se hospedada em um servidor *on premises*[10] da FAU, enquanto a FEA-RP Aberta não estava hospedada em local algum.

A hospedagem em servidores locais da USP apresentava um desafio na implementação de novas mudanças feitas nas páginas, pois cada uma dessas mudanças a ser implantada necessitava do auxílio da equipe técnica de cada um dos institutos em questão, e quaisquer falhas nos servidores ou nas aplicações do CPqs Abertas tinham uma demora maior no tempo de diagnóstico e de correção se comparado a um servidor em posse da equipe de desenvolvimento.

[9] Ver **Apêndice A**.

[10] Ver **Apêndice A**.

Capítulo 3

Atividades desenvolvidas

3.1 Novas páginas

3.1.1 IME Aberto

A contribuição no projeto iniciou-se em Abril, quando, ao reunir os alunos João Gabriel Loureiro de Lima Lembo, Leonardo Alves Pereira, Victor Pereira Lima, responsáveis pelo desenvolvimento em 2021, e os alunos responsáveis em 2022, realizou-se a passagem de bastão. Através de uma conversa inicial, houve a orientação sobre os próximos passos a serem tomados e uma listagem das pendências existentes. Nesse momento, o desenvolvimento do IME Aberto encontrava-se em fase final. Esse contexto permitiu que os novos alunos se unissem aos veteranos na finalização do mesmo. Pouco código foi necessário para isso, mas permitiu uma transição mais fluida e um melhor entendimento, por parte dos novatos, do código já utilizado no projeto.

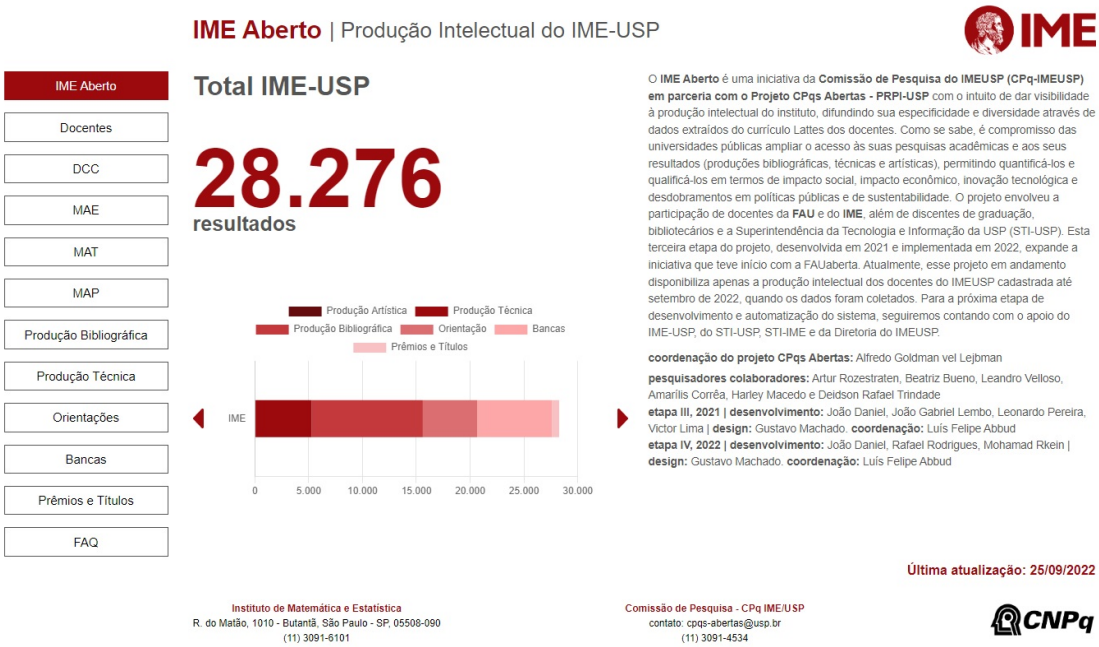

Figura 3.1: *IME Aberto*

O IME Aberto, assim como as outras páginas CPqs Abertas, conta com as seguintes funcionalidades:

1. Página inicial sumarizando dados de todas as produções acadêmicas produzidas pelo instituto

2. Páginas separadas para dados de cada tipo de produção acadêmica (Produção Técnica, Produção Bibliográfica, Orientações dos docentes, Bancas, Prêmios e Títulos)

3. Páginas para cada departamento

4. Filtros das produções por departamento

5. Página para cada docente

6. Filtro das produções por docente

3.1.2 Hub CPqs Abertas

Com os resultados positivos obtidos pelo CPqs Abertas e sua consequente expansão, após a apresentação do projeto na PRPI da USP feita pelo professor Alfredo Goldman, novos institutos interessados em aderir ao projeto começaram a entrar em contato. Visando facilitar a inscrição destes institutos e também centralizar o acesso às páginas já existentes, a proposta do Hub CPqs Abertas foi lançada. O Hub deveria funcionar como um portal central.

Figura 3.2: *Hub CPqs Abertas*

Assim, após a equipe de design definir os *mockups*[1], iniciou-se o desenvolvimento do Hub. Para esse desenvolvimento, buscou-se manter o mesmo estilo de código utilizado nas outras páginas do CPqs, de modo a manter uniformidade, facilitar o entendimento e aproveitar a modularização do projeto. Por isso, muitos componentes já existentes no *front-end* das outras páginas foram reaproveitados ou adaptados para o Hub, sendo um dos principais exemplos o modal da seção de FAQ, que é idêntico em ambos contextos.

Foram desenvolvidas as seguintes funcionalidades:

1. Carrossel de Páginas CPqs Abertas - um carrossel presente na página inicial que mostra, através de figuras clicáveis e resumos descritivos, as unidades que já possuem uma página no CPqs Abertas e aquelas que estão na fila para lançamento.

[1] Ver **Apêndice A**.

Figura 3.3: *Carrossel do Hub, com página na fila para desenvolvimento*

2. Página de descrição do projeto, semelhante ao que se tinha na página inicial dos institutos
3. FAQ no estilo *pop-up*[2] com descrições sobre o projeto CPqs Abertas
4. Página de inscrição de institutos, incluindo uma fila de páginas em desenvolvimento, e um formulário para preenchimento por partes interessadas em inscrever seu instituto, o qual envia automaticamente um e-mail para a equipe de desenvolvimento com os dados do inscrito

