**UNIVERSIDADE VEIGA DE ALMEIDA- UVA**

**BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

CONSTRUÇÃO DE UM SISTEMA DE GESTÃO DE RECRUTAMENTO E SELEÇÃO (R&S)

LUCAS MORAES CAVALCANTE

**RIO DE JANEIRO**

2023

**UNIVERSIDADE VEIGA DE ALMEIDA- UVA**

LUCAS MORAES CAVALCANTE

Monografia apresentada ao curso de Sistemas de informação da Universidade Veiga de Almeida, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Douglas Ericson Marcelino de Oliveira, [DSc.]

CONSTRUÇÃO DE UM SISTEMA DE GESTÃO DE RECRUTAMENTO E SELEÇÃO (R&S)

**RIO DE JANEIRO**

2023

****

**UNIVERSIDADE VEIGA DE ALMEIDA- UVA**

**BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

LUCAS MORAES CAVALCANTE

CONSTRUÇÃO DE UM SISTEMA DE GESTÃO DE RECRUTAMENTO E SELEÇÃO (R&S)

Monografia apresentada como requisito parcial à conclusão do curso em Bacharel em Sistemas de Informação.

APROVADA EM:

CONCEITO: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

BANCA EXAMINADORA:

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROF. [DOUGLAS ERICSON MARCELINO DE OLIVEIRA], [DSc.]**

**ORIENTADOR**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROF. [NOME COMPLETO], [DSc.]**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROF. [NOME COMPLETO], [DSc.]**

**Coordenação de Sistemas de Informação**

Rio de Janeiro, 99 de Xxxxxx de 2023.

*Dedico este trabalho a Deus e aos meus pais que sempre se esforçaram em uma educação melhor para mim*

**“Epígrafe:** Os dados são apenas resumos de milhares de histórias - conte algumas dessas histórias para ajudar a tornar os dados significativos.”

Chip & Dan Heath

**Resumo**

O trabalho se iniciou ao perceber que algumas micro e pequenas empresas ainda não possuíam uma automatização que permitisse uma gestão eficiente nos processos de recrutamento e seleção por parte dos seus setores de RH.

Este trabalho foi conduzido com objetivos de desenvolver um *software* para o auxílio na gestão de recrutamento e seleção para um pequeno mercado visando informatizar os seus processos de recrutamento e seleção.

Foi utilizada a linguagem Java para construir uma aplicação de arquitetura *Web service* API REST. Como armazenador duradouro das informações processadas pela aplicação, foi utilizado o banco de dados PostgreSQL. Para realizar cache de dados, o Redis, um banco de dados em memória, foi escolhido. Para ajudar na construção do *Web service*, no acesso aos dados no PostgreSQL e no Redis, foi utilizado o framework Spring.

**Palavras-chave: Recursos humanos; Java; Recrutamento e seleção; API REST.**

***Abstract***

*The work began with the realization that some micro and small companies still did not have automation that would allow efficient management in the recruitment and selection processes by their HR sectors.*

*This work was carried out with the objective of developing a software to help in the management of recruitment and selection for a small market, aiming to computerize its recruitment and selection processes.*

*The Java language was used to build a REST API Web service architecture application. As a permanent store of the information processed by the application, the PostgreSQL database was used. To perform data caching, Redis, an in-memory database, was chosen. To help build the Web service, access data in PostgreSQL and Redis, the Spring framework was used.*

***Keywords: Human resources; Java; Recruitment and selection; REST API.***

**LISTA DE ILUSTRAÇÕES**

[Figura 1 - Procedimentos de R&S mais utilizados 27](#_Toc147081265)

[Figura 2 - Procedimentos de R&S considerados mais importantes 28](#_Toc147081266)

[Figura 3 - A TI pode auxiliar no processo de R&S 29](#_Toc147081267)

[Figura 4 - Diagrama de Caso de Uso 35](#_Toc147081268)

[Figura 5 - Diagrama de Classe 37](#_Toc147081269)

[Figura 6 - Diagrama de Máquina de Estados Etapa 39](#_Toc147081270)

[Figura 7 - Diagrama de Máquina de Estados Candidatura 40](#_Toc147081271)

[Figura 8 - Diagrama de Máquina de Estados Processo Seletivo 41](#_Toc147081272)

[Figura 9 - Diagrama de Entidade Relacionamento 43](#_Toc147081273)

[Figura 10 - *Wireframe* tela de *login* 45](#_Toc147081274)

[Figura 11 - *Wireframe* tela incial para um candidato 45](#_Toc147081275)

[Figura 12 - *Wireframe* tela de processos seletivos em aberto 46](#_Toc147081276)

[Figura 13 - *Wireframe* tela de detalhes de um processo seletivo 47](#_Toc147081277)

[Figura 14 - *Wireframe* tela da trilha de etapas de um processo seletivo 48](#_Toc147081278)

[Figura 15 - Arquitetura interna aplicação servidora 50](#_Toc147081279)

[Figura 16 - Fluxograma candidato 52](#_Toc147081280)

[Figura 17 - Fluxograma recrutador 53](#_Toc147081281)

[Figura 18 - Fluxograma administrador 54](#_Toc147081282)

**LISTA DE TABELAS**

[Tabela 1 - Requisitos funcionais usuário não autenticado 30](#_Toc145524233)

[Tabela 2 - Requisitos funcionais para um recrutador 30](#_Toc145524234)

[Tabela 3 - Requisitos funcionais para um candidato 32](#_Toc145524235)

[Tabela 4 - Requisitos funcionais para um administrador 32](#_Toc145524236)

[Tabela 5 - Requisitos não funcionais 33](#_Toc145524237)

**LISTA DE ABREVIATURAS E SIGLAS**

ACID Atomicidade, Consistência, Isolamento e Durabilidade

ANSI *American National Standards Institute*

API Interface de Programação de Aplicação

HTTP *Hypertext Transfer Protocol*

IBM *International Business Machines Corporation*

ISO *International Organization for Standardization*

JVM Máquina Virtual Java

JWT *Json Web Token*

NoSQL Banco de Dados Não Relacionais

REST Transferência de Estado Representacional

RF Requisito Funcional

RFN Requisito Não Funcional

RH Recursos Humanos

RJ Rio de Janeiro

RPC Chamada Remota de Procedimento

R&S Recrutamento e Seleção

SGBD Sistema de Gerenciamento de Banco de Dados

SOAP Protocolo Simples de Acesso a Objetos

SQL Linguagem de Consulta Estruturada

S.A. Sociedade Anônima

TI Tecnologia da Informação

UML Linguagem de Modelagem Unificada

**Sumário**

[**1. INTRODUÇÃO** 13](#_Toc150465154)

[1.1 Contexto 13](#_Toc150465155)

[1.2 Problema 13](#_Toc150465156)

[1.3 Justificativa 14](#_Toc150465157)

[1.4 Objetivo Geral 15](#_Toc150465158)

[1.4.1 Objetivos Específicos 15](#_Toc150465159)

[1.5 Delimitação do estudo 16](#_Toc150465160)

[1.6 Metodologia 16](#_Toc150465161)

[1.7 Estrutura do trabalho 17](#_Toc150465162)

[**2. EMBASAMENTO TEÓRICO** 18](#_Toc150465163)

[2.1 Resumo do sistema 18](#_Toc150465164)

[2.2. Arquitetura 19](#_Toc150465165)

[2.3. Tecnologias 20](#_Toc150465166)

[2.3.1 Java 20](#_Toc150465167)

[2.3.2 PostgreSQL 21](#_Toc150465168)

[2.3.3 Redis 22](#_Toc150465169)

[2.3.4 *Spring* 22](#_Toc150465170)

[2.4 Modelagem 23](#_Toc150465171)

[**3. CONSTRUÇÃO DO SISTEMA** 25](#_Toc150465172)

[3.1. Análise de soluções correlatas 25](#_Toc150465173)

[3.2. Levantamento e análise de requisitos 26](#_Toc150465174)

[3.2.1. Análise dos resultados da pesquisa 27](#_Toc150465175)

[3.2.2. Requisitos funcionais 29](#_Toc150465176)

[3.2.3. Requisitos não funcionais 33](#_Toc150465177)

[3.3. Modelagem do sistema 33](#_Toc150465178)

[3.3.1. Diagrama de caso de uso 34](#_Toc150465179)

[3.3.2. Diagrama de classes 35](#_Toc150465180)

[3.3.3. Diagrama de máquina de estados 38](#_Toc150465181)

[3.3.4. Diagrama de entidade relacionamento 41](#_Toc150465182)

[3.3.5. *Wireframe* 43](#_Toc150465183)

[**4. IMPLEMENTAÇÃO DO SISTEMA** 49](#_Toc150465184)

[4.1. Aplicação *backend* 49](#_Toc150465185)

[4.1.1. Documentação API 51](#_Toc150465186)

[4.2. Aplicação *frontend* 51](#_Toc150465187)

[**5. CONCLUSÃO** 55](#_Toc150465188)

[5.1. Trabalhos futuros 55](#_Toc150465189)

[**REFERÊNCIAS** 57](#_Toc150465190)

[**APÊNDICE A – QUESTIONÁRIO DE LEVANTEMENTO DE PERFIL DAS PEQUENAS EMPRESAS REFERNTE A GESTÃO DO PROCESSO DE R&S** 59](#_Toc150465191)

[**APÊNDICE B – QUESTIONÁRIO DE LEVANTEMENTO DE PERFIL DOS PROFISSIONAIS DE RH** 61](#_Toc150465192)

# **1. INTRODUÇÃO**

Os tópicos abaixo possuem o intuito de contextualizar um problema vivenciado atualmente nas empresas que não possuem um setor de recursos humanos estruturado mais especificamente na área de recrutamento e seleção e apresentar uma solução tecnológica para contornar o problema apresentado.

## 1.1 Contexto

A Tecnologia da Informação – TI – tornou-se uma ferramenta indispensável para todos os segmentos sociais: no setor privado e público, na educação, no lazer e entretenimento e na comunicação. A TI afeta a maneira de como os conhecimentos são adquiridos, impactando nas escolhas a serem tomadas e seguidas por cada indivíduo.

Albertin (1999) e Meirelles (1999) apontam que ao longo dos últimos anos, a tecnologia da informação ganhou espaço e começou a ser utilizada mais intensamente por organizações em todo o planeta. Através de grandes investimentos feitos na TI, as organizações cada vez mais a utilizam como ferramenta para o aumento da competitividade, conforme destacam os autores.

Devido a essa importância que a TI representa hoje, muitas empresas buscaram automatizar seus negócios a fim de potencializar o aumento da produtividade dos seus processos internos e fazer com que eles fiquem mais organizados e eficientes. Uma das áreas que vem passando por esse processo de automatização é o setor de recursos humanos - RH.

## 1.2 Problema

Lawler (1995) e Mohrman (1995) já comentavam sobre a necessidade de reformulação e gestão de informação no setor de recursos humanos, proporcionando, assim, informações sobre as condições da organização humana. De acordo com esses autores, a administração de RH precisa estar atenta aos avanços tecnológicos, no sentido de se comprometer a utilizar os sistemas da informação, a fim de melhorar a qualidade do serviço prestado.

Conforme Chiavenato (2010), a estrutura de recursos humanos molda talentos e realiza a gestão do capital humano da organização, através de processos integrados com o setor de recrutamento e seleção – R&S. O processo de R&S é uma operação dentro da área de recursos humanos, que exerce uma grande importância para os demais setores nas companhias. A partir dessa informação de Chiavenato, pode-se entender que o departamento de R&S desempenha uma função relevante, pois é através dele que se mapeia e recruta novos talentos para a organização.

Hoje em dia, pode-se perceber algumas plataformas que realizam a gestão de recrutamento e seleção de pessoas. Essas plataformas visam trazer para o setor de RH alguns benefícios, como poder estruturar processos seletivos de forma ágil, os quais possam ser gerenciados de forma simples e segura. Existem sistemas que são focados em *softwares* de gestão de recrutamento e seleção.

Uma empresa bastante famosa é a Gupy[[1]](#footnote-2), que possui uma estrutura bem robusta com várias funcionalidades para esses casos de gestão de um processo seletivo. Muitas empresas de grande porte, como Globo Comunicação e Participações S.A.[[2]](#footnote-3), Natura Cosméticos S/A[[3]](#footnote-4) e Americanas S.A.[[4]](#footnote-5), utilizam essa plataforma para seus processos de seleção, entretanto não são todas as empresas que fazem uso de um *software* sofisticado como esse. Algumas empresas de micro a pequeno porte, quando vão realizar um processo seletivo para preencher uma posição dentro do seu quadro de funcionários, não possuem um sistema que realiza a gestão de recrutamento e seleção de candidatos, o que faz com que a gestão e o acompanhamento desses processos sejam feitos de forma manual por não ter um *software* que atenda a demanda e a realidade dessas micro e pequenas empresas.

## 1.3 Justificativa

Com modernização do RH e de uma gestão de pessoas eficiente, faz-se necessário utilizar as ferramentas fornecidas pela TI. Segundo Pontes (2010, p. 13):

Vivemos hoje num mundo em que a tecnologia propicia grande quantidade de informações instantaneamente. Pessoas podem ter fácil acesso a informações de mercados, empresas e de seus concorrentes

Assim, as organizações necessitam de se atualizar e de se adequar aos avanços tecnológicos cada vez mais dinâmicos, causando impactos na empresa. O autor ainda complementa que:

É um paradoxo, mas, quanto maior for a velocidade dos avanços tecnológicos, maior a necessidade de mudanças nas próprias empresas que criaram tais avanços tecnológicos, e o processo de inovação continua, de forma cada vez mais acelerada. A empresa que não dispõe de flexibilidade para atualização fica fora do mercado. Nesse processo pessoas que pensem nas estratégias, que desenvolvam novos produtos são essenciais (Pontes, 2010, p. 20).

