TiJob - Aplicação de Contratação de Profissionais da Tecnologia da Informação

Thiago Camargo da Silva[1], Leandro Ledel[2]

[1]Instituto Federal de São Paulo – Campus Hortolândia

Avenida Thereza Ana Cecon Breda, N.o 1896, Vila São Pedro, Hortolândia – SP

thiagocamargo2128@gmail.com, ledel@ifsp.edu.br

Abstract. *This article presents the development of a Web application intended to facilitate the registration or selection of IT professionals for interested companies. The objective is for candidates to apply for vacancies at companies, and for companies to select candidates. A selection of the best candidates for the vacancy will also be made by the system, showing if a candidate who applied for a vacancy has a good profile for it. Through some parameters informed during the publication of the vacancy, the system will compare the candidate's profile with these parameters (necessary technologies, knowledge in frameworks). The article will describe the application development detailing the technologies used and Agile methodology.*

Resumo. *Este artigo apresenta o desenvolvimento de uma aplicação Web que tem como intenção facilitar a inscrição ou seleção de profissionais de TI para empresas interessadas. O objetivo é que os candidatos se inscrevam nas vagas das empresas, e as empresas selecionem os candidatos. Uma seleção dos melhores candidatos para a vaga será feita também pelo sistema, mostrando se um candidato que se inscreveu em uma vaga possui um bom perfil para ela. Através de alguns parâmetros informados durante a publicação da vaga, o sistema comparará o perfil do candidato com esses parâmetros (tecnologias necessárias, conhecimento em frameworks). O artigo descreverá o desenvolvimento da aplicação detalhando as tecnologias utilizadas e metodologia Ágil.*

1. Introdução

Atualmente o mercado de TI está muito aquecido, vagas de emprego estão em alta, ao mesmo tempo em que profissionais qualificados são difíceis de serem localizados. Muitos recrutadores buscam profissionais até que estão contratados por outras empresas. Constantemente profissionais recebem propostas para vagas de outras empresas. O Brasil precisa formar 70 mil profissionais de TI por ano, entretanto só consegue formar 46 mil profissionais (BRASSCOM, 2019). É possível notar um deficit em relação aos novos profissionais no mercado de trabalho no setor de tecnologia, promovendo o aumento da procura por especialistas do setor.

Hoje existem diversos *sites* que tem como foco divulgação de vagas pelas empresas, em que os candidatos se apliquem as vagas, através de currículo *on-line*, ou PDF padrão. Isso faz com que os recrutadores tenham que analisar todos os currículos, e muitas vezes poucos se aplicam para aquela vaga. O mercado de tecnologia teve um aumento de 310% de vagas em 2020, por conta disso a busca pelos profissionais aumentou, bem como a dificuldade de selecionar o melhor perfil para a vaga certa (VALOR INVESTE, 2021).

Dentro desse contexto, surgiu a ideia de desenvolver uma aplicação *Web*, que ofereça a possibilidade dos candidatos na área de TI criarem seu currículo de forma *on-line,* bem como

permitir às empresas contratantes que tenham acesso a estes currículos, além da divulgação de vagas do setor.

2. Justificativa

A ideia do software a ser desenvolvido é que os candidatos realizem um cadastro de currículo *on-line* e vejam somente vagas da área de Tecnologia da Informação. As vagas e currículos possuem um detalhamento maior em relação às habilidades dos profissionais da área, como tecnologias, *frameworks*, projetos e outras. Possibilitando utilizar o sistema a seu favor, buscando o melhor candidato através de filtragem de perfis disponível no *site*.

Portanto, tendo um sistema de vagas exclusivas de TI, isso faz com que os detalhes de cada vaga e de cada profissional seja melhor explorados durante o cadastro. Além disso, a disponibilização de funcionalidades de filtragem de perfis permite que os recrutadores não precisem olhar todos os currículos dos candidatos por vaga, e sim apenas aqueles que atendam às regras utilizadas junto ao filtro realizado.

A tecnologia está sendo utilizada a favor dos contratantes e dos candidatos, e o sistema almeja prover uma chance maior de selecionar o profissional correto para a vaga em aberto.

3. Referencial Teórico

Nessa seção será abordado o referencial teórico do trabalho, o qual serviu como base de estudo para elaborar a criação da aplicação e do artigo.

Para o trabalho atingir seus objetivos e conseguir ser implementado, foram pesquisados e adquiridos conhecimentos acerca de tecnologias existentes, arquiteturas e softwares similares.

O referencial teórico permite também verificar o estado do problema a ser pesquisado, sob o aspecto teórico, bem como identificar outros estudos e pesquisas anteriormente realizados (LAKATOS; MARCONI, 2003).

Os *assuntos* que serão tratados nesta seção de Referencial Teórico são: 3.1) Arquitetura MVC; 3.2) Metodologia Ágil *Scrum*; 3.3) Visão geral de tecnologias a serem utilizadas na elaboração do sistema; 3.4) Engenharia *Web* e 3.5) Trabalhos Correlatos.

3.1 Arquitetura MVC

O sistema empregará a arquitetura Modelo-Visão-Controle (DEVMEDIA, 2013). Na Figura 1 está especificado como funciona essa arquitetura.

A Figura 1 ilustra uma situação em que o cliente faz uma requisição HTTP (no caso de uma aplicação *Web*) para o *controller*, o qual é responsável por receber as requisições e enviar uma resposta para o cliente.

A camada *Model* é onde fica localizada a regra de negócio da aplicação, e também é a camada que faz a comunicação com o banco de dados. Após a execução dessa camada, ela devolve uma resposta para o *Controller*, o qual então formata uma V*iew* - que se trata de uma página HTML, no caso de uma aplicação *Web* - que será enviada como resposta para o cliente.

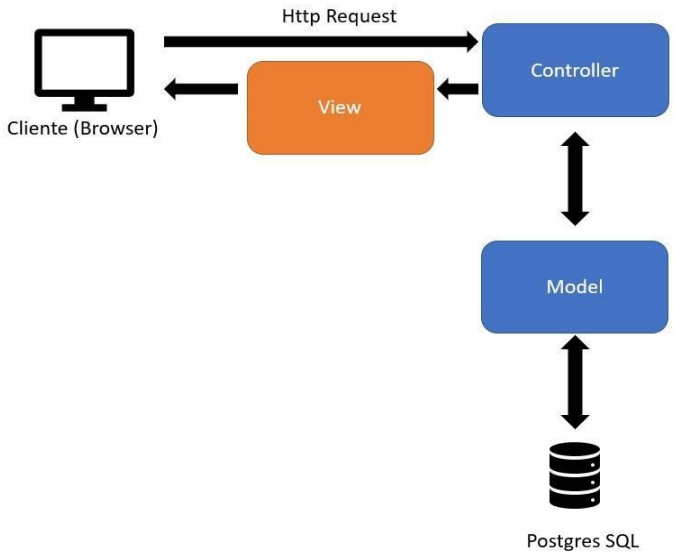

Figura 1: Arquitetura MVC (DevMedia, 2013).

3.2 Metodologia Ágil Scrum

Como forma de organizar o desenvolvimento do projeto, foi escolhida como metodologia o *Scrum*. Esse tópico demonstra os benefícios de se utilizar essa metodologia em um projeto de desenvolvimento de software.

Em um projeto