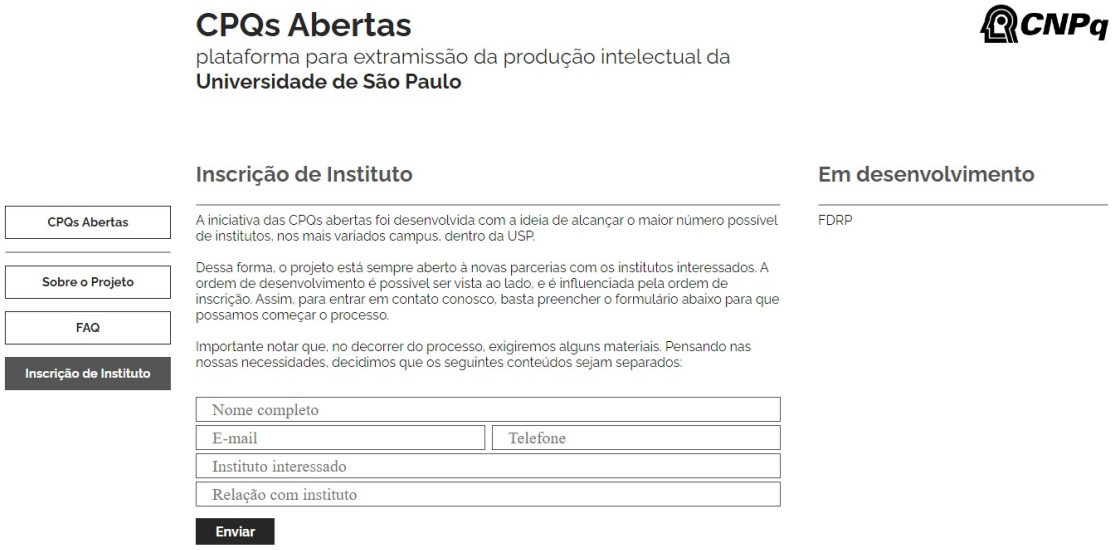

Figura 3.4: *Página de inscrição do Hub CPqs Abertas*

[2] Ver **Apêndice A**.

3.2 Hospedagem das páginas

3.2.1 Migração para a AWS

Devido ao caráter do projeto, a manutenção das plataformas do CPQs Abertas é responsabilidade direta dos alunos envolvidos. Entretanto, nem sempre depende apenas deles: há terceiros com grande impacto.

Com o passar dos anos, relatos e reclamações sobre a baixa disponibilidade do site FAU Aberta alcançaram os professores e alunos do projeto. O FAU Aberta era, desde o princípio, hospedado nos servidores da própria instituição. Com isso, tudo precisava ser enviado ao setor técnico responsável. Esse fator dificultava diversos processos cruciais na jornada de desenvolvimento, como: coletar e debugar erros, realizar *deploys*, disponibilizar atualizações, entre outros.

A fim de mudar esse cenário, melhorar a experiência de desenvolvimento e aumentar a disponibilidade do site, o professor responsável Alfredo Goldman sugeriu aos alunos envolvidos tornar a hospedagem das plataformas do CPqs Abertas independente da universidade.

Para tal propósito, escolheu-se a AWS como solução. Essa escolha possuiu duas motivações principais: a partir de uma parceria entre a Amazon e a USP, obtiveram-se créditos para a utilização dos serviços AWS sem gerar custos. Além disso, a AWS é atualmente a maior e mais completa solução de serviços de computação em nuvem mundialmente, o que garante alta disponibilidade, fácil integração com outras ferramentas, boa documentação, entre outras vantagens.

Após a escolha do produto, a escolha da capacidade de processamento do servidor se deu por meio de cálculos estimativos de carga de acesso. As estimativas realizadas sugeriram que a instância micro - instância mais básica oferecida pelo Amazon EC2 - seria suficiente para lidar com os acessos.

Os servidores Amazon EC2 contratados foram configurados para rodar instâncias Linux. Neles, realizou-se a instalação do Docker, ferramenta de conteinerização utilizada pelo projeto. A partir daí, quando é necessário realizar alguma atualização num dos sites do CPQs, realiza-se a conexão aos servidores por meio de SSH e através do *plugin*[3] Compose do Docker realiza-se o *deploy* dos contêineres dos sites.

Os servidores Amazon EC2 possuem um índice de 99,99% de disponibilidade, o que significa um máximo de 1h fora do ar por ano. Além disso, podem ser diretamente acessados pelos alunos envolvidos no desenvolvimento para realizar manutenções necessárias.

[3] Ver **Apêndice A**.