Nesse momento de avanços tecnológicos, a automatização dos processos de RH faz com que esses processos se tornem mais rápidos, auxiliando na tomada de decisões. Pontes (2010) traz a ideia de que o uso da internet auxilia fortemente o recrutamento, pelo fato de as informações poderem ser acessadas em tempo real, existe ganho de tempo e eficiência nos processos, reduzindo tarefas desnecessárias.

Devido a essas contextualizações, a partir das ideias de Pontes, propõe-se elaborar um sistema de gestão de recrutamento e seleção. A motivação de desenvolver esse tema é ajudar a acelerar o processo de transformação tecnológica no setor de recursos humanos mais precisamente na área de recrutamento e seleção.

## 1.4 Objetivo Geral

O tema a ser trabalhado criou e disponibilizou uma solução tecnológica em formato de API chamada *recruitment-selection*. Essa solução tem como finalidade informatizar o processo de gestão de R&S de candidatos para um pequeno mercado de bairro situado na cidade do Rio de Janeiro-RJ, no bairro Lins de Vasconcelos.

Além disso, o objetivo desse trabalho é descrever as características do *software* apresentado para a solução do problema, expor todo o processo de construção do sistema desde a fase de levantamento e análise de requisitos até os relatos de como o sistema está organizado internamente à nível de código.

### 1.4.1 Objetivos Específicos

Para chegar-se ao o objetivo principal, o mesmo pode ser separado em perspectivas menores e específicas para que seja possível tanto mapear quanto tratar os problemas de maneira mais eficiente. Abaixo estão listados os objetivos específicos do trabalho:

* Buscar a utilização de tecnologias consolidadas e que possuem uma boa eficiência para a construção de um aplicativo *web*;
* Durante a construção da solução tecnológica, com o objetivo de garantir uma entrega mais assertiva para o cliente, deve-se seguir alguns dos processos definidos pela engenharia de *software*;

## 1.5 Delimitação do estudo

O trabalho apresentado delimitou-se a apresentar um problema vivenciado nas empresas que não possuem as suas disposições um *software* que vise realizar a gestão dos processos de recrutamento e seleção de candidatos para as vagas de trabalhado ofertadas nessas empresas. Além disso, o trabalho também tem como intuito expor todo o processo de construção do *software recruitment-selection*, desde a escolha das tecnologias utilizadas até a implementação do aplicativo *backend*.

Nesse trabalho, não será abordado o processo de escolhas de tecnologias e a implantação à nível de código do aplicativo *frontend*, porém será exposto o processo de desenvolvimento dos protótipos das telas de navegação e como irá funcionar a transição dos fluxos no aplicativo *frontend* para o usuário entender o ponto de partida de cada tela construída.

## 1.6 Metodologia

Com o intuito de aumentar o conhecimento sobre o problema a ser destacado e entender como as soluções atuais visam corrigir esses problemas, neste estudo, a metodologia utilizada na pesquisa foi a do modelo exploratório. Através dessa metodologia, realizou-se a pesquisa bibliográficas através de defesas de teses, artigos disponibilizados e análise de soluções (*softwares*) correlatas já existentes no mercado.

Além disso, para realizar a construção do *software* *recruitment-selection* que visa corrigir o problema destacado, utilizou-se algumas técnicas e ferramentas do conceito da engenharia de *software*. Alguns processos como entrevistas com os *stakeholders* do produto, o levantamento e a análise dos requisitos e a modelagem do sistema através da linguagem UML, são alguns dos métodos da engenharia de *software* que foram utilizados para ajudar no desenvolvimento da solução proposta nesse trabalho.

## 1.7 Estrutura do trabalho

O conteúdo a ser discutido neste trabalho estrutura-se em cinco capítulos: Introdução, Embasamento Teórico, Construção do Sistema, Implementação do Sistema e Conclusão.

A Introdução buscou contextualizar um problema vivenciado, atualmente, em algumas empresas que não possuem uma gestão de R&S, justificar o tema a ser desenvolvido nesse trabalho, expor os objetivos gerais e específicos a serem atingidos, delimitar a atuação do estudo do trabalho e apresentar as metodologias utilizadas para a construção do referido trabalho.

O Embasamento Teórico tem como finalidade apresentar um resumo do funcionamento do sistema *recruitment-selection* desenvolvido neste trabalho. Além disso, tem por objetivo apresentar e explicar sobre a arquitetura e as tecnologias utilizadas para a construção do *software* assim como explanar de forma resumida quais foram os passos seguidos para iniciar o processo de modelagem do sistema.

O objetivo do capítulo de Construção do Sistema é apresentar de forma detalhada todo o processo de modelagem do *recruitment-selection*. Esse item buscou apresentar o processo de análise de soluções correlatas, detalhar o processo de levantamento e análise de requisitos, expor os requisitos funcionais e não funcionais da aplicação e apresentar como se deu a modelagem do sistema através de diagramas e *wireframes*.

O capítulo de Implementação do Sistema tem o objetivo de descrever como o sistema *recruitment-selection* se encontra estruturado a nível de código e de arquitetura interna, além de expor os seus fluxos internos.

A Conclusão tem como função apresentar os resultados obtidos com a construção deste trabalho e apresentar os trabalhos futuros que irão agregar mais valor ao trabalho desenvolvido nesse documento.

# **2. EMBASAMENTO TEÓRICO**

Os tópicos a seguir têm como objetivos explicar a arquitetura utilizada nas tecnologias, nas modelagens do sistema *recruitment-selection* e quaisquer outros pontos relevantes a serem destacados e dar uma breve descrição do sistema.

## 2.1 Resumo do sistema

O trabalho a ser apresentado tem como foco informatizar o processo de gestão de recrutamento e seleção de candidatos para um pequeno mercado de bairro situado na cidade do Rio de Janeiro-RJ, no bairro Lins de Vasconcelos.

Em linhas gerais, o *software* *recruitment-selection* apresenta funcionalidades que permite que sejam cadastrados processos seletivos para preencher determinadas vagas em aberto em uma organização. Esses processos seletivos serão compostos por uma trilha de etapas e o responsável pelo processo, no momento de criar o processo seletivo no sistema, poderá montar a trilha de etapas da forma que achar melhor.

Após a criação do processo seletivo, um usuário que possui conta como candidato no sistema, poderá visualizar os processos seletivos em aberto e aqueles nos quais ele está inscrito. Além de visualizá-los, ele também poderá realizar a sua inscrição nos processos seletivos que achar interessante. A aplicação também permite que o candidato realize algumas etapas de um processo seletivo dentro da própria plataforma, como realização de testes teóricos e envio de arquivos. Entretanto, etapas como vídeo entrevista deverão ser realizadas em plataformas externas que suportem o *streaming* de vídeo chamadas. Nestes tipos de etapas externas, o candidato terá, à disposição, na própria plataforma, a data/hora e o *link* de onde será realizada essa etapa externa.

Outra funcionalidade disponibilizada pelo *recruitment-selection* é que os responsáveis pelos processos seletivos poderão registrar resultados para cada etapa realizada por um candidato e poderão também visualizar uma classificação de melhores candidatos de um processo seletivo, cuja classificação será montada, de acordo com as médias das avaliações que foram registradas para cada candidato, ao longo da trilha de etapas de um processo seletivo. Além dessas funcionalidades apresentadas, as demais serão descritas ao longo desse trabalho, na Seção 3.2, na parte de requisitos funcionais e não funcionais.

## 2.2. Arquitetura

O sistema desenvolvido segue uma arquitetura Cliente Servidor com o formato *Web service*, cujo servidor é uma API REST que tem a função de receber solicitações de uma aplicação cliente e devolver a resposta para o cliente. O protocolo de comunicação utilizado na construção da API será o HTTP. Essa arquitetura apoia-se nas ideias de Abinader, Menéndez, Galindo Junior, Dias Rocha, Souza Maciel e Fielding que serão mencionados abaixo.

Abinader (2006) informa que inúmeras ferramentas e tecnologias surgiram nos últimos anos, destinadas ao desenvolvimento de aplicações para a *internet*. Entretanto, nenhuma é completa e todas possuem suas limitações e particularidades, como, por exemplo, a troca de informação de forma simples e automática.

A necessidade da integração entre aplicações, a utilização unificada de processos encontrados em diferentes sistemas e escritos em diferentes linguagens são alguns exemplos de carências encontradas em empresas nos dias de hoje. A fim de sanar estas questões, a tecnologia dos *Web services* foi criada, permitindo, assim, disponibilizar formas de integrar sistemas distintos, modularizar serviços, capacitar a integração e consumo de informações (Menéndez, 2002).

A partir das ideias de Menéndez, pode-se entender que através da arquitetura *Web service* é possível trocar informação entre dois sistemas, sem necessidade de recolher informação detalhada sobre o funcionamento de cada sistema. Os *Web services* permitem integrar qualquer tipo de aplicação, independentemente do sistema operacional (Windows, Linux, MacOS, entre outras) e linguagens de programação (Java, C#, Go, etc.) utilizadas.

Como os *Web services* possuem inúmeras abordagens tecnológicas para serem implementados, como SOAP, RPC e REST, a tecnologia escolhida para trabalhar usando a arquitetura *Web service* foi o modelo REST.

Segundo Galindo Junior, Dias Rocha e Souza Maciel (2021, p. 2):

API REST é uma solução baseada em redes para que os componentes que formam o sistema possam interagir entre si. O desacoplamento desse sistema proporciona vantagens sobre o desenvolvimento, pois haverá apenas um *backend* se comunicando com vários *frontends*, trazendo impacto positivo na engenharia de *software*, reduzindo o tempo e os custos de desenvolvimento.

Outro fator positivo do REST é a boa aderência ao conhecido protocolo HTTP, como o principal meio de comunicação (Fielding, 2000).

A partir das ideias de Abinader, Menéndez, Galindo Junior, Dias Rocha, Souza Maciel e Fielding, pode-se concluir que a escolha de uma arquitetura *Web service* aliada ao modelo REST permite que uma única aplicação que contenha tanto a lógica do negócio quanto a tecnologia de visualização possa ser separada em duas: uma aplicação com foco no *frontend*, contendo as telas de navegação, e outra aplicação com foco no *backend*, contendo toda a lógica do negócio.

Devido a essa possível separação do *frontend* com o *backend*, a motivação de trabalhar em uma arquitetura *Web service* com o modelo REST dá-se pela aplicação *backend* conseguir evoluir de forma independente de qualquer outra aplicação cliente, desde que os contratos de integração definidos anteriormente sejam mantidos os mesmos para garantir a integração de aplicações clientes que já estejam consumindo informações de um *Web service* a fim de unificar um processo.

O padrão REST, por ser extremamente difundido e bem aceito no mercado e na comunidade de desenvolvimento, foi escolhido devido à sua manipulação por usar conceitos bem diretos e definidos, associados à sua simplicidade e segurança.

## 2.3. Tecnologias

Todas as tecnologias escolhidas – Java, PostgreSQL, Redis e Spring – têm como base a sua forte consolidação no mercado e na comunidade de desenvolvimento de *software*, com vastas documentações e tutoriais disponíveis de forma gratuita na *internet*.

### 2.3.1 Java

Como o sistema *recruitment-selection* foi construído para ser uma aplicação predominantemente orientada a objetos, foi escolhida a linguagem de programação Java para auxiliar na escrita do *software*.

Horstmann e Cornell (2010) comentam que o Java nunca foi apenas uma linguagem qualquer. Atualmente existem muitas linguagens de programação disponíveis e, poucas, talvez, nem mesmo se aproximam do poder do Java. Ela é uma plataforma integral, com uma biblioteca imensa, com uma grande quantidade de códigos disponíveis para a reutilização e um ambiente de execução que fornece serviços com segurança, portabilidade, para vários tipos de sistema operacionais.

Segundo Gradvohl (2008, p. 4):

A arquitetura ou plataforma Java tem, basicamente, dois componentes: a máquina virtual Java (JVM) e a interface de programação de aplicações (“*Application Programming Interface*” - API). A JVM, como o próprio nome diz, emula um ambiente computacional. A máquina virtual é a principal responsável pela portabilidade que a plataforma Java provê, pois, uma vez instalada em um ambiente computacional real, qualquer programa Java pode ser executado sobre a JVM.

A linguagem Java é uma tecnologia criada nos anos 90. Hoje essa linguagem é gerida por uma grande corporação de alto conceito na comunidade de desenvolvimento. A linguagem Java possui como seu principal paradigma a orientação a objetos, porém é também uma linguagem multiparadigma, permitindo a combinação de outros paradigmas com a orientação a objetos. Essas características associadas à capacidade de ser uma linguagem multiplataforma e de rodar em qualquer sistema operacional que suporte a JVM, de forma segura e de boa performance, são alguns dos fatores que permitiram a escolha dessa linguagem.

### 2.3.2 PostgreSQL

Para realizar o armazenamento duradouro das informações processadas pela aplicação *recruitment-selection*, foi escolhido o banco de dados relacional PostgreSQL. Esse banco usa a linguagem SQL para executar operações.

Para Elmasri e Navathe (2011) o SQL é uma linguagem de consulta de banco de dados bem vasta: possui comandos para definição de banco de dados, consultas, atualizações e remoções. “SQL é uma linguagem de definição e de manipulação de dados relacionais, desenvolvida nos laboratórios da IBM (*International Business Machines*) nos anos 70 e hoje padronizada pelos comitês ISO/ANSI” (GUIMARÃES, 2003, p. 99).

A partir da ideia de Elmasri e Navathe, o motivo de escolher um banco de dados SQL para o projeto deu-se devido à possibilidade de trabalhar com uma poderosa linguagem consolidada e reconhecida na comunidade, a qual permite realizar filtros de consultas das mais variadas formas.