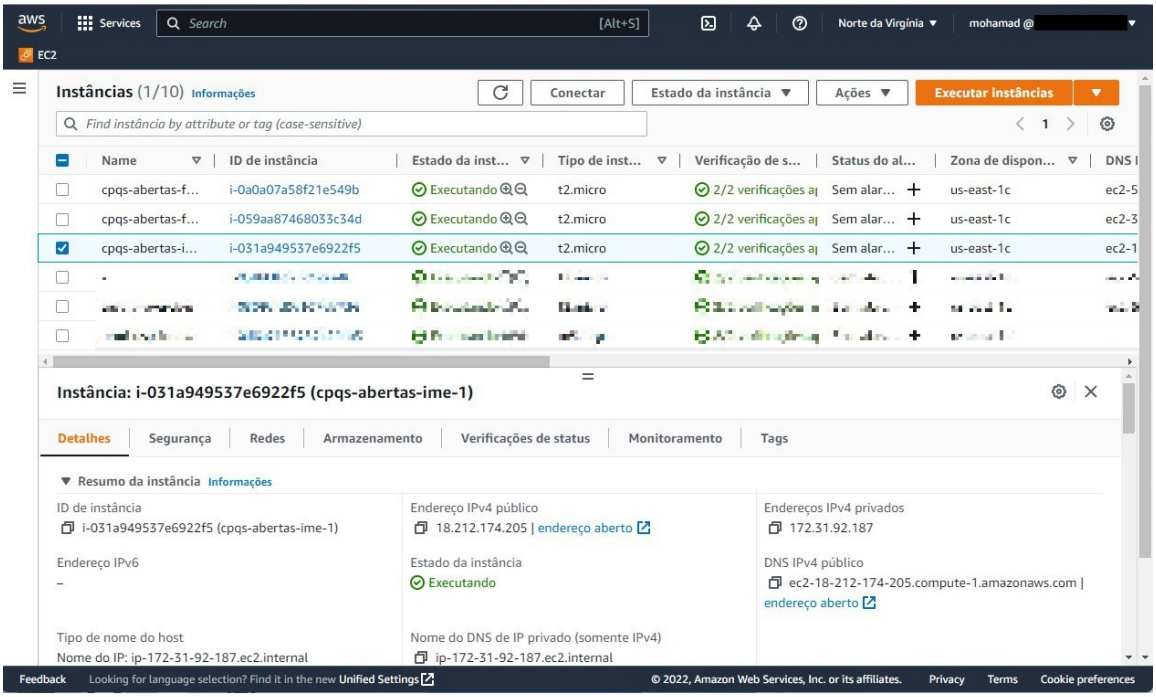

Figura 3.5: *Lista de servidores no painel do EC2 na AWS*

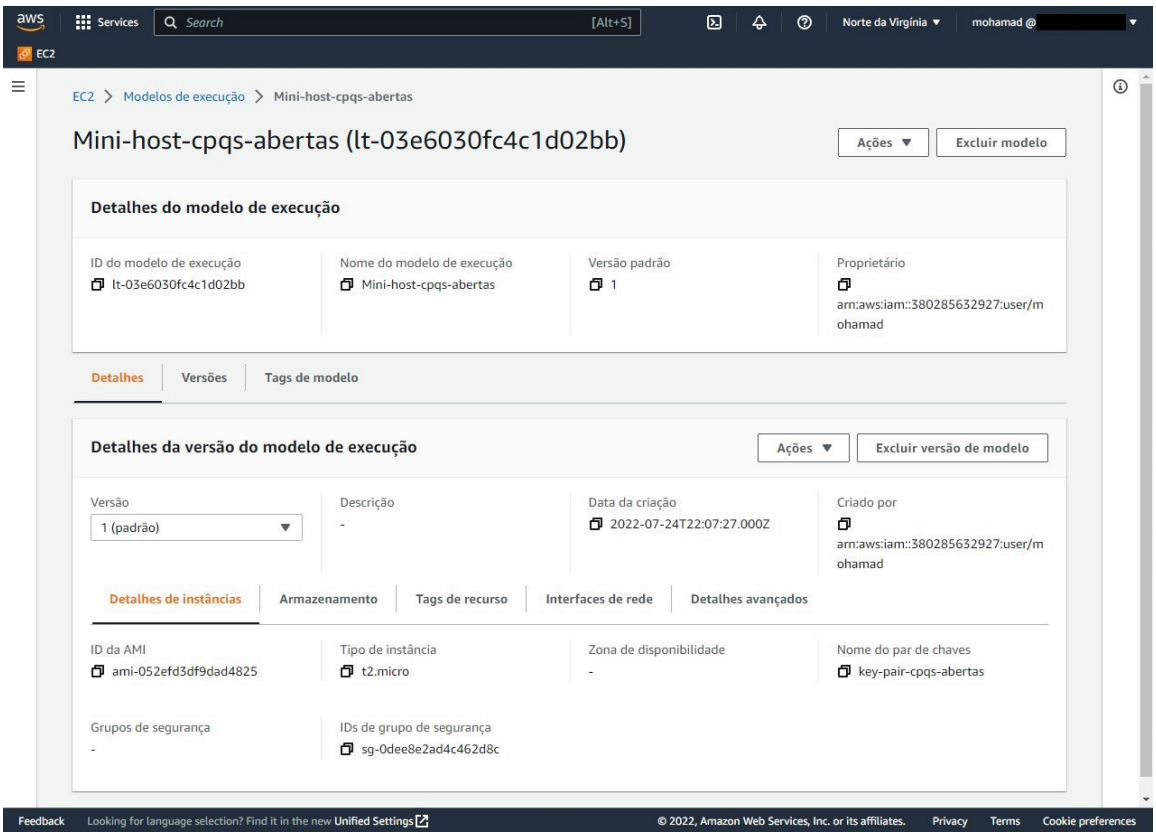

Figura 3.6: *Modelo para criação de servidor padronizado para o CPqs Abertas*

3.2.2 Teste de carga

A fim de comprovar a corretude das estimativas, realizou-se um teste de carga na instância mais simples oferecida na AWS (t2.micro). Nesse teste, foram realizadas até 5 requisições por segundo, e isso não afetou a latência de resposta do servidor. De acordo com a estimativa de uso inicial das páginas feita com o orientador do projeto, 5 TPS está acima da quantidade de requisições necessárias, o que levou a escolha desta instância para a fase inicial do projeto.

O teste de carga consistiu em um código que utilizava Selenium - *framework* muito comum para interação e testes com páginas web - e Python, e executava requisições nas páginas de maneira constante. A comprovação obtida foi importante para manter os custos do projeto baixos, através de uma solução barata e comprovadamente eficaz.

Entretanto, um ponto negativo de utilizar um *host* tão simples também se destacou: não é possível atribuir ao próprio *host* a tarefa de realizar o *build* do projeto, pois o *build* necessita mais processamento do que o oferecido. No caso, a solução encontrada foi atribuir à *pipeline*[4] do Gitlab o trabalho de *build*.

3.3 Melhorias na *pipeline*

Como dito anteriormente, repositório do CPqs Abertas conta com o uso da infraestrutura gratuita de *pipeline* oferecida pelo Gitlab. Um problema encontrado foi que a *pipeline* não bloqueava a passagem de mudanças de código que causassem uma falha na execução dela, como, por exemplo, um teste que deixa de ser bem sucedido. Isso acaba indo contra o propósito de possuir uma *pipeline*, pois deixa-se de averiguar a qualidade das mudanças de código.

3.3.1 Correção dos testes do *back-end*

Ao final da transição de equipe, o código-fonte do projeto contava com alguns pequenos problemas que impediam a correta execução da *pipeline* no Gitlab. Muitos testes executados pela *pipeline* estavam desatualizados em relação ao código e por isso não entregavam o resultado correto durante a execução, levando a *pipeline* a falhar mesmo quando não devesse.

Apesar de ser uma tarefa enfadonha, entende-se que a *pipeline* é um recurso de grande importância num projeto em que há desenvolvimento contínuo, e por isso foi decidido priorizar a correção desses *bugs* que impediam sua completude.

Os problemas nos testes tinham como origens a má configuração de algumas das rotas do *back-end*, e a execução de um comando incorreto durante a execução do *script* de testes. Com a correção destes problemas, aproveitou-se para adicionar testes para uma rota que não era testada anteriormente, e com isso a *pipeline* voltou a comportar-se de maneira esperada, validando mudanças corretas e barrando grande quantidade de erros antes que chegassem ao repositório central.

[4] Ver **Apêndice A**.

3.3.2 Deploy contínuo

Ao realizar uma mudança de código, com a passagem da posse dos servidores descrita na seção anterior, a tarefa de lançar as mudanças passou a ser de responsabilidade dos desenvolvedores da equipe. Dito isso, os mantenedores do projeto gastavam uma grande quantidade de tempo colocando essa mudança no ar, devido ao trabalhoso processo de *deploy* manual, explicado anteriormente na seção de Migração para a AWS. Esse tempo gasto limita a contribuição dos desenvolvedores, atrasa entregas e diminui a frequência de atualizações.

Diante desse cenário, buscou-se implementar um processo de *deploy* contínuo: passa-se por todos os passos de *build*[5], testagem e lançamento, e a aplicação é entregue ao usuário final de maneira automática. Como o projeto já é inteiramente armazenado em repositório no GitLab, optou-se por utilizar as ferramentas disponíveis nessa plataforma para atingir o objetivo estabelecido.

O *workflow*[6] definido para o repositório já contava com um processo de integração contínua, explicado anteriormente, que consiste em *build* e realização automática de testes de *back-end*. Assim, acrescentou-se a esse *workflow* os passos necessários para o *deploy* contínuo nas máquinas AWS que hospedam o projeto. Portanto, todo *pull request* aprovado e enviado à *branch* principal do projeto passa pela *pipeline* e, em caso de completar assertivamente o conjunto de testes, segue para o *deploy* sem necessitar de interação humana.

Devido a certas limitações do plano utilizado pelo repositório, a tarefa do *deploy* precisou ser configurada em baixo nível pelos desenvolvedores, com a criação de *scripts* para a execução automática da construção da imagem de Docker nos executores de testes da *pipelines* do Gitlab, e então do envio dessa imagem aos servidores, finalizando com a execução de novos contêineres com essas imagens. Os *scripts* citados serão explicados em mais detalhes na seção de Refatoramentos e Automações.

Figura 3.7: *Fluxograma explicando o processo de integração e deploy contínuos*

3.3.3 Criação de testes de integração

Até o momento de início deste trabalho, as páginas do projeto CPqs Abertas contavam apenas com testes no *back-end* feitos sobre as integrações do Django. Nenhum teste era realizado no *front-end*, criando-se uma vulnerabilidade no caso em que um *bug* fosse introduzido neste código e não causasse um erro durante a inicialização do contêiner.

[5] Ver **Apêndice A**.