Segundo Eduardo Boeing e Ricardo Boeing (2018, p. 22):

A Comunidade Brasileira de PostgreSQL (2016) afirma que ele é totalmente compatível com ACID, tem suporte completo a chaves estrangeiras, junções (JOINs), visões, gatilhos e procedimentos armazenados (em múltiplas linguagens). Suporta também o armazenamento de objetos binários, incluindo figuras, sons ou vídeos, e possui uma vasta documentação.

A partir da documentação oficial, pode-se entender que o PostgreSQL é um sistema de banco de dados *open source* e possui tanto o modelo relacional quanto o objeto-relacional. Ele tem quase 2 décadas de presença na comunidade e uma arquitetura comprovada que ganhou uma forte reputação de confiabilidade, integridade de dados e correção.

### 2.3.3 Redis

Como mecanismo para realização de cache na aplicação *recruitment-selection*, foi utilizado o banco de dados NoSQL Redis.

Souza e Santos (2015) descrevem o Redis como um banco de dados que utiliza um algoritmo interno de chave-valor, com a característica de que todas as suas operações são feitas em memória e, periodicamente, os dados são salvos em disco.

O Redis é um dos principais bancos de dados quando se trata de cache de dados nas aplicações. Com uma ótima performance, comandos simples e diretos em relação à manipulação de dados e, já possuindo uma boa integração ao *framework* *Spring*, consolidou a escolha dele como a tecnologia de cache para as informações.

Pela documentação oficial do Redis e pelas ideias de Souza e Santos pode-se concluir que o Redis é um armazenamento de estrutura de dados em memória, usado como banco de dados em cache.

### 2.3.4 *Spring*

Para auxiliar no desenvolvimento da camada *web* escrita em Java, no acesso aos dados no PostgreSQL através da linguagem de consulta SQL e na gestão de cache utilizando o Redis, foi escolhido o *framework* *Spring*. Essa ferramenta tem por objetivo tornar a programação Java mais ágil, fácil, simples e segura.

Segundo Fonseca Júnior (2007, p. 4):

Atualmente, inúmeras são as soluções de frameworks passíveis de utilização em aplicações *web*, porém, o processo de configuração destas soluções para trabalho em conjunto ainda é oneroso demais. Logo, uma das grandes contribuições do *framework Spring* está no fato de configurar de forma transparente as soluções tecnológicas escolhidas para uso pela aplicação, liberando os desenvolvedores para um foco exclusivo em questões relacionadas ao negócio ou área fim da empresa ou organização.

O *Spring* é o principal *framework* utilizado em sistemas *web* desenvolvidos em Java. A sua alta popularidade na comunidade aliada a bastantes projetos, funcionalidades e tecnologias já integrados ao *Spring* torna esse o principal *framework*, quando trabalhamos com o Java para a *web*. O *Spring* é uma ferramenta de código aberto onde constantemente sofre atualizações para adicionar novas funcionalidades ao seu baralho.

## 2.4 Modelagem

Como ponto de partida para a modelagem do sistema *recruitment-selection*, foram realizados os processos de análise de soluções correlatas e realização de entrevistas com os especialistas do negócio e com as potenciais partes interessadas do produto. Essas etapas são importantes processos na modelagem do sistema porque buscam orientar os objetivos principais que a aplicação precisa atingir para conseguir entregar valor diante de outras soluções já existentes e atingir os desejos dos usuários a utilizar o sistema.

Após a realização dos processos de análise de soluções correlatas e do processo de entrevistas, deu-se início às etapas de levantamento e análise dos requisitos. Essas etapas são importantes processos na construção de um *software*, pois buscam obter informações detalhadas de como o cliente espera que a futura aplicação irá se comportar para atender às suas demandas. Essas análises são cruciais para facilitar os processos seguintes, garantindo que o sistema atenda às necessidades e expectativas do cliente de forma eficiente. É através desses processos que se define o que a aplicação deve ser capaz de realizar e como deve ser o seu comportamento em variados fluxos de execução.

Após a realização da etapa de levantamento e análise dos requisitos, foi iniciado o processo de mapeamento e distinção entre os requisitos funcionais e os requisitos não funcionais que a aplicação irá ofertar. Em resumo, os requisitos funcionais são funcionalidades específicas que a aplicação deve disponibilizar, ou seja, uma ação que o *software* precisará executar. Já os requisitos não funcionais referem-se às características ou qualidades do sistema, como desempenho, segurança, facilidade de uso, confiabilidade, e assim por diante.

Depois de ter realizado o processo de mapeamento e distinção dos requisitos funcionais e não funcionais, realizaram-se a modelagem dos diagramas da aplicação *backend* através da UML e a modelagem dos protótipos de telas da aplicação *frontend* através do conceito de *wireframes*, para descrever e facilitar o desenvolvimento do sistema. A linguagem UML possui bastante destaque no trabalho de desenvolvimento de sistemas. A UML é uma linguagem visual que possibilita aos desenvolvedores criarem uma representação visual e uma forma de comunicação eficaz às diferentes etapas da construção de uma aplicação desde a análise e design até a implementação e testes. O *wireframe* é um diagrama visual que possui o objetivo de ser um rascunho inicial para representar o *layout* de uma interface visual de um site ou de aplicativo.

## 

# **3. CONSTRUÇÃO DO SISTEMA**

Através de algumas metodologias da engenharia de *software*, nesse capítulo será apresentado como foi o início do processo de construção do *software* *recruitment-selection*.

A construção do sistema iniciou-se a partir das etapas de análise de soluções correlatas e da realização de entrevistas com os especialistasdo negócio e com as partes interessadas do produto. Após essa etapa, deu-se início ao processo de levantamento e análise dos requisitos, e, por último, foram realizadas as modelagens do sistema *backend* através da linguagem UML e as representações dos esboços das principais telas do aplicativo *frontend* através do conceito de *wireframes*.

## 3.1. Análise de soluções correlatas

A análise de trabalhos correlatos é crucial para a construção de uma solução mais aderente às expectativas do cliente. A utilização desse método possibilita identificar pontos fortes e fracos das soluções disponíveis no mercado e, assim, viabilizar uma opção mais atrativa, com o intuito de não cometer os mesmos deslizes, aperfeiçoando as soluções já disponíveis ou desenvolvendo novas funcionalidades e conceitos.

Durante a pesquisa para a construção deste trabalho, encontraram-se *softwares* que visavam automatizar a gestão de alguns processos da área de R&S. Entretanto, algumas soluções acabavam por não automatizar outras funcionalidades relevantes da área de R&S. Ao deixar de lado a automatização dessas funcionalidades, a solução analisada poderia não alcançar as expectativas quanto a criação de uma plataforma eficiente e unificada na gestão de processos de recrutamento. Outro ponto destacado foi a observação de que algumas soluções robustas já desenvolvidas tinham como foco e objetivo ofertar as suas funcionalidades para grandes empresas no mercado. Por consequência, pequenas empresas ou de baixo capital acabavam por não conseguir usufruir dessas plataformas.

Na análise de trabalhos correlatos não foi possível identificar uma solução no mercado que buscasse atender as demandas de pequenos negócios, ofertando um sistema eficaz para as principais funcionalidades e que tivesse um baixo custo, porém foi possível encontrar duas soluções interessantes as quais serviram de base para a análise de requisitos da construção desse trabalho.

A primeira solução analisada foi a da organização corporativa Catho Online Ltda[[5]](#footnote-6). Essa empresa possui o objetivo de ser um site de classificados de empregos, ofertando determinadas funcionalidades mediante assinaturas de planos mensais para usuários que queiram enviar currículos para as vagas disponibilizadas na plataforma. Ao explorar os recursos desse sistema para mapear as funcionalidades da plataforma, observa-se, logo de início, um ponto negativo: a Catho realiza a venda de planos mensais, oferecendo vantagens para usuários assinantes se comparados com usuários não assinantes, gerando assim um possível desequilíbrio originado pela própria aplicação na concorrência entre os candidatos por uma vaga. Essa atitude, por consequência, acaba indiretamente coagindo o usuário que queira se candidatar as vagas ofertadas na plataforma e queira aumentar as suas chances de contratação a realizar uma assinatura mensal, gerando custos monetários a mais para o navegante do sistema.

A segunda solução analisada foi a plataforma da empresa Gupy. Essa corporação oferta um sistema de R&S bem estruturado com diversas funcionalidades de alto valor agregado para os clientes, porém, ao cruzar algumas análises entre a Catho Online Ltda e a Gupy, percebeu-se que muitas empresas de determinados perfis que ofertavam suas vagas na Catho Online Ltda acabavam por não ofertar essas mesmas vagas na Gupy. A partir dessa última análise, observou-se a Gupy, ao contrário da Catho, era uma plataforma mais robusta e disponibilizava funcionalidades de alto valor agregado para os usuários navegantes da plataforma. Contudo, para os clientes do sistema, o perfil de atuação da Gupy aparentava ter como foco atuar com clientes já bem estruturados e estabilizados no mercado.

## 3.2. Levantamento e análise de requisitos

Para a construção do levantamento de requisitos deste trabalho, além da pesquisa bibliográfica e trabalhos correlatos, foram utilizados questionários com o objetivo de buscar entender e mapear as principais funcionalidades que um sistema de recrutamento e seleção deve ofertar. Foram aplicados dois questionários, com objetivo de analisar dois perfis, o perfil dos gestores dos pequenos negócios e o perfil dos profissionais de RH.

A estratégia de divulgação para coleta dos dados foi utilizar a rede social *LinkedIn* para entrar em contato com profissionais do setor de RH e realizar contatos presenciais com os gestores dos pequenos negócios.

### 3.2.1. Análise dos resultados da pesquisa

Como intuito de identificar os objetivos propostos na visão do gestor de uma pequena empresa e na visão do profissional de RH, foram apresentados 2 tipos de questionários contendo cada um o total de 8 questões, sendo todas as questões fechadas, os modelos utilizados para a entrevista para cada visão podem ser encontrados respectivamente em APÊNDICE A e APÊNDICE B.

A seguir, serão apresentados os detalhamentos das respostas das questões mais relevantes para o levantamento de requisitos contidas nos questionários aplicados às empresas e aos profissionais. Ao todo foram coletadas 3 amostras de perfis de empresas e 4 amostras para o questionário aplicado aos profissionais.

Figura 1 - Procedimentos de R&S mais utilizados

Fonte - O Autor (2023)

A partir do gráfico apresentado na Figura 1, pode-se concluir que os procedimentos mais utilizados no momento de analisar o perfil de uma candidatura são: análise de currículos, entrevista com gestor e avaliações técnicas.

Figura 2 - Procedimentos de R&S considerados mais importantes

Fonte - O Autor (2023)

A partir do gráfico apresentado na Figura 2, pode-se concluir que os procedimentos considerados mais importantes a serem utilizados no momento de analisar o perfil de uma candidatura são: análise de currículos, entrevista com gestor e avaliações técnicas.

Figura 3 - A TI pode auxiliar no processo de R&S

Fonte - O Autor (2023)

A partir do gráfico apresentado na Figura 3, pode-se concluir que de forma unânime, os entrevistados concordaram que a Tecnologia da Informação pode aumentar a eficiência de um processo de recrutamento e seleção.

Após a finalização da pesquisa bibliográfica, do processo de análise dos trabalhos correlatos e da etapa de entrevistas, foi possível mapear 21 funcionalidades que o sistema *recruitment-selection* deve disponibilizar para os seus usuários, porém, para utilizar algumas dessas 21 funcionalidades, o usuário precisará estar autenticado no sistema e também terá que possuir o perfil adequado para executar determinada funcionalidade.

### 3.2.2. Requisitos funcionais

Os requisitos funcionais consistem em funcionalidades e características a serem implementadas no sistema. As funcionalidades que a aplicação *recruitment-selection* oferta estão apresentadas abaixo e seus acessos estão divididos em perfis de acesso.

A partir da Tabela 1, pode-se vislumbrar as funcionalidades que o sistema disponibiliza para um usuário não autenticado (não possui perfil de acesso):

Tabela 1 - Requisitos funcionais usuário não autenticado

|  |  |
| --- | --- |
| ID Requisito | Descrição |
| RF1 | **Consultar processos seletivos em aberto**  Essa funcionalidade permite que o usuário busque, de forma resumida, informações de todos os processos seletivos que se encontram abertos a inscrições dentro da empresa. |
| RF2 | **Consultar processo seletivo específico**  Essa funcionalidade permite que o usuário busque informações específicas de um processo seletivo que ele deseja consultar. |
| RF3 | **Efetuar *login***  Essa funcionalidade permite que o usuário possa se autenticar no sistema, usando as suas credenciais válidas. |
| RF4 | **Criar conta como candidato**  Essa funcionalidade permite que o usuário possa se cadastrar no sistema apenas como um candidato. |
| RF5 | **Esquecer senha**  **Essa funcionalidade permite que o usuário possa redefinir a sua senha de acesso.** |

Além das funcionalidades RF1 e RF2, através da Tabela 2 pode-se visualizar que o *recruitment-selection* possui as seguintes funcionalidades para um usuário com perfil de recrutador:

Tabela 2 - Requisitos funcionais para um recrutador

|  |  |
| --- | --- |
| ID Requisito | Descrição |
| RF6 | **Cadastrar etapas de execução**  Essa funcionalidade **permite** ao recrutador que ele possa criar etapas que poderão ser utilizadas posteriormente na montagem de processo seletivo. |
| RF7 | **Cadastrar questões teóricas**  Essa funcionalidade **permite** que o recrutador possa cadastrar questões discursivas ou de múltipla escolha, no intuito de poder utilizar essas questões na construção de uma etapa. |
| RF8 | **Cadastrar processo seletivo**  Essa funcionalidade **permite** que o recrutador possa montar um processo seletivo composto por etapas que estejam cadastradas no sistema. |
| RF9 | **Encerrar processo seletivo**  Essa funcionalidade possibilita que o recrutador, sendo ele o responsável pelo processo seletivo, possa encerrar um processo seletivo. |
| RF10 | **Registrar avaliação sobre uma etapa realizada por um candidato**  Essa funcionalidade possibilita que o recrutador, sendo ele o responsável pelo processo seletivo, possa registrar avaliações para etapas realizadas pelos candidatos de um processo seletivo. |
| RF11 | **Liberar execução de etapa de um processo seletivo para um candidato**  Essa funcionalidade permite que o recrutador, sendo ele o responsável pelo processo seletivo, possa transicionar o estado de uma etapa, informando que ela já está disponível para um determinado candidato poder executá-la. |
| RF12 | **Consultar resposta de uma etapa realizada por um candidato**  Essa funcionalidade permite que o recrutador, sendo ele o responsável pelo processo seletivo, possa consultar as respostas que um candidato enviou ao realizar determinada etapa. |
| RF13 | **Consultar classificação geral de aproveitamento dos candidatos de um processo seletivo**  Essa funcionalidade permite que o recrutador, sendo ele o responsável pelo processo seletivo, consulte uma classificação dos candidatos baseadas em seus aproveitamentos de determinada etapa. |
| RF14 | **Redefinir senha**  **Essa funcionalidade permite que o recrutador possa trocar sua senha de acesso para outra.** |

Além das funcionalidades RF1, RF2 e RF14, na Tabela 3 está descrita as seguintes funcionalidades para um usuário com perfil de candidato na aplicação:

Tabela 3 - Requisitos funcionais para um candidato

|  |  |
| --- | --- |
| ID requisito | Descrição |
| RF15 | **Candidatar-se a um processo seletivo**  Essa funcionalidade permite que o candidato possa se candidatar a um processo seletivo em aberto da empresa. |
| RF16 | **Consultar suas candidaturas**  Essa funcionalidade permite que o candidato busque, de forma resumida, informações sobre todas as candidaturas que ele realizou. |
| RF17 | **Consultar uma candidatura específica**  Essa funcionalidade possibilita que o candidato possa consultar os detalhes de uma candidatura, que ele realizou em específico. |
| RF18 | **Encerrar sua candidatura em um processo seletivo**  Essa funcionalidade permite que o candidato possa encerrar sua participação em um processo seletivo. |
| RF19 | **Efetuar uma etapa do processo seletivo**  Essa funcionalidade permite que o candidato possa executar uma etapa disponível de um processo seletivo, sendo ele um candidato. |

Além das funcionalidades RF1, RF2, RF6, RF7, RF8, RF9, RF10, RF11, RF12, RF13 e RF14, a Tabela 4 representa no sistema as seguintes funcionalidades para um usuário com perfil de administrador:

Tabela 4 - Requisitos funcionais para um administrador

|  |  |
| --- | --- |
| ID requisito | Descrição |
| RF20 | **Registrar novo funcionário (Administrador/Recrutador)**  Essa funcionalidade permite ao administrador que ele possa cadastrar usuários que não sejam candidatos, no sistema. |
| RF21 | **Cadastrar perfil de acesso**  Essa funcionalidade permite que o administrador possa criar novos perfis de acesso, combinando quais funcionalidades aquele perfil terá direito de executar. |

### 3.2.3. Requisitos não funcionais

Além das funcionalidades apresentadas nas Tabelas 1, 2, 3 e 4, o *recruitment-selection* possui alguns requisitos não funcionais que são transparentes para o usuário, porém essas funcionalidades possuem um alto valor agregado para garantir uma boa performance da aplicação, uma boa segurança, uma boa sustentação do produto ofertado, além de poder exportar informações de comportamento do usuário no sistema para a equipe de negócio explorar tais informações no intuito de continuar entregando valor ao usuário.

Na Tabela 5 estão listados os requisitos não funcionais:

Tabela 5 - Requisitos não funcionais

|  |  |
| --- | --- |
| ID requisito | Descrição |
| RNF1 | Ter como forma de autenticação um modelo baseado em token cujo usuário informa o seu login e senha na plataforma e, após as validações das credenciais informadas, o sistema retornará um token de acesso no formato JWT cujo token deverá ser passado pelo usuário com o intuito de se identificar para o sistema como um usuário autenticado. |
| RNF2 | Exportar métricas para mensurar o desempenho da aplicação ao executar determinado trecho de código. |
| RNF3 | Exportar métricas para mensurar o comportamento do usuário ao acessar determinadas funcionalidades. |
| RNF4 | Realizar cache de conteúdos elegíveis para serem cacheados com o intuito de otimizar o tempo de resposta da aplicação. |

## 3.3. Modelagem do sistema

Para ilustrar a modelagem do sistema no aplicativo *backend*, foram construídos 6 diagramas UML: O primeiro foi o diagrama de caso de uso; o segundo foi o diagrama de classes; o terceiro, o quarto e o quinto foram diagramas de máquina de estados, sendo que estes 3 últimos diagramas (terceiro, quarto e quinto) tiveram como foco apresentar entidades diferentes, e o último foi o diagrama de entidade relacionamento.

Como ferramenta para a modelagem do sistema no *backend*, foi utilizado a versão gratuita do software *Draw.IO* através da versão *web* *diagrams.net*.

Para realizar o mapeamento e a construção das telas de acordo com as especificações do cliente, foi realizado o processo de *wireframe*.

### 3.3.1. Diagrama de caso de uso

Após a realização da análise de requisitos, iniciou-se a construção da modelagem de caso de uso do *recruitment-selection*. O diagrama de caso de uso tem como objetivo mapear o comportamento de um sistema, facilitar a visualização dos requisitos da aplicação, identificar as interações entre o sistema e os atores que irão acessar essas funcionalidades e buscar a delimitação do escopo da aplicação. Os componentes principais de um diagrama de caso de uso são os atores e os casos de uso. Como o intuito de uma modelagem de caso de uso é representar de forma simples e de fácil entendimento o que o sistema realiza do ponto de vista do usuário, os casos de uso acabam por ser representados da forma mais compreensível possível, para que os especialistas de negócio, que são os clientes, possam entender as funcionalidades que irão ser modeladas durante o processo de construção do *software*.

Figura 4 - Diagrama de Caso de Uso



Após a finalização da construção do diagrama de caso uso representado na Figura 4, ao todo foram identificados 21 casos de uso e 4 atores que irão interagir com essas funcionalidades de acordo com as delimitações dos escopos das funcionalidades.

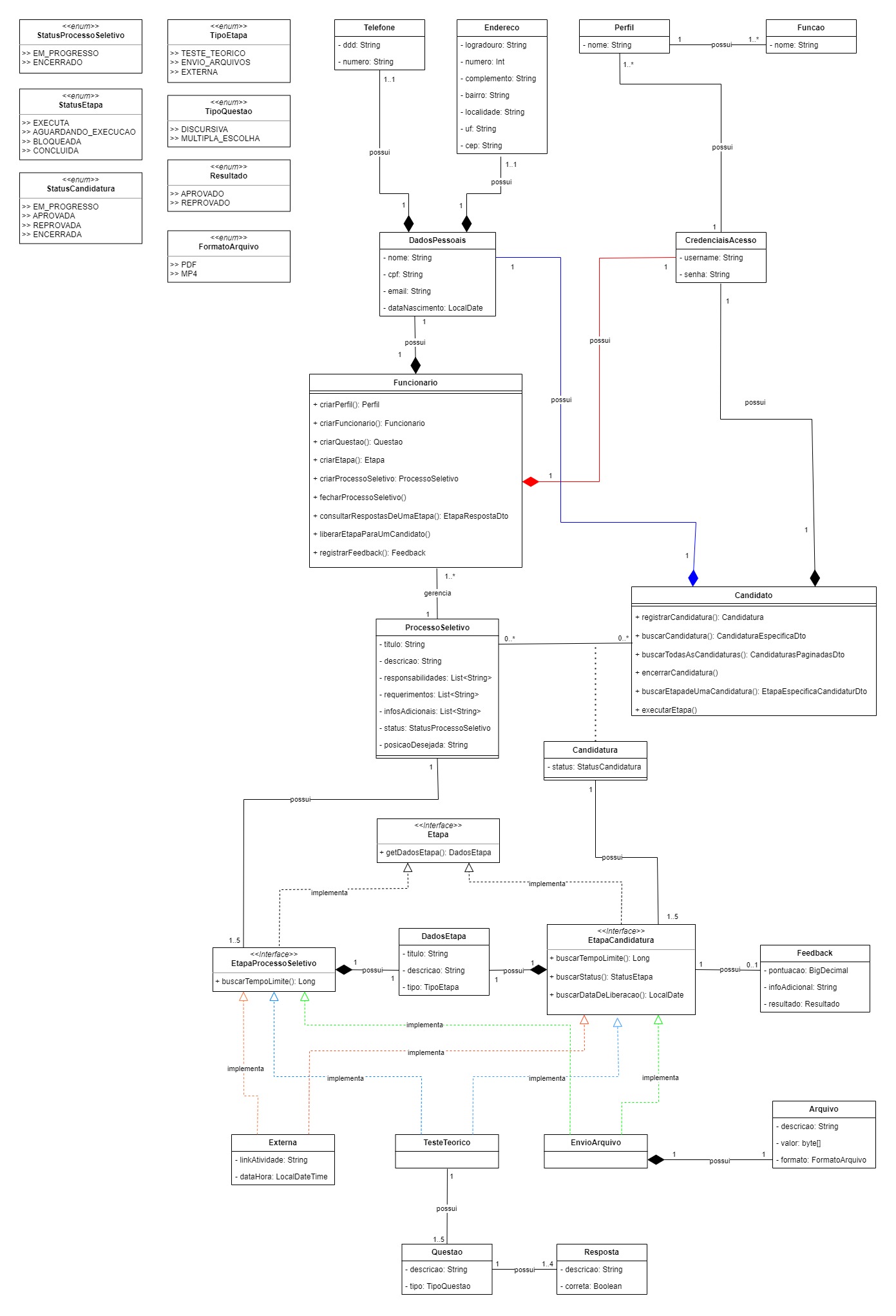
### 3.3.2. Diagrama de classes

Após a finalização da modelagem de caso de uso, teve início a construção da modelagem de classes do *recruitment-selection* representado na Figura 5, a qual teve como objetivo definir a estrutura da aplicação. O diagrama de classes é um diagrama do tipo estrutural que é importante para a construção da modelagem de objetos e da exibição dos relacionamentos entre esses objetos. Como o sistema tem por base o paradigma da orientação a objetos, a modelagem de classes permitiu representar as classes que foram utilizadas na aplicação e como elas iriam se relacionar entre si.

Desse modo, a primeira etapa buscou identificar os objetos tangíveis que iriam compor a estrutura do *recruitment-selection*. Após essa primeira análise, deu-se início a parte da identificação dos atributos que cada classe deveria conter. Para realizar essa identificação, buscou-se mapear as características que cada objeto deveria possuir.

Terminada a modelagem dos atributos das classes, iniciou-se o processo de identificar quais comportamentos uma classe deveria possuir e com quais outras classes ela deveria se relacionar a fim de criar uma determinada funcionalidade. Assim, após ter feito essa última análise, o próximo passo foi começar a identificar as semelhanças comuns (atributos, comportamentos ou relações) entre as classes mapeadas. A partir dessa verificação de semelhanças, foi permitido visualizar onde poderia ser aplicado conceitos como herança, composição e polimorfismo nas classes mapeadas.

Figura 5 - Diagrama de Classe



### 3.3.3. Diagrama de máquina de estados

Com a finalização da modelagem de classes, percebeu-se que existiam objetos que, ao longo do seu ciclo de vida no *recruitment-selection*, passariam por diferentes estados, sendo que cada estado poderia ter uma influência no fluxo de execução de uma determinada funcionalidade, caso um comportamento desse objeto fosse invocado.

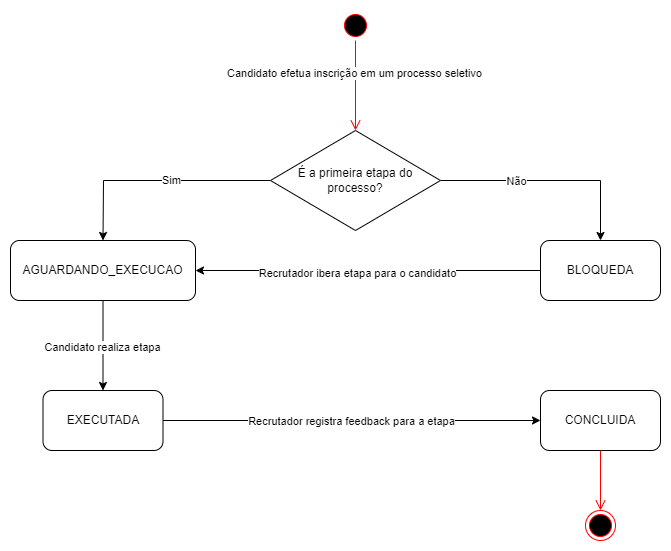
A partir dessa identificação, deu-se início a construção da modelagem de máquina de estados. Um diagrama de máquina de estados tem por objetivo representar os estados ou condições de um objeto ao longo do tempo de execução de uma funcionalidade na aplicação. Essa modelagem visou apresentar e descrever como seria feita a transição de estados pela qual determinados objetos iriam passar, ao longo do seu ciclo de vida dentro da aplicação e quais impactos um objeto, em um determinado estado, iria gerar no fluxo de execução de determinadas funcionalidades, quando um comportamento dele fosse acionado.

Ao todo foram construídos três diagramas de máquinas de estados que tiveram como intuito representar de forma visual como seria o ciclo de vida desses três objetos dentro da aplicação.

#### 3.3.3.1 Diagrama de máquina de estados de uma etapa

Uma etapa de um processo seletivo pode possuir vários estados e, dependendo do estado, é acionado um fluxo de execução diferente, caso o usuário acesse essa entidade. A modelagem retratada na Figura 6 visou representar como seria a transição de estados e os fluxos de execuções baseados nos estados das etapas que compõem um processo seletivo para o qual um candidato estava inscrito.