[6] Ver **Apêndice A**.

Assim, foi decidido criar testes de integração, os quais não apenas testariam o *front-end*, visto que os mesmos verificam se o sistema como um todo está operante de maneira esperada.

A tecnologia escolhida para realizar estes testes foi o Selenium, biblioteca de Python utilizada para controlar um navegador de forma automática, interagindo com elementos HTML de páginas web e realizando requisições se passando por um humano. O Selenium pode conectar-se a diversos navegadores bastante utilizados no mercado, como o Google Chrome e o Mozilla Firefox.

Os testes implementados acessam a página do instituto sob teste, e realizam múltiplas verificações:

- Verifica se o número de produções renderizadas na tela é maior que zero
- Interage com diferentes botões e seções da tela
- Confirma que páginas seguintes são carregadas com sucesso
- Verifica se os textos foram renderizados corretamente

Por fim, estes testes foram incluídos na *pipeline*, de forma que são executados de maneira automática a cada mudança de código, garantindo que o sistema está funcional antes de realizar o *deploy*.

3.4 Refatoramentos e Automações

3.4.1 Configuração inicial de um novo servidor

Quando um novo servidor é inicializado para hospedar uma aplicação do CPqs Abertas, certas ferramentas precisam ser instaladas e atualizadas para que seja possível executar o Docker e as imagens do projeto.

Por conta disso, foi criado um *script* para a realização automática destas configurações, garantindo que cada novo servidor contará com todas as tecnologias necessárias para o uso do projeto.

Esse *script* pode ser encontrada no **Apêndice B**.

3.4.2 Configuração e inicialização dos contêineres

Lidar com os vários comandos do Docker durante a inicialização das aplicações pode ser um pouco complexo. É necessário um comando para montar as imagens, outro para inicializar os contêineres, outro para executar as migrações do Django no banco de dados e outro para popular o banco de dados. Para facilitar este processo, os antigos mantenedores do projeto já haviam criado *scripts* em Bash para unir os comandos e executá-los em conjunto, diminuindo o trabalho e a complexidade desta inicialização.

Visando dar continuidade ao trabalho feito, primeiramente realizou-se a correção e atualização dos *scripts* de execução, a fim de possibilitar outras operações ao usuário - no caso, desenvolvedor que está utilizando o código.

Inicialmente, havia apenas um *script* que permitia a execução em modo produção e que necessitava de variáveis de ambiente definidas em arquivos auxiliares. Esse trabalho podia tornar-se confuso e manual demais. Por isso, primeiramente foi criado um *script* adicional que permitiu a execução em modo de desenvolvimento. Posteriormente, optou-se por unir os dois *scripts* em um, permitindo ao usuário escolher o modo de execução através de argumentos passados na linha de comando. Por fim, foi realizado um refatoramento a fim de facilitar a execução automática, recebendo mais argumentos diretamente no comando, sem a necessidade de modificar arquivos auxiliares para variáveis de ambiente, possibilitando que a mesma configuração fosse feita em dois ambientes diferentes com apenas a repetição de um comando no terminal, e diminuindo espaço para erro humano.

Além disso, efetuou-se a atualização dos comandos Docker presentes no *script* de execução, substituindo a utilização do antigo - e deprecado - projeto Python "docker-compose"pela sua versão nova, o *plugin* Compose, novo projeto escrito em Go e suportado oficialmente pela equipe do Docker. Tal mudança buscou evitar manter dependências deprecadas no projeto.

A versão final do *script* pode ser encontrada no **Apêndice B**.

3.4.3 Geração de imagens para produção

Semelhante ao *script* anterior, outro foi criado com o intuito de gerar imagens prontas para serem enviadas para os servidores, garantindo que variáveis de ambiente e configurações específicas de cada servidor fossem feitas corretamente. Esse *script* foi utilizado na configuração do *deploy* contínuo.

O *script* pode ser encontrado no **Apêndice B**.

3.4.4 *Deploy* dos contêineres por SSH

Para completar o processo de *deploy* contínuo, um último *script* foi desenvolvido a fim de ser executado pela *pipeline* do Gitlab, iniciando uma conexão SSH com o servidor escolhido, copiando a imagem gerada pelo *script* anterior e executando o *setup* dos contêineres. Para obter as credenciais destes servidores, foi utilizado um par de chaves criptografado, armazenado no Gitlab, sendo usado posteriormente por este *script* para conseguir acesso aos servidores.

Este *script* pode ser encontrado no **Apêndice B**.

3.4.5 Coleta dos currículos Lattes

No passado, a coleta dos currículos Lattes dos docentes dos institutos era feita de forma automática por meio de um *web crawler*[7], como foi explicado anteriormente. Quando essa solução deixou de funcionar, a equipe passou a coletar estes dados de maneira manual com a ajuda dos institutos.

[7] Ver **Apêndice A**.

Como essa solução não era escalável e demandava a atenção e trabalho de múltiplas partes, decidiu-se por remover o intermediário (a equipe do instituto) no acesso ao banco de dados interno da universidade, contendo os currículos desejados. Para isso, foi assinado um contrato com o Serviço de Apoio Institucional do IME para obter credenciais para acesso a este banco. Garantido o acesso, fez-se um estudo sobre a estrutura deste banco, descobrindo como estavam armazenados os currículos e outros dados de interesse, como a data da última atualização. Feito isso, foi criado um *script* em Python, que realizava a conexão ao banco de dados de forma automática por meio de um túnel SSH feito para dentro da rede do IME, contornando problemas relacionados a autenticação, visto que a conexão ao banco deveria sempre ser realizada por meio da rede interna da USP.

O Secure Shell (SSH) é um protocolo de rede que pode ser usado para criar uma conexão segura com um computador remoto. Feita a conexão, o terminal local do computador assume o comportamento de um terminal no computador remoto. Para o caso de uso deste projeto, o requisito era semelhante, com a preferência de manter o ambiente de execução sendo local. Para isso, a solução encontrada foi de utilizar o Tunelamento SSH, que consiste em vincular uma porta de rede do computador local com uma porta escolhida no servidor remoto. Dessa forma, requisições enviadas ao banco de dados seriam reconhecidas como vindas de dentro da rede da USP.

Assim, com a execução de um *script*, se faz possível a atualização automática dos currículos.

Uma versão simplificada deste *script* pode ser encontrada no **Apêndice B**.

3.4.6 Coleta dos IDs Lattes dos docentes

Dados apenas os currículos Lattes coletados com o *script* explicado anteriormente, não era possível vincular os docentes aos departamentos de forma confiável e reprodutível, visto que este dado não é explicitamente declarado no currículo.

Dessa forma, optou-se por tomar as páginas oficiais dos institutos e seus departamentos como a fonte de verdade para essa informação. Assim, cada docente era vinculado ao instituto cuja página oficial continha seus dados como membro do mesmo. O dado a ser vinculado de fato era o ID Lattes, e para fazer isso com o menor esforço, foi criado um *script* em Python utilizando a biblioteca BeautifulSoup, facilitador para interpretação e interação com HTML, para acessar as páginas e coletar estes dados.

Uma versão simplificada deste *script* pode ser encontrada no **Apêndice B**.