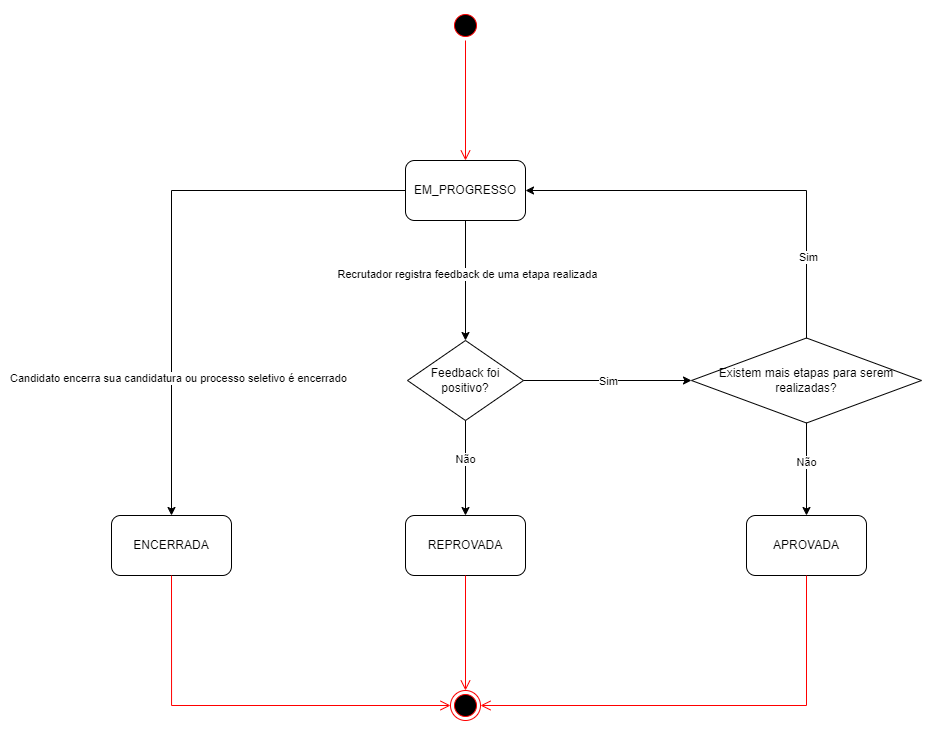
Figura 6 - Diagrama de Máquina de Estados Etapa



#### 3.3.3.2 Diagrama de máquina de estados de uma candidatura

Como uma candidatura será uma entidade no sistema e possuirá estados que adicionam características e novos fluxos de execução para essa candidatura, a modelagem na Figura 7 visou representar como seria o ciclo de estados e os fluxos de execuções baseados nessa candidatura.

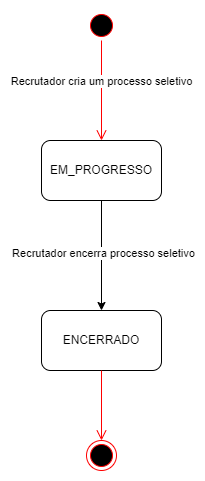
Figura 7 - Diagrama de Máquina de Estados Candidatura



#### 3.3.3.3 Diagrama de máquina de estados de um processo seletivo

Um processo seletivo baseado no estado em que ele se encontra, poderá receber novas candidaturas ou não. Para representar como seria o ciclo de estados e os fluxos de execuções baseados nesse processo seletivo, foi construída a modelagem representada pela Figura 8.

Figura 8 - Diagrama de Máquina de Estados Processo Seletivo



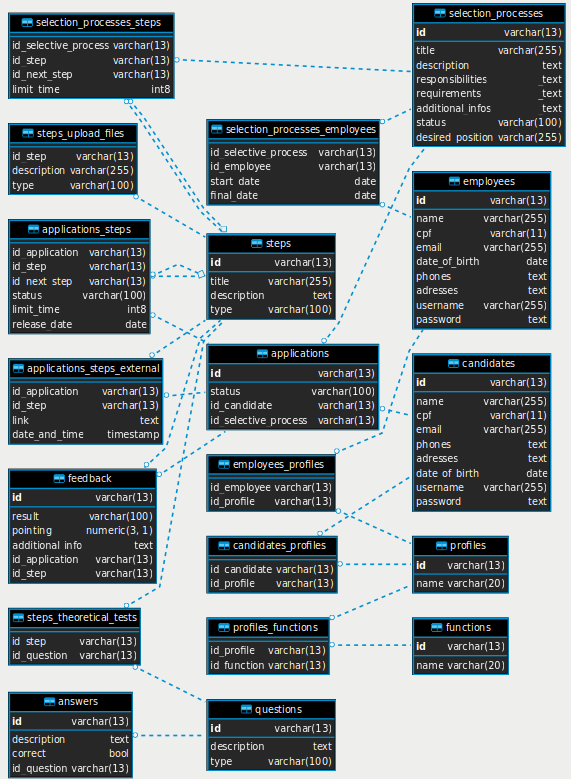
## 

### 3.3.4. Diagrama de entidade relacionamento

Como o *recruitment-selection* segue o paradigma orientado a objetos e o banco de dados PostgreSQL não possui um modelo de armazenamento compatível com o paradigma de programação da orientação a objetos, buscou-se realizar a criação do diagrama de entidade relacionamento representado na Figura 9. Esse diagrama tem o objetivo de ser uma representação visual de como as entidades irão se relacionar dentro do modelo relacional que o PostgreSQL utiliza. Além disso, nesse diagrama é possível vislumbrar os atributos, as chaves primárias e estrangeiras das tabelas e a cardinalidade entre as entidades que possuem algum tipo de relacionamento.

De forma grosseira pode-se traçar um paralelo de que um diagrama de entidade relacionamento tem o mesmo objetivo que um diagrama de classes, porém a diferença é que o diagrama de entidade relacionamento serve para representar como o modelo lógico dos seus dados estão estruturados em um a nível de organização de banco de dados, já o diagrama de classes serve para mostrar como o seu sistema orientado a objetos irá se organizar a nível de aplicativo.

Figura 9 - Diagrama de Entidade Relacionamento



### 3.3.5. *Wireframe*

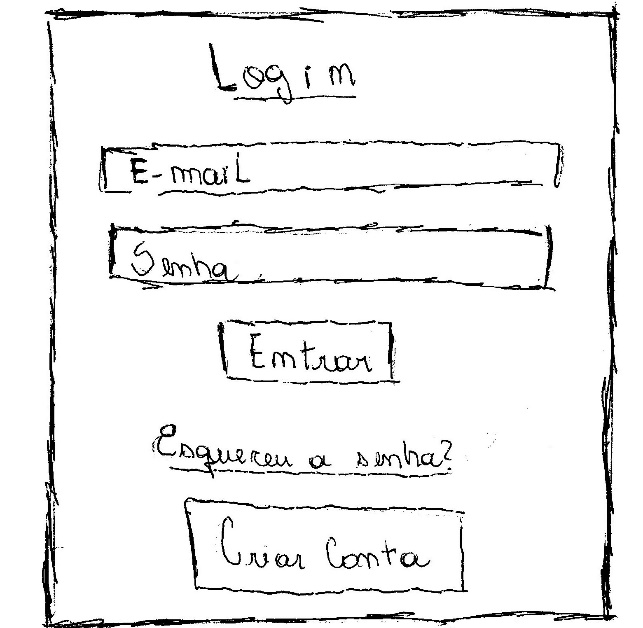
Após a finalização do processo da modelagem do aplicativo *backend*, iniciou-se a etapa da construção dos esboços (os *wireframes*) de algumas interfaces do aplicativo *frontend*. Os *wireframes* construídos tiveram como objetivo representar, para o cliente, os protótipos iniciais das interfaces da aplicação *frontend* a ser desenvolvida.

A ideia principal do *wireframe* é ilustrar, para os *stakeholders*, como ficou a espinha dorsal da interface de um site ou aplicativo, através da utilização de gráficos ou desenhos. Dessa forma, a abordagem do *wireframe* visa colher, de forma recorrente, avaliações do cliente a partir da apresentação dos esboços dos designs das telas de navegação. Sendo assim, tal abordagem tem como finalidade reduzir ou até anular a quantidade de alterações e adaptações exigidas pelo cliente, visto que já houve uma aprovação de *layout* anteriormente.

Para construir os esboços das interfaces, utilizou-se um modelo de baixa fidelidade. Os *wireframes* que seguiam esse modelo foram representados em desenhos no formato de rascunhos em folha de papel. Usar um modelo de baixa fidelidade no lugar de um de alta fidelidade, permitiu que os esboços, rapidamente, fossem modelados e modificados de acordo com as avaliações recebidas pelo cliente.

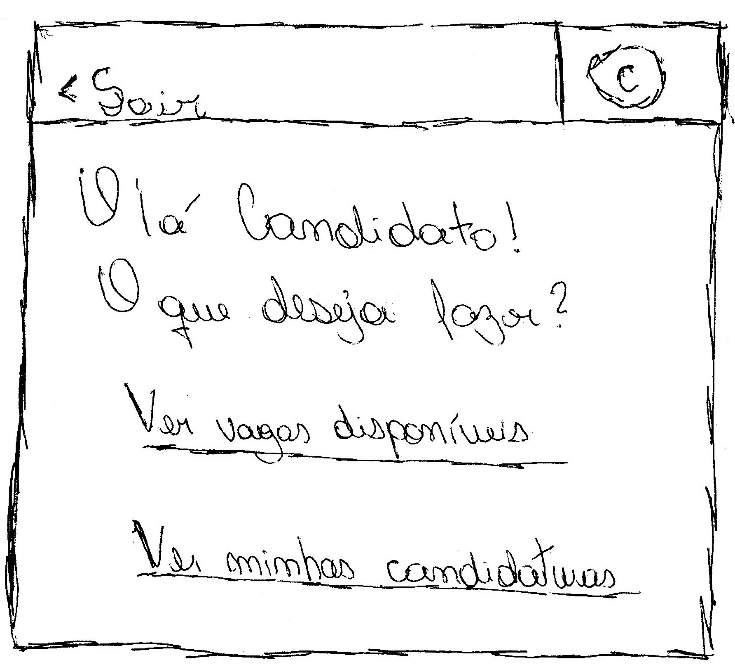
O *wireframe* da Figura 10 tem por objetivo representar o esboço da tela de *login* que o usuário irá acessar para efetuar o acesso ao sistema seja como um funcionário ou como um candidato. A ideia principal dessa tela é ter dois campos de preenchimento de dadosonde serão informados o e-mail e a senha de acesso ao sistema e ter também dois botões: um para efetuar a autenticação das credencias informadas nos campos e outro para criar uma conta no sistema. Além disso, será disponibilizado um *link* onde o usuário poderá clicar para solicitar uma redefinição de senha caso ele tenha esquecido a mesma.

Figura 10 - Wireframe tela de login



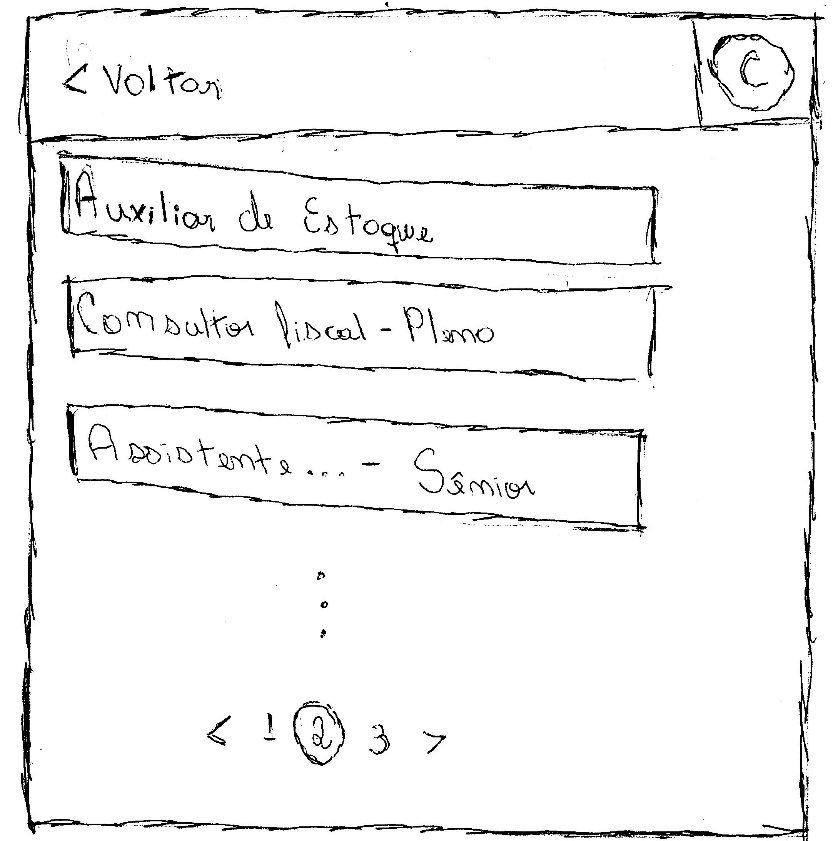
O protótipo da Figura 11 tem o intuito de representar a tela inicial para um candidato, após ter feito a sua autenticação no sistema. Nessa tela estão sendo representados dois *links* de redirecionamento para o candidato: um para poder visualizar os processos seletivos em aberto e outro para poder ver as candidaturas realizadas pelo mesmo. Além disso, nessa tela estará representada uma barra superior que irá conter um *link* cuja função será efetuar o *logout* do usuário e também irá possuir, no canto superior direito, um símbolo mostrando a letra inicial do usuário autenticado no sistema.