3.4.7 Formatação de Código

O Linter é uma ferramenta de análise estática de código-fonte, utilizada para encontrar erros, *bugs* e problemas de estilização de código antes da execução do mesmo. Junto dessa ferramenta, o Formatador também é comumente utilizado, encontrando problemas de espaçamento, nomes de variáveis, entre outras questões ligadas ao visual do código. Duas destas ferramentas bastante utilizadas para Javascript são o ESLint (Linter) e o Prettier (Formatador).

Dito isso, além dos *scripts* em Bash, os *scripts* do React, também sofreram modificação. Utilizando comandos do NPX, executor de pacotes do NPM (gerenciador de pacotes de Node), e as duas ferramentas de estilização citadas acima, foram adicionadas funcionalidades de padronização de código. Assim, antes de submeter uma mudança de código, um desenvolvedor pode verificar e corrigir de maneira automática este novo trecho escrito para que se encaixe nos padrões de formatação do projeto. Dessa forma, o código-fonte mantém alta legibilidade.

3.4.8 Atualizações no *front-end*

Por ser uma área em crescimento, a computação em geral passa por constantes mudanças, e no projeto em questão não foi diferente. A grande maioria das dependências utilizadas pelo *front-end* encontravam-se desatualizadas, e uma boa parte delas estava até mesmo deprecada, estado no qual não se obtém suporte da equipe responsável, podendo conter vulnerabilidades e problemas publicamente conhecidos. Diante disso, julgou-se necessário realizar a atualização de todas as dependências utilizadas.

Apesar de soar como uma tarefa simples, consagrou-se um desafio. Em algumas dependências houve mudança da versão principal da biblioteca, o que implicou em código defeituoso após a atualização por conta de mudanças na API. Algumas atualizações até mesmo deprecaram por completo funcionalidades que estavam sendo utilizadas pelo código do CPqs Abertas. Com isso, inúmeras atualizações e correções no código foram necessárias, apenas para adequar o projeto existente aos novos padrões das dependências usadas.

Ademais, outras pequenas mudanças foram feitas no *front-end*. Atendendo a uma solicitação, atualizou-se os textos presentes nas páginas, inserindo os novos conteúdos fornecidos pela equipe de design e pelas equipes da FEA-RP e do IME.

Em seguida, moveu-se do *front-end* para o *back-end* algumas responsabilidades de manipulação de dados, como a de análise e edição de dados textuais dos docentes. Isso se deu pela ideia de manter manipulação de dados somente no *back-end* do projeto, deixando para o *front-end* apenas as responsabilidades estritamente ligadas à exibição gráfica.

3.4.9 Refatoramento de *queries*

A interação entre o *back-end* do CPqs Abertas e o seu banco de dados se dava por meio de *queries*[8] feitas de duas maneiras: utilizando a API Django integrando com o banco de dados, e também utilizando a linguagem SQL. Com os objetivos de melhorar a eficiência dessa interação, padronizar e organizar o código e até mesmo diminuir a quantidade do mesmo, optou-se por realizar um refatoramento das *queries* usadas.

Nesse refatoramento, substituiu-se todas as *queries* feitas nativamente em SQL por comandos da API do Django, mantendo uma única linguagem e um único padrão de código, optando assim pela opção de mais fácil uso e manutenção, visto que o uso de *queries* SQL pode tornar-se complexo. Além disso, consultas individuais foram agrupadas em consultas em grupo, buscando-se melhorar a eficiência dessas, já que consultas em grupo costumam

[8] Ver **Apêndice A**.

ter tempo de execução menor por conta da redução de múltiplas conexões iniciadas com o banco para apenas uma.

Capítulo 4

Experiências Pessoais

Neste capítulo, será apresentada a opinião de cada um dos autores quanto às atividades de desenvolvimento do projeto e o contato com o restante da equipe.

4.1 Mohamad Hussein Rkein

"Trabalhar em um projeto já existente pode ser uma tarefa muito simples ou muito complicada, e isto vai depender da forma em que o projeto foi feito e estruturado. Em casos onde temos boa documentação, organização estrutural de arquivos e funções bem definida seguindo boa práticas, a introdução de um novo desenvolvedor acaba por ser uma tarefa simples e rápida, e na primeira semana, já é possível fazer a primeira contribuição (seja ela pequena ou grande). Infelizmente ou felizmente, este não foi o caso com o CPqs Abertas. A equipe passada fez um ótimo trabalho na estruturação do código relacionado ao *front-end* e ao *back-end*, porém, arquivos relacionados à configuração do projeto e *scripts* da interpretação dos currículos Lattes dos professores eram confusos, e podia se notar a presença de muito código legado, o que dificultava a inserção de código que fugisse destas más práticas. Além disso, a maior parte da documentação disponível foi escrita pela equipe da FAU Aberta, e deixou de ser relevante há tempos. Por conta disso, bastante esforço precisou ser investido no início do trabalho apenas para entender e corrigir *scripts* e documentações, tornando o processo de entrada no projeto mais tortuoso que seria no cenário ideal. Dito tudo isso, projetos com este aspecto são mais comuns que projetos bem estruturados, e esta acabou por ser uma experiência bastante construtiva para mim no sentido de lidar com projetos neste estado, e tentar aprender o mais rápido possível para que pudesse iniciar as contribuições rapidamente.

Quanto a trabalhar com uma equipe de design, foi uma experiência interessante, dado que não havia passado por algo semelhante durante minha carreira. Ter uma equipe responsável por visualizar o produto final e trazer esta visão para algo concreto e palpável com imagens foi algo que contribuiu bastante no desenvolvimento das novas páginas. Além disso, poder discutir esses designs com a equipe e com o orientador foi interessante para exercitar o pensamento estratégico como desenvolvedor, ao invés de focar apenas em questões técnicas menores relacionadas a funcionalidades.

Por fim, o que me trouxe maior satisfação foi o desenvolvimento direcionado a nós, os desenvolvedores. Reportar problemas, fossem eles de eficiência ou relacionados a funcionalidades, e então encontrar a causa raiz destes problemas e solucioná-los em nosso código foi o processo mais fluído e satisfatório para mim, revelando a área de maior afinidade que tenho com o desenvolvimento de software. "

4.2 Rafael Rodrigues Vieira dos Santos

"Receber um projeto complexo já em funcionamento e ter o dever de mantê-lo é uma grande responsabilidade. E, apesar de todo desenvolvedor passar por esse cenário com frequência, toda experiência como essa gera estresse e nervosismo, pelos mais diversos motivos.

Ao iniciar o ano e me deparar com o código-fonte do CPqs Abertas pela primeira vez, confesso que tive muita dificuldade em compreender tudo ali presente. Algumas partes pareciam bastante desconexas do resto do código, e a documentação não batia. Diante disso, muito esforço inicial foi necessário para compreender o código e ter confiança e conhecimento suficiente para realizar as primeiras contribuições.

Apesar disso, uma vez imerso no código do projeto, foi prazeroso poder contribuir e ver mudanças surtindo efeito. Implementar melhorias no código e funcionalidades para facilitar o desenvolvimento foram atividades que motivaram minha participação.

Além disso, a experiência de ter uma equipe de design como aliada mimetizou um contexto comum na vida de um desenvolvedor no mercado de trabalho, proporcionando preparação para alunos ainda pouco experientes profissionalmente. Poder discutir e contribuir com os designers permitiu que tivéssemos maior participação em todos os âmbitos do projeto.