Figura 11 - Wireframe tela incial para um candidato



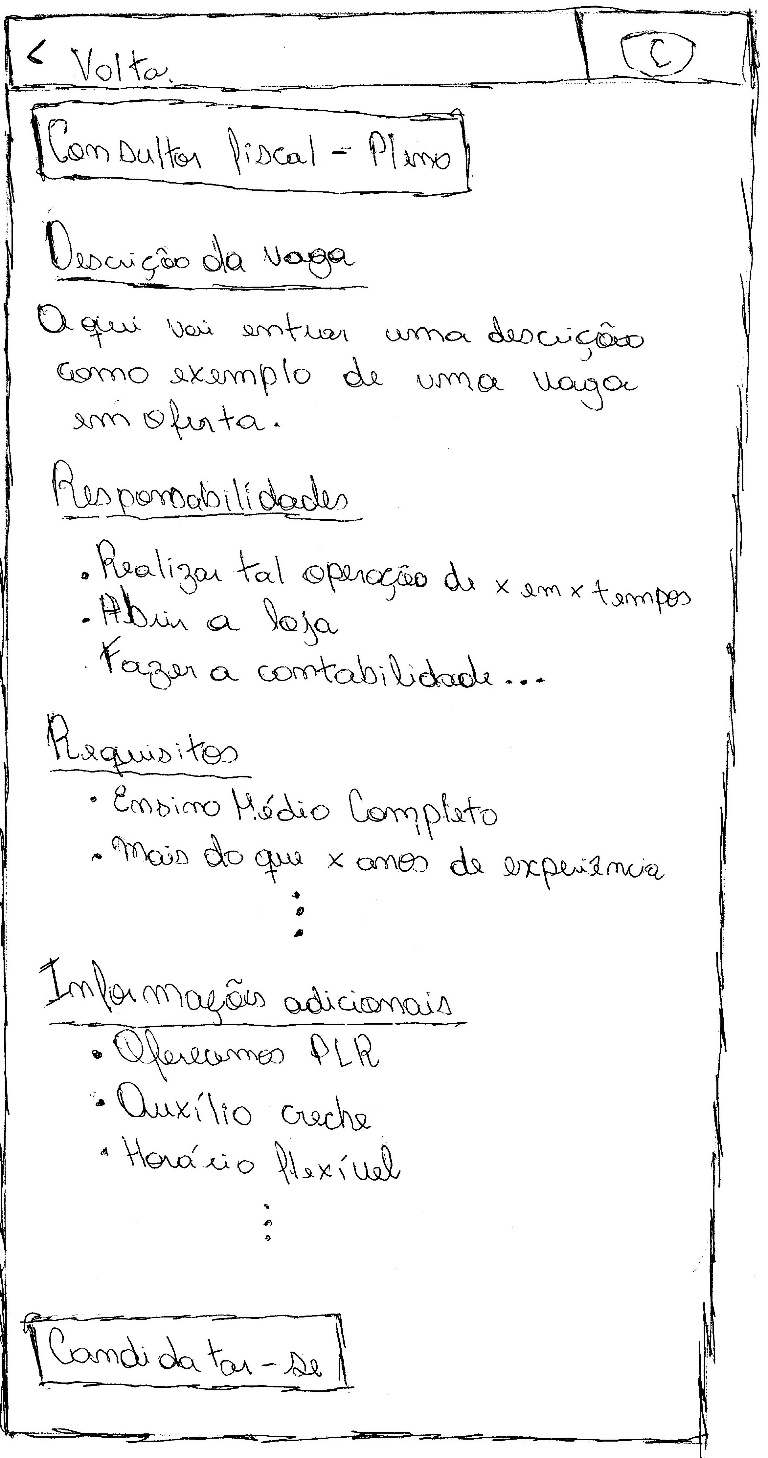
No próximo *wireframe* representado pela Figura 12 será mostrado o rascunho de como ficará a tela que irá apresentar os processos seletivos em aberto. Cada vaga disponível será apresentada ou como um botão ou como um *link* de redirecionamento. Esse botão ou *link* irá direcionar o usuário para uma outra tela onde ele poderá ver os detalhes daquele processo seletivo que ele clicou. Além disso, na parte debaixo da tela, mais próximo ao rodapé, será incluído uma visualização de processos seletivos em aberto, de forma paginada, por meio do menu de navegação representados por números e por setas de avanço e retrocesso. Também será incluído um menu superior contendo um *link* no canto esquerdo que, ao ser clicado, irá retornar o usuário para a tela a qual o mesmo estava anteriormente.

Figura 12 - Wireframe tela de processos seletivos em aberto



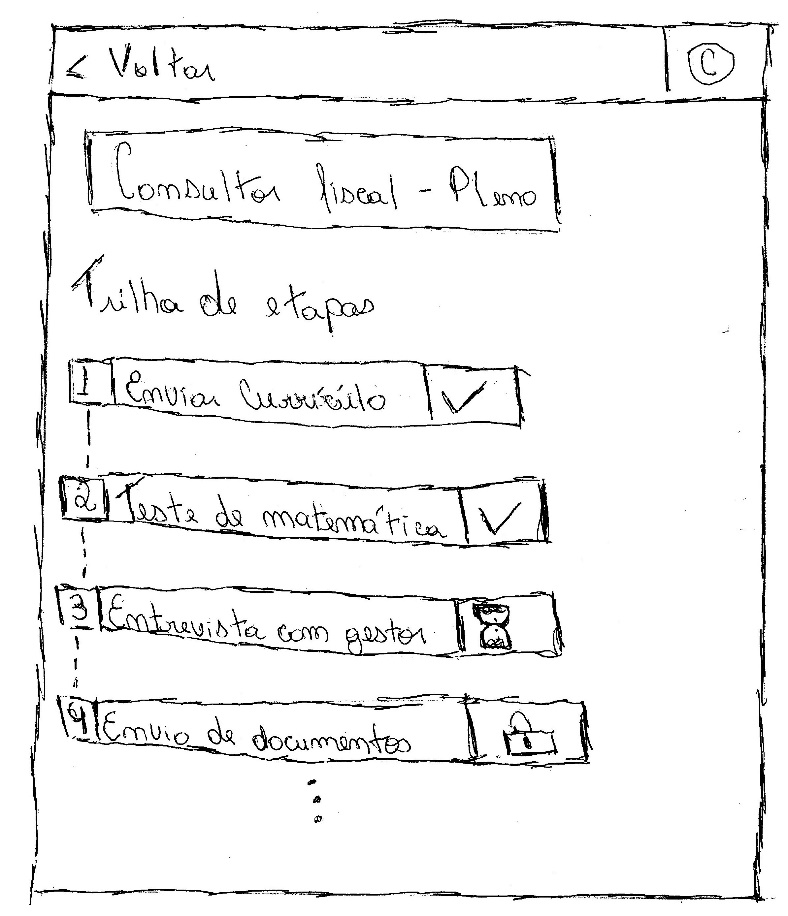
O esboço retratado na Figura 13 representa a tela de detalhes de um processo seletivo. A ideia é que o nome do processo seletivo fique em destaque dentro de uma caixa ou use algum tipo de marcação de destaque. Além disso, as seções de informações do processo seletivo também terão algum tipo de marcação de destaque para chamar a atenção do usuário. Os dados de responsabilidades, requisitos e informações adicionais de um processo seletivo serão representados em um formato de lista não ordenada. Também, ao final da página, será incluído um botão para o usuário poder clicar para efetuar a sua candidatura naquele processo seletivo.

Figura 13 - Wireframe tela de detalhes de um processo seletivo



O último protótipo apresentado na Figura 14 terá como função representar a tela da trilha de etapas para uma candidatura de um processo seletivo. Nessa tela, as etapas serão mostradas como uma lista ordenada na qual cada item irá apresentar o nome da etapa e o seu status atual. O status da etapa será apresentado através de um ícone que poderá ser interpretado por um sinal de certo (informando que uma etapa já foi realizada), por uma ampulheta (informando que uma etapa está aguardando a execução do candidato) ou por um cadeado (informando que uma etapa está aguardando liberação de execução para o candidato poder executá-la). Cada etapa terá um *link* para, quando o usuário efetuar o clique nesse *link*, ele ser redirecionado para uma tela onde irá mostrar os detalhes daquela etapa que foi clicada.

Figura 14 - Wireframe tela da trilha de etapas de um processo seletivo



# **4. IMPLEMENTAÇÃO DO SISTEMA**

Os tópicos a seguir têm como objetivo explicar como o sistema está estruturado internamente à nível de fluxos de execuções.

## 4.1. Aplicação *backend*

A API *recruitment-selection* foi construída internamente seguindo uma divisão lógica de três camadas: a camada *Controller*, a camada *Service* e a camada *Repository*. Dentro do sistema, cada camada é responsável por processar a solicitação do cliente dentro de um contexto específico.

A camada *Controller* é um componente de borda da aplicação que tem como objetivo expor as classes do sistema que irão receber, dar o tratamento primário das requisições do protocolo HTTP e responder às solicitações que os clientes da API irão realizar para acessar as funcionalidades do sistema. Algumas funcionalidades adicionais como validação de borda (validar se os dados recebidos na requisição estão em conformidade com os contratos de API que foram definidos entre o cliente e o servidor) e a autenticação e a autorização de usuários, encontram-se nas classes que compõem essa camada *Controller*. Em resumo, essa estrutura tem a função de tratar a comunicação com o cliente da API.

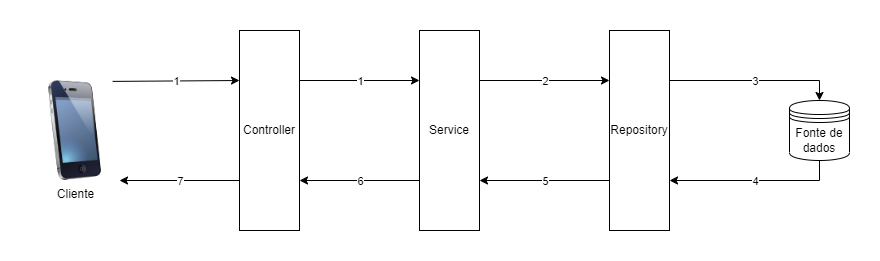
A camada *Service* é uma estrutura que possui o objetivo de conter as regras de negócio da aplicação. Como as regras de negócio estão contidas nos casos de uso que foram mapeados na modelagem UML, cada caso de uso será uma classe no sistema cujas classes fazem parte da estrutura *Service*. Como a lógica das funcionalidades se encontra nos casos de uso, essa camada se torna o coração do sistema. Sem essa camada, a aplicação não iria ter funcionalidades para ofertar.

Diferentemente da camada *Controller*, a qual tem o objetivo de tratar a comunicação com o cliente da API, a camada *Repository* que, também é um componente de borda da aplicação, tem a função de tratar a comunicação com as fontes de dados (SGBD’s, arquivos, fluxo de *bytes*, ...), armazenando informações que a camada *Service* pode precisar manipular durante a execução de um caso de uso. Nessa camada *Repository*, encontram-se as classes responsáveis por realizar a comunicação de acordo com a fonte de dados escolhida para armazenar as informações que a aplicação precisa manipular.

A motivação de utilizar essa divisão lógica dentro da API foi buscar aumentar o desacoplamento e o encapsulamento entre os objetos, além de poder utilizar o padrão da Inversão de Controle através do mecanismo da Injeção de Dependência. O uso desses mecanismos tem por objetivo melhorar a longevidade do *software* usando uma escrita de código mais concisa, mais coesa, mais reutilizável, mais flexível e de fácil testagem, mediante testes unitários automatizados.

De forma geral, o fluxo de execução de uma solicitação feita pelo cliente da API funciona da seguinte forma: o cliente realiza uma requisição HTTP e essa solicitação é tratada na camada *Controller*, onde é realizada o tratamento da requisição. Caso o tratamento da solicitação ocorra com sucesso, a camada *Controller* chama a camada *Service* e delega o processamento da solicitação para essa camada. A partir disso, um caso de uso irá tratar a requisição, aplicando as regras de negócio de uma determinada funcionalidade. Caso a *Service* precise manipular alguma informação que esteja armazenada fora da memória interna da aplicação, ela irá chamar a camada *Repository*, a última camada, que, por sua vez, irá na fonte de dados buscar a informação que foi solicitada pela *Service*.

Figura 15 - Arquitetura interna aplicação servidora

****

A partir da descrição de como funciona o fluxo de processamento de uma solicitação, pode-se vislumbrar que existe uma certa dependência entre as camadas. A camada *Controller* depende da camada *Service*, que por sua vez, depende da camada *Repository*. Para diminuir o acoplamento entre as classes de diferentes camadas, buscou-se realizar a Inversão de Controle, utilizando o conceito de Interfaces do paradigma da Orientação a Objetos. Assim sendo, com o uso das Interfaces, a classe que depende da abstração, não fica dependente de uma implementação concreta, e sim de uma fachada que pode ter a sua implementação concreta facilmente substituída sem a classe cliente precisar ficar ciente dessa modificação.

O código-fonte da API construída encontra-se disponibilizado de forma pública e pode ser consultado em um repositório[[6]](#footnote-7) na plataforma de hospedagem GitHub.

### 4.1.1. Documentação API

A documentação dos *endpoints* da API foi escrita, utilizando a especificação OpenAPI. Segundo a documentação oficial da OpenAPI, essa especificação é um modelo de linguagem para APIs HTTP, o qual fornece um meio padronizado para definir sua API a outras pessoas, por meio de uma interface padrão independente da linguagem de programação utilizada para construir o sistema.