Portanto, creio que foi uma experiência árdua, porém muito proveitosa e com bons resultados."

Capítulo 5

Próximos passos

Pensando no valor trazido pelo projeto CPqs Abertas e no potencial de expansão do mesmo para novos institutos, deseja-se que continue sendo desenvolvido no futuro próximo, aproveitando-se do projeto extensível criado pela equipe anterior, e também das melhorias de desenvolvimento implementadas pela equipe atual. Dito isso, diversas oportunidades surgiram durante o processo de desenvolvimento, e nem todas puderam ser aproveitadas. Esta seção foca em sumarizar algumas destas oportunidades para servirem como guia de primeiras atividades para equipes futuras de desenvolvimento.

5.1 Relatório de erros no *parsing*

Durante o *parsing* dos currículos Lattes dos docentes, diversos erros acontecem, seja por falta de dados em alguns campos, ou por conta de dados mal configurados e mal formatados. Atualmente, estes casos apenas são ignorados, diminuindo a quantia de dados disponibilizados para estes docentes. Neste caso, a oportunidade apresentada seria de gerar um relatório de erros, explicitando quais as produções acadêmicas ignoradas e por quais motivos, e então enviar estes relatórios em um cronograma (mensal, por exemplo) para os docentes ou aos institutos, visto que o responsável pelo currículo Lattes é o único que tem permissão para realizar a correção.

5.2 Melhor aproveitamento da infraestrutura da AWS

A AWS conta com diversas ferramentas para desenvolvimento em nuvem. Atualmente, o projeto CPqs Abertas utiliza uma quantia mínima destes recursos, com o uso do EC2 para hospedar os servidores. Existem ferramentas para análise constante dos servidores e dos *logs*[1] gerados pelas aplicações por meio do painel do CloudWatch. É possível também configurar testes de canários para verificar constantemente o estado das páginas que estão no ar, e acionar um alarme caso alguma das páginas saia do ar, enviando um e-mail para

[1] Ver **Apêndice A**.

a equipe de desenvolvimento. No contexto da AWS, canários são geradores de tráfego sintético, realizando chamadas às páginas determinadas e verificando o estado das mesmas de tempos em tempos.

O uso destas ferramentas traz uma oportunidade de aperfeiçoar o monitoramento das páginas, diminuindo o tempo de resposta para problemas operacionais, e também trazendo experiências aos desenvolvedores com ferramentas fortemente presentes no mercado.

5.3 Criação de novas páginas para outros institutos

Um objetivo que fazia parte dos planos iniciais deste trabalho, mas que não pode ser muito explorado, foi o de criar novas páginas para mais institutos. Apesar de possuir-se o conhecimento de quais institutos estavam interessados, houve dificuldade de comunicação com as equipes de design, o que acabou atrasando a entrega de novas páginas (IME e Hub) e impedindo a criação de novas para institutos que demonstraram interesse para a equipe anterior.

Nesta situação, como o fluxo de desenvolvimento e lançamento de uma página nova já está bem definido por ferramentas deixadas pelas equipes anteriores, a oportunidade se dá por uma melhor interação entre os membros da equipe de desenvolvimento e das equipes de design, visando impedir o bloqueamento de uma equipe pela outra, e assim podendo estender o projeto e usufruir mais rapidamente de seu potencial.

5.4 Melhoria da acessibilidade das páginas

Durante a apresentação deste projeto para docentes e colegas, um dos últimos notou, sendo um daltônico, que não era capaz de fazer a distinção das cores utilizadas para os diferentes departamentos do IME Aberto. Isto revelou uma falha no processo de escolha de cores e no design da página, pois não se levou em conta quesitos de acessibilidades.

A oportunidade existente seria de realizar um estudo sobre acessibilidade nas páginas, buscando entender onde existem outras falhas, e trabalhar em conjunto com a equipe de design para criar uma experiência mais inclusiva no projeto.

5.5 Uso da API do Lattes

Como foi explicado ao longo do trabalho, atualmente, os dados utilizados tem como origem o banco de dados interno dos institutos da USP, os quais armazenam os currículos Lattes dos docentes. Dessa forma, fica-se dependente de obter o acesso a estes bancos, e isso deve ser feito com cada instituto diferente.

A CNPq oferece uma API para busca dos currículos Lattes de usuários cadastrados na plataforma. Porém, para que o uso desta API seja possível, é necessário um cadastro e a assinatura de um contrato junto com a CNPq justificando a motivação do uso destes dados. Neste caso, o trabalho para realizar esta melhoria seria simples, mas o valor de integrar

o uso desta API de forma automática pelos servidores seria enorme, garantindo sempre que as páginas contenham a última versão possível dos dados advinda da fonte da verdade destes.

Capítulo 6

Conclusões

As contribuições realizadas ao longo do trabalho promoveram, em geral, bons resultados. Acredita-se que os objetivos previamente estabelecidos foram atingidos, mesmo diante das dificuldades e desafios encontrados.

Entende-se que, por meio de todas as atividades desenvolvidas, foi possível alcançar o objetivo geral de dar continuidade ao CPqs Abertas, uma vez que o projeto agora conta com novas funcionalidades, maior facilidade de manutenção e mais páginas de institutos no ar.

Além disso, através da criação do Hub CPqs Abertas, entregou-se uma plataforma unificada e centralizada para acesso às páginas do projeto, contribuiu-se para a expansão da divulgação para mais institutos e facilitou-se a inscrição de novos institutos.

Por meio da migração de hospedagem do projeto, obteve-se maior autonomia sobre a entrega de atualizações ao projeto, melhorando a experiência de desenvolvimento dos envolvidos através da redução do tempo e esforço gastos.

Também contribuíram para a melhoria da jornada de desenvolvimento os aprimoramentos na *pipeline* e as automações feitas, já que reduziram a suscetibilidade das entregas a erro humano e aceleraram o ciclo de vida do desenvolvimento. Da mesma forma, os refatoramentos aumentaram a legibilidade do código e, juntamente com as recomendações deixadas, contribuíram para uma imersão facilitada de novos desenvolvedores no projeto.

Dessa forma, é esperado que o projeto continue em desenvolvimento, mantendo-se o objetivo original de difundir informações sobre a quantidade e qualidade de produções acadêmicas originadas na USP para uma maior parcela da população brasileira.

Apêndice A

Termos técnicos

Os seguintes termos foram utilizados em inglês ao longo de todo o texto pela ausência de um termo melhor na língua portuguesa, e também por serem termos comumente utilizados desta maneira no contexto do desenvolvimento de software. A seguir, tem-se a definição de cada um desses termos:

Back-end: A parte de um sistema ou aplicativo que não é acessada diretamente pelo usuário, e é normalmente responsável por armazenar e manipular dados.

Branch: Ramificação feita a partir do repositório de um projeto, onde mudanças são feitas antes de serem submetidas ao repositório principal.

Bug: Defeito, falha ou erro no código de um programa que provoca seu mau funcionamento.

Build: Processo ou o resultado da conversão dos arquivos de código em artefatos de software que podem ser executados em um computador.

Captcha: Medida de segurança utilizada para distinguir computadores de pessoas.

Client-side: No contexto de programação, é um programa que é executado no computador do próprio cliente.

Commit: Operação de submissão de um conjunto de mudanças de código feitas em um repositório por meio da ferramenta Git.

Deploy: Processo de implantação e execução de um sistema em um determinado computador.

Framework: Estruturas de códigos genéricos que permitem o desenvolvimento de sistemas e aplicações.

Front-end: A parte de um sistema ou aplicativo que é vista e acessada diretamente pelo usuário, disponibilizando dados de forma estruturada.

Log: Registro contínuo de eventos e mensagens geradas pelo sistema, geralmente salvo em um arquivo com data e hora de cada mensagem.

Mockup: Representação visual utilizada por designers para apresentar uma ideia.

On-premise: Um servidor on-premise é aquele em que a própria equipe tem a responsabilidade de controlar processar suas aplicações de hardware e software.

Parser: Ferramenta de análise sintática e interpretações de textos, transformando-os em dados.

Pipeline: Uma cadeia de etapas que são realizadas até que a versão final de um software seja disponibilizado.

Plugin: Programa de computador usado para adicionar funcionalidades ao software em que é aplicado.

Pop-up: Janela que abre no navegador enquanto se navega em um site.

Pull request: Processo de análise das mudanças feitas em uma *branch* para serem submetidas ao repositório principal.

Push: Operação de envio de commits para um repositório remoto.

Query: No contexto da computação, é uma operação de consuta a um banco de dados.

Script: Conjunto de instruções a serem executadas em determinado aplicativo.

Web Crawler: Software utilizado para caminhar por páginas web e obter dados das mesmas.

Workflow: Sequência de processos que abrange desde o início ao fim de um trabalho.

Apêndice B

Scripts

Programa B.1 *Script* de inicialização do sistema, configurável.

```
1   #!/usr/bin/env bash
2
3   DEV_STRING="dev"
4   PROD_STRING="prod"
5
6   if [ "$#" -ne 2 ]; then
7      echo "" >&2
8      echo "Execute o script com o nome do instituto a ser configurado e o modo
           de execução (\"$DEV_STRING\"->desenvolvimento, \"$PROD_STRING\"->produç
           ão) como argumentos" >&2
9      echo "Ex: $ $0 fau $DEV_STRING" >&2
10     exit 1
11  fi
12
13  mode=$2
14
15  case $mode in
16     $DEV_STRING)
17        server_service="server-dev"
18        client_service="client-dev"
19     ;;
20     $PROD_STRING)
21        server_service="server"
22        client_service="client"
23     ;;
24     *)
25        echo "Modo de desenvolvimento '$mode' desconhecido. Modos disponíveis:
               [$DEV_STRING, $PROD_STRING]" >&2
26        exit 1
27     ;;
28  esac
29
30  export INSTITUTE=$1
31
```

cont ⟶

⟶ *cont*

```
32  cd ./src/
33
34  echo ""
35  echo "Montando imagens..."
36  docker compose --profile $mode build
37
38  echo ""
39  echo "Inicializando containers..."
40  docker compose --profile $mode up -d
41
42  sleep 60
43
44  echo ""
45  echo "Rodando migrations..."
46  docker compose --profile $mode exec $server_service python manage.py migrate
47
48  echo ""
49  echo "Aplicação está operante. Populando banco de dados..."
50  docker compose --profile $mode exec $server_service python util/populate/code/
        main.py -v
51  docker compose --profile $mode exec $server_service python util/populate/code/
        mongo_aggregate.py -v
52
53  echo ""
54  echo "Finalizado"
55
56  if [ "$mode" = "$DEV_STRING" ]; then
57     echo ""
58     echo "Capturando logs dos conteineres..."
59     docker compose logs --follow
60  fi
```

Programa B.2 *Script* de geração da imagem para *deploy* em produção.

```
1   #!/bin/sh
2
3   echo ""
4   echo "Script para criação de imagem de produção"
5
6   if [ "$#" -ne 2 ]; then
7      echo "" >&2
8      echo "Execute o script com o nome do instituto e host" >&2
9      echo "Ex: $ $0 fau localhost" >&2
10     exit 2
11  fi
12
13  export INSTITUTE=$1
14  HOST_ADDRESS=$2
15
16  DB_USER=${DB_NAME:-user}
17  DB_PASSWORD=${DB_PASSWORD:-password}
```

cont ⟶

⟶ *cont*

```
18  DB_NAME="${INSTITUTE}_db"
19
20  cd ./src/client
21  echo "REACT_APP_BACKEND_URL=http://$HOST_ADDRESS:8000" > .env
22
23  cd ../server
24  echo "
25  DB_USER=${DB_USER}\n\
26  DB_PASSWORD=${DB_PASSWORD}\n\
27  DB_HOST=database\n\
28  DB_PORT=5432\n\
29  \n\
30  AGGREGATE_DB_HOST=mongo\n\
31  AGGREGATE_DB_PORT=27017\n\
32  \n\
33  DJANGO_ALLOWED_HOSTS=$HOST_ADDRESS\n\
34  DJANGO_DEBUG=False\n\
35  DJANGO_LOGGING_LEVEL=info\n\
36  DJANGO_SECRET_KEY=${DJANGO_SECRET_KEY:-secretkey}\n\
37
38  EMAIL_CPQS_ABERTAS=cpqs-abertas@usp.br\n" > .env
39
40  cd ..
41
42  echo ""
43  echo "Montando imagens..."
44  docker compose --profile prod build client server --no-cache
45
46  cd ..
47
48  echo ""
49  echo "Salvando imagens..."
50  docker image save cpqs_client:latest cpqs_server:latest | pigz --fast > /tmp/
        cpqs_images.tar.gz
51  wait
```

Programa B.3 *Script* do *deploy* dos contêineres por SSH.

```
1   #!/bin/sh
2
3   echo ""
4   echo "Script para envio de imagem e subida de container"
5
6   if [ "$#" -ne 3 ]; then
7      echo "" >&2
8      echo "Execute o script com o nome do instituto, do host e o usuário" >&2
9      echo "Ex: $ $0 fau localhost ubuntu" >&2
10     exit 2
11  fi
12
13  INSTITUTE=$1
```

cont ⟶

⟶ *cont*

```
14  HOST_ADDRESS=$2
15  USER=$3
16
17  scp /tmp/cpqs_images.tar.gz $USER@$HOST_ADDRESS:/tmp/cpqs_images.tar.gz
18
19  ssh $USER@$HOST_ADDRESS bash -c "'
20
21  echo Running command: docker load /tmp/cpqs_images.tar.gz
22  docker load < /tmp/cpqs_images.tar.gz
23
24  echo Running command: rm /tmp/cpqs_images.tar.gz
25  rm /tmp/cpqs_images.tar.gz
26
27  echo Running command: git pull master
28  cd ~/cpqs-abertas
29  git checkout master
30  git pull
31
32  cd ./src
33
34  echo Running command: docker compose --profile prod down
35  INSTITUTE=$INSTITUTE docker compose --profile prod down
36
37  echo Running command: docker image prune -f
38  INSTITUTE=$INSTITUTE docker image prune -f
39
40  echo Running command: docker compose --profile prod up -d
41  INSTITUTE=$INSTITUTE docker compose --profile prod up -d
42  '"
```