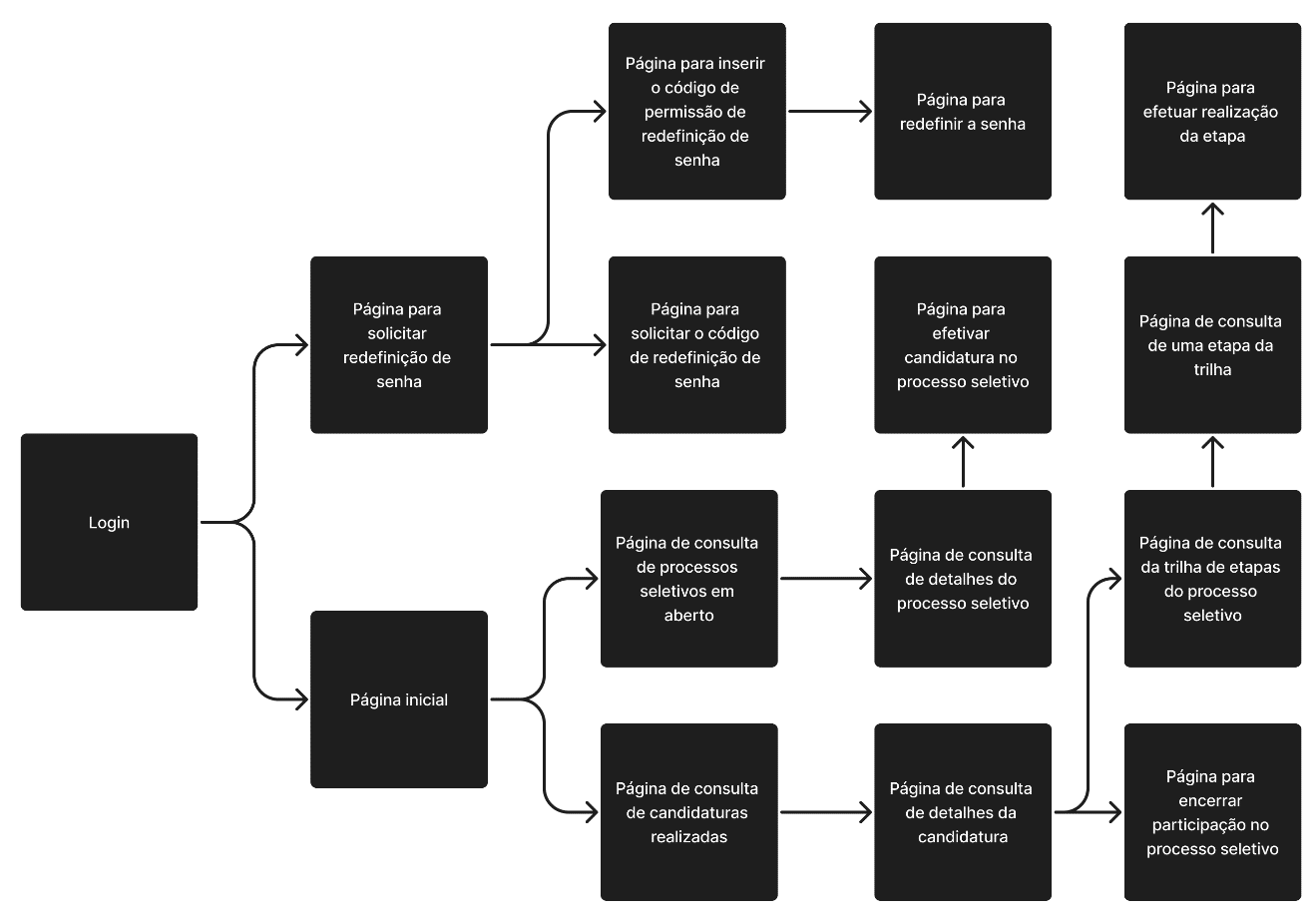
Para consultar a documentação dos *endpoints* da API, pode-se acessar o arquivo[[7]](#footnote-8) YAML, o qual foi construído seguindo a especificação do OpenAPI. Para uma visualização mais agradável da documentação, aconselha-se copiar o conteúdo do arquivo YAML e colá-lo na ferramenta *Swagger Editor*[[8]](#footnote-9) na seção de cor preta à esquerda da tela de navegação. Após isso, será possível consultar a documentação dos *endpoints* que a API disponibiliza.

## 4.2. Aplicação *frontend*

Para representar como seriam os fluxos de navegação para os usuários quando eles estivessem utilizando o aplicativo *frontend*, foi construído um fluxograma de navegação. Como a interação com as funcionalidades do sistema baseiam-se nos perfis de acesso que um usuário autenticado possui, cada usuário, de acordo com o seu perfil de acesso, teria o seu próprio fluxograma dentro do aplicativo.

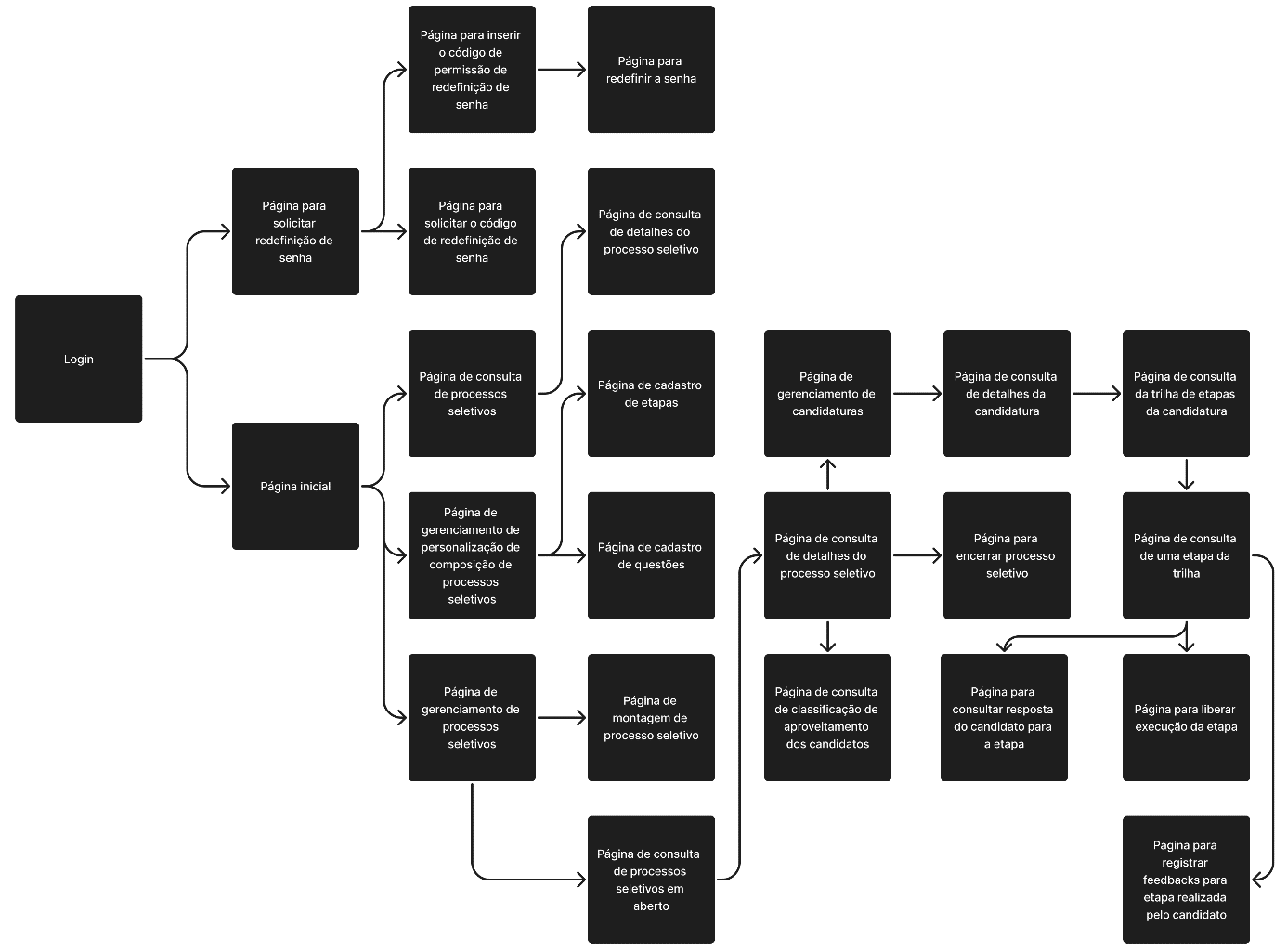
Para um usuário autenticado usando um perfil de candidato, o fluxograma representado através da Figura 16 iria consistir em efetuar operações de consultas nas entidades que representam os processos seletivos abertos e as candidaturas efetuadas por aquele candidato, efetuar operações de escrita como realizar ou encerrar uma candidatura em um processo seletivo e efetivar realização de uma etapa de um processo seletivo, e por último, iria poder realizar operações nas funcionalidades de autenticação do sistema como realizar *login* e redefinição de senha de acesso.

Figura 16 - Fluxograma candidato



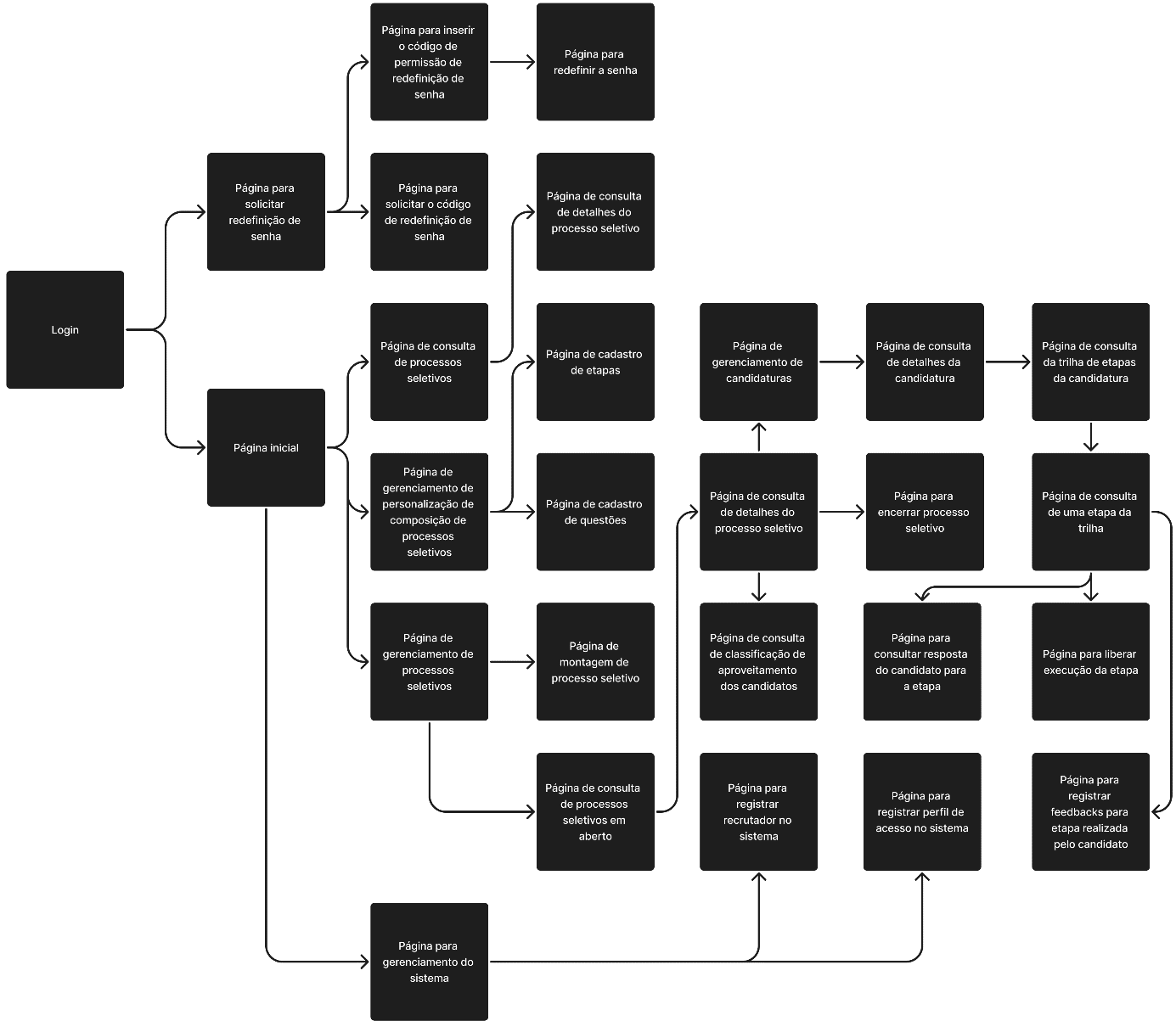
Para um usuário autenticado usando um perfil de recrutador, o fluxograma retratado na Figura 17 iria consistir em efetuar operações de consultas nas entidades que representam os processos seletivos e as candidaturas efetuadas pelos candidatos, efetuar operações de escrita como realizar a gestão de processos seletivos e de candidaturas, e por último, iria poder realizar operações nas funcionalidades de autenticação do sistema como realizar *login* e redefinição de senha de acesso.

Figura 17 - Fluxograma recrutador



Para um usuário autenticado usando um perfil de administrador, o fluxograma descrito na Figura 18 iria consistir em efetuar as mesmas operações permitidas para um recrutador. Além disso, através da Figura 18 pode-se vislumbrar que o administrador terá dois fluxos a mais relacionados a operações de escrita para criar registrar recrutadores e novos perfis de acesso no sistema.

Figura 18 - Fluxograma administrador



# **5. CONCLUSÃO**

Esse capítulo tem como objetivo apresentar os resultados obtidos com o desenvolvimento deste trabalho, além de explanar os trabalhos futuros que terão como intuito garantir a evolução do conhecimento em tecnologias e arquiteturas de desenvolvimento de aplicações.

O presente trabalho apresenta o processo da construção de um *software* de API complexo, seguindo algumas metodologias da engenharia de *software*. Os processos de análise de trabalhos correlatos, de levantamento e análise de requisitos e da modelagem do sistema através de diagramas UML foram tratados ao longo deste trabalho, a fim de, concomitantemente, conhecer e aprender um pouco mais sobre como as metodologias da engenharia de *software*, quando aplicadas, podem ajudar a construir um *software* mais organizado, eficiente e agradável ao gosto do cliente. Além disso, outro ponto bastante trabalhado e que possuiu uma evolução muito grande no conhecimento do autor deste trabalho, durante o ciclo de construção do *software*, foi o processo de modelagem das informações no banco de dados. Esse foi o ponto onde foi percebido a maior evolução de aprendizagem e conhecimento causada pelo desenvolvimento desse trabalho.

Os resultados obtidos com o desenvolvimento do sistema foram satisfatórios, e, por se tratar de uma aplicação com o objetivo de construir um *software* para um trabalho acadêmico, ainda pode ser aperfeiçoado no futuro. Novas funcionalidades, tecnologias, arquiteturas e outros processos da engenharia de *software* podem ser acrescentadas na aplicação e em seu processo de evolução para que se torne mais eficiente na questão de aderência ao gosto do cliente.

O propósito de se desenvolver um *software* para o setor de RH - uma área extremante importante que vem ganhando mais atenção por parte das empresas nos dias de hoje, além de passar por um processo de reformulação tecnológica - tem o objetivo de adquirir conhecimentos nos processos de engenharia de *software*, principalmente voltado mais ao lado dos processos de levantamento e análise de requisitos, junto ao cliente e ao processo de modelagem e construção de diagramas de aplicações de grande porte.

## 5.1. Trabalhos futuros

Para trabalhos futuros foram mapeados 2 pontos.

O primeiro ponto tem como objetivo explorar e desenvolver conhecimentos em tecnologias de construção de aplicativos móveis *android’s* para o lado *frontend* da engenharia de *software*. Dessa forma, o teste para validar os conhecimentos adquiridos ao longo do estudo das novas tecnologias será desenvolver um aplicativo cliente *frontend*, a qual terá como intuito consumir a API que foi desenvolvida ao longo desse trabalho. Essa aplicação *frontend* a ser desenvolvida irá representar a parte do sistema onde será implementado os designs e as telas de navegação para o usuário acessar e vislumbrar as funcionalidades ofertadas pela API de forma mais agradável para um ser humano.

O segundo ponto a ser explorado é buscar iniciar um estudo sobre a arquitetura de microsserviços. Recentemente, essa arquitetura vem ganhando bastante popularidade nos desenvolvimentos de API’s, e bastantes empresas já adotam essa arquitetura na construção de seus negócios. Em resumo: essa arquitetura permite que uma aplicação monolítica, que centraliza todo um contexto de negócio, seja desmembrada em pequenas aplicações na qual cada uma dessas micro aplicações será responsável por um contexto de negócio. Atualmente cada vez mais pessoas estão conectadas à internet, então um *software* precisa saber “crescer” e “diminuir” automaticamente, baseado em seu volume de acesso. Por isso, uma vantagem dessa arquitetura de microsserviços é permitir que esses *softwares* sejam escalados mais facilmente, com mais agilidade e com menores custos operacionais e financeiros para a organização responsável pela gestão dos servidores. Entretanto, essa arquitetura de microsserviços também traz alguns desafios, quando comparada a uma arquitetura monolítica. Devido a isso, para explorar os conceitos dos microsserviços deve-se entender as motivações da criação desses microsserviços, entender os cenários propícios para a implantação dessa arquitetura, os desafios impostos por essa estrutura e como isto vai impactar a engenharia de *software*. Para buscar validar os conhecimentos a serem adquiridos nesse estudo sobre a arquitetura descrita, será iniciado um desenvolvimento que terá como objetivo implantar, na API desenvolvida, ao longo desse trabalho, os conceitos da arquitetura de microsserviços.

# **REFERÊNCIAS**

ABINADER NETO, J. A.; LINS, R. D. *Web Services* em Java. 1. ed. [S.l.]: Brasport, 2006. ISBN 857452249X.

ALBERTIN, A. Comércio Eletrônico: benefícios e aspectos de sua aplicação. Revista de Administração de Empresas, v. 38, n.1, p.52-63, Janeiro/Março, 1998. \_\_\_\_ Comércio Eletrônico: Aspectos e benefícios. Relatório de pesquisa do Núcleo de Pesquisas e Publicações da Escola de Administração de Empresas de São Paulo - FGV. São Paulo, NPP, Número 23-1999.

BOEING, R. N.; BOEING, E. N. Estudo de tecnologias para melhorar a performance na busca de dados. 2018. Monografia (Sistemas de Informação) — Universidade do Sul de Santa Catarina. Disponível em: [https://repositorio.animaeducacao.com.br/bitstream/ANIMA/11097/ 1/TCC-FINAL.pdf](https://repositorio.animaeducacao.com.br/bitstream/ANIMA/11097/%201/TCC-FINAL.pdf).

CHIAVENATO, Idalberto. Gestão de pessoas: o novo papel dos recursos humanos nas organizações. 3ª. ed. Rio de Janeiro: Elsevier, 2010. 579 p

ELMASRI, R.; NAVATHE, S. B. Sistemas de Banco de Dados. 6. ed. [S.l.]: Pearson Universidades, 2010. ISBN 9788576390855.

FIELDING, R. T. Architectural Styles and the Design of Network-based *Software* Architectures. 2000. Tese (Information and Computer Science) — UNIVERSITY OF CALIFORNIA. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>.

FONSECA JÚNIOR, J. R. N. Estudo de caso usando o Spring Framework para a criação de aplicações *web* com J2EE. 2007. Monografia (Especialização em Sistemas de Computação para *Web*) — UNIVERSIDADE FEDERAL DE SANTA MARIA. Disponível em: [https://repositorio.ufsm.br/bitstream/handle/1/1725/Fonseca\_Junior\_Jose\_Ronaldo\_ Nogueira.pdf?sequence=1&isAllowed=y](https://repositorio.ufsm.br/bitstream/handle/1/1725/Fonseca_Junior_Jose_Ronaldo_%20Nogueira.pdf?sequence=1&isAllowed=y).

GALINDO JUNIOR, E. A.; ROCHA, R. D.; MACIEL, R. de S. DESENVOLVIMENTO DE API REST COM SPRING BOOT. Revista Científica do UniRios 2021.1. Disponível em: [https://www.unirios.edu.br/revistarios/media/revistas/2021/29/desenvolvimento\_de\_api\_rest\_ com\_spring\_boot.pdf](https://www.unirios.edu.br/revistarios/media/revistas/2021/29/desenvolvimento_de_api_rest_%20com_spring_boot.pdf).

GRADVOHL, A. L. S. Introdução à Linguagem de Programação Java. 2004.

HORSTMAN, C. S.; CORNELL, G. Core Java: Volume 1: Fundamentos. 8. ed. [S.l.]: Pearson Universidades, 2009. ISBN 8576053578.

JAVA. Documentação oficial. Disponível em: <https://docs.oracle.com/en/java/>

LAWLER, E., MOHRMAN, A., Administração de recursos humanos: construindo uma

parceria estratégica. In: GALBRAITH, J., LAWLER, E., Organizando para competir no

futuro. São Paulo, Makron Books, 1995.

MEIRELLES, F. de S. Estudo dos Gastos e Investimentos em Informática: Avaliação, Evolução e Tendências nas Médias e Grandes Empresas. 1999. Disponível em: <https://pesquisa-eaesp.fgv.br/sites/gvpesquisa.fgv.br/files/publicacoes/Rel04-1999.pdf>.

MENÉNDEZ, A. I. M. Uma ferramenta de apoio ao desenvolvimento de *Web Services*. 2002. Dissertação (Pós-Graduação em Informática) — UNIVERSIDADE FEDERAL DE CAMPINA GRANDE. Disponível em: [https://ri.ufs.br/bitstream/riufs/1089/1/ UmaFerramentaDeApoio.pdf](https://ri.ufs.br/bitstream/riufs/1089/1/%20UmaFerramentaDeApoio.pdf).

PONTES, B. R. Planejamento, Recrutamento e Seleção de Pessoal. 6. ed. [S.l.]: LTr Editora, 2010.

POSTGRESQL. Documentação oficial. Disponível em: <https://www.postgresql.org/about/>

REDIS. Documentação oficial. Disponível em: <https://redis.io/docs/>.

SOUZA, V. C. O. de; SANTOS, M. V. C. dos. Amadurecimento, Consolidação e Performance de SGBDs NoSQL - Estudo Comparativo. In: XI Brazilian Symposium on Information System. [s.n.], 2015. Disponível em: [https://sol.sbc.org.br/index.php/sbsi/article/view/5822/ 5720](https://sol.sbc.org.br/index.php/sbsi/article/view/5822/%205720).

SPRING. Documentação oficial. Disponível em: <https://spring.io/>.

OpenAPI. Documentação oficial. Disponível em: <https://www.openapis.org/>.

ULRICH, D. et al. RH de Dentro para Fora: Seis Competências para o Futuro da Área de Recursos Humanos. [S.l.]: Bookman, 2013

# **APÊNDICE A – QUESTIONÁRIO DE LEVANTEMENTO DE PERFIL DAS PEQUENAS EMPRESAS REFERNTE A GESTÃO DO PROCESSO DE R&S**

1. A sua empresa possui um sistema de automação eficiente para cuidar da gestão de R&S?
   1. Sim
   2. Não
2. Na empresa, quem é responsável pelo processo de R&S de novos talentos?
   1. Dono
   2. Diretor
   3. Gerente do setor
   4. Departamento de RH
   5. Não possui responsável específico
3. Em relação aos procedimentos que envolvem o processo de R&S, assinale as alternativas que sua empresa utiliza:
   * Análise de currículos
   * Entrevista com gestor
   * Avaliações psicológicas
   * Avaliações técnicas
   * Serviços terceirizados
   * Recrutamento interno
   * Aproveitamento de currículos
   * Dinâmicas
   * Nenhum
   * Outros
4. Em relação aos procedimentos que envolvem o processo de R&S, assinale as 3 alternativas que considera mais importantes:
   * Análise de currículos
   * Entrevista com gestor
   * Avaliações psicológicas
   * Avaliações técnicas
   * Serviços terceirizados
   * Recrutamento interno
   * Aproveitamento de currículos
   * Dinâmicas
   * Nenhum
   * Outros
5. Dos procedimentos considerados importantes e que não são feitos pela empresa, assinale as alternativas que justifique esse cenário.
   * Fragilidade econômica da empresa
   * Falta de estrutura interna
   * Falta de tempo
   * Falta de profissional capacitado
   * Todos os procedimentos importantes são executados pela empresa
   * Outros
6. Na sua opinião, quando bem estruturado e planejado para atingir as demandas da vaga a ser ofertada, a longo prazo uma boa gestão de R&S de novos talentos acaba tendo um alto custo para a empresa?
   1. Sim
   2. Não
7. A escolha de profissionais não qualificados e/ou com perfil diferente do desejado pode causar impactos financeiros negativos a empresa?
   1. Sim
   2. Não
8. Na sua opinião, soluções de Tecnologia da Informação e Comunicação podem auxiliar na melhor escolha de um novo colaborador?
   1. Sim
   2. Não

# **APÊNDICE B – QUESTIONÁRIO DE LEVANTEMENTO DE PERFIL DOS PROFISSIONAIS DE RH**

1. Assinale as técnicas que você utiliza na análise de perfil de novos talentos:
   * Análise de currículos
   * Entrevista com gestor
   * Avaliações psicológicas
   * Avaliações técnicas
   * Serviços terceirizados
   * Recrutamento interno
   * Aproveitamento de currículos
   * Dinâmicas
   * Nenhum
   * Outros
2. Dos procedimentos que envolvem o processo de R&S, assinale as 3 alternativas que considera mais importantes:
   * Análise de currículos
   * Entrevista com gestor
   * Avaliações psicológicas
   * Avaliações técnicas
   * Serviços terceirizados
   * Recrutamento interno
   * Aproveitamento de currículos
   * Dinâmicas
   * Nenhum
   * Outros
3. Assinale as ferramentas que você utiliza para auxiliar a análise de perfil de novos talentos?
   * Videoconferência
   * *Softwares* de Gestão de RH
   * Serviços de mensagens instantâneas
   * E-mail
   * Ligações telefônicas
   * Formulários físicos e/ou digitais
4. A escolha de profissionais não qualificados e/ou com perfil diferente do desejado pode causar impactos financeiros negativos a empresa?
   1. Sim
   2. Não
5. Você acredita que é possível fazer a análise de perfil utilizando somente ferramentas computacionais?
   1. Sim
   2. Não
6. Na sua opinião, quando bem estruturado e planejado para atingir as demandas da vaga a ser ofertada, a longo prazo uma boa gestão de R&S de novos talentos acaba tendo um alto custo para a empresa?
   1. Sim
   2. Não
7. A escolha de profissionais não qualificados e/ou com perfil diferente do desejado pode causar impactos financeiros negativos a empresa?
   1. Sim
   2. Não
8. Na sua opinião, soluções de Tecnologia da Informação e Comunicação podem auxiliar na melhor escolha de um novo colaborador?
   1. Sim
   2. Não

1. Disponível em: < <https://www.gupy.io/> > [↑](#footnote-ref-2)
2. Disponível em: < <https://somos.globo.com/> > [↑](#footnote-ref-3)
3. Disponível em: < <https://www.natura.com.br/> > [↑](#footnote-ref-4)
4. Disponível em: < <https://www.americanas.com.br/> > [↑](#footnote-ref-5)
5. Disponível em: < <https://www.catho.com.br/> > [↑](#footnote-ref-6)
6. Disponível em: < <https://github.com/LMoraes7/recruitment-selection/tree/master/> > [↑](#footnote-ref-7)
7. Disponível em: < <https://github.com/LMoraes7/recruitment-selection/blob/master/openapi.yml/> > [↑](#footnote-ref-8)
8. Disponível em: < <https://editor.swagger.io/> > [↑](#footnote-ref-9)