Programa B.4 *Script* de *setup* de um novo servidor.

```
1   #!/bin/bash
2
3   ###
4   # Este script deve ser executado apenas uma vez ao criar um host novo para
        instalar o docker compose
5   ###
6
7   echo ""
8   echo "Script para setup do host CPqs Abertas"
9
10  if [ "$#" -ne 1 ]; then
11     echo "" >&2
12     echo "Execute o script com o nome do instituto" >&2
13     echo "Ex: $ $0 fau" >&2
14     exit 2
15  fi
16
17  INSTITUTE=$1
18
```

cont ⟶

⟶ *cont*

```
19  echo "export INSTITUTE=$INSTITUTE" >> ~/.bashrc
20  source ~/.bashrc
21
22  sudo apt update
23  sudo apt install \
24     ca-certificates \
25     curl \
26     gnupg \
27     lsb-release
28
29  sudo mkdir -p /etc/apt/keyrings
30  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -
        o /etc/apt/keyrings/docker.gpg
31
32  echo \
33    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.
        gpg] https://download.docker.com/linux/ubuntu \
34    $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
        /dev/null
35
36  sudo apt update
37  sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin
38
39  sudo service docker start
```

Programa B.5 *Script* de download dos Lattes usando Túnel SSH.

```
1   import ... #Ocultado para melhorar a legibilidade
2
3   logging... #Ocultado para melhorar a legibilidade
4
5   INSTITUTE = os.environ.get("INSTITUTE")
6   if INSTITUTE is None:
7      logging.error("Variável 'INSTITUTE' não está definida")
8      exit(1)
9
10
11  ssh_config = {
12     "ADDRESS": os.environ.get("SSH_ADDRESS"),
13     ... #Ocultado para melhorar a legibilidade
14  }
15
16
17  ime_lattes_db_config = {
18     "SERVER": os.environ.get("IME_LATTES_DB_SERVER"),
19     ... #Ocultado para melhorar a legibilidade
20  }
21
22  logging.info("Abrindo túnel SSH...")
23  ssh_tunnel = SSHTunnelForwarder(
24     ssh_address_or_host=ssh_config["ADDRESS"],
```

cont ⟶

⟶ *cont*

```
25      ssh_username=ssh_config["USER"],
26      ssh_password=ssh_config["PASSWORD"],
27      ssh_private_key=ssh_config["PRIVATE_KEY"],
28      remote_bind_address=(str(ime_lattes_db_config["SERVER"]), int(
            ime_lattes_db_config["PORT"]))
29  )
30  ssh_tunnel.start()
31
32  logging.info(f"Abrindo conexão com o Banco de docentes '{ime_lattes_db_config
        ['NAME']}'")
33  conn = pymssql.connect(
34     server="localhost",
35     user=ime_lattes_db_config["USER"],
36     password=ime_lattes_db_config["PASSWORD"],
37     database=ime_lattes_db_config["NAME"],
38     port=ssh_tunnel.local_bind_port
39  )
40  cursor = conn.cursor()
41  query_str = "SELECT * FROM LATTES_SENIORES"
42  logging.info(f"Executando query:\n{query_str}")
43  cursor.execute(query_str)
44
45  row = cursor.fetchone()
46  while row:
47     id_lattes = row[2]
48     zip_xml = row[3]
49     logging.info(f"Obteve currículo {id_lattes} do banco")
50     with open(f"/tmp/{id_lattes}.zip", "wb") as zip_xml_file:
51        logging.info(f"Escrevendo currículo {id_lattes} em um arquivo zip")
52        zip_xml_file.write(zip_xml)
53
54     with ZipFile(f"/tmp/{id_lattes}.zip", "r") as xml_zipfile:
55        logging.info(f"Extraíndo para {BASE_DIR}/util/{INSTITUTE}/lattes/{
              id_lattes}.xml")
56        xml_zipfile.extract(f"{id_lattes}.xml", f"{BASE_DIR}/util/{INSTITUTE}/
              lattes/")
57
58     try:
59        os.remove(f"/tmp/{id_lattes}.zip")
60     except:
61        pass
62     row = cursor.fetchone()
63
64  conn.close()
65  ssh_tunnel.close()
```

Programa B.6 *Script* de scraping dos IDs Lattes usando BeautifulSoup.

```
1   import #Ocultado para melhorar a legibilidade
2
3   DEPARTAMENTOS_IME = [
```

cont ⟶

⟶ *cont*

```
4      "DCC",
5      "MAE",
6      "MAT",
7      "MAP"
8   ]
9
10  try:
11     with open("./docentes.json", "r") as f:
12        docentes = json.loads(f.read())
13  except:
14     docentes = {}
15
16  try:
17     for departamento in DEPARTAMENTOS_IME:
18        docentes_url = f"https://www.ime.usp.br/{departamento}/docentes/"
19        logging.info(f"Accessing {docentes_url}")
20        docentes_page = requests.get(docentes_url)
21        soup_docentes_page = BeautifulSoup(docentes_page.content, "html.parser")

22        all_docentes = soup_docentes_page.find(id="todos")
23        docente_tag_list = all_docentes.find_all("div", class_="card m-1")
24
25        for docente in docente_tag_list:
26           docente_url = docente.find("h5").find("a")["href"]
27           sleep(0.5)
28           logging.info(f"Accessing {docente_url}")
29           try:
30              docente_page = requests.get(docente_url)
31           except requests.exceptions.MissingSchema:
32              docente_page = requests.get(f"http://{docente_url}")
33           if docente_page.status_code == HTTPStatus.NOT_FOUND:
34              logging.error(f"Erro ao processar '{docente_url}': Página não
                    encontrada")
35              continue
36           soup_docente_page = BeautifulSoup(docente_page.content, "html.parser
                ")
37           docente_lattes_tag = soup_docente_page.find("a", text = "Curriculum
                Lattes")
38           if not docente_lattes_tag:
39              logging.error(f"Erro ao processar '{docente_url}': Curriculo
                    lattes não encontrado")
40              continue
41           docente_lattes_url = docente_lattes_tag["href"]
42           lattes_url_pattern = r".*lattes.cnpq.br/(\d{16})"
43           match = re.match(lattes_url_pattern, docente_lattes_url)
44           if match is not None:
45              docente_lattes_id = match.group(1)
46              docentes[docente_lattes_id] = departamento
47              logging.info(f"Id {docente_lattes_id} salvo")
48           else:
49              logging.error(f"Erro ao processar '{docente_url}': URL do
                    Curriculo lattes incorreta")
```

cont ⟶

⟶ *cont*

```
50                continue
51    except:
52        logging.warning("Falha durante o scraping. Salvando dados já obtidos")
53        with open("./docentes.json", "w") as f:
54            json.dump(docentes, f)
```

Referências

[INCT-CPCT 2021] Instituto Nacional de Ciência e Tecnologia em Comunicação Pública da Ciência e Tecnologia. *O que os jovens brasileiros pensam da ciência e da tecnologia?* Survey. Instituto Nacional de Ciência e Tecnologia em Comunicação Pública da Ciência e Tecnologia, 2021. URL: https://www.inct-cpct.ufpa.br/wp-content/uploads/2021/02/LIVRO_final_web_2pag.pdf (citado na pg. 2).

[Science Group 2019] Web of Science Group. *Research in Brazil: Funding excellence.* Research. Clarivate Analytics, 2019. URL: https://jornal.usp.br/wp-content/uploads/2019/09/ClarivateReport_2013-2018.pdf (citado na pg. 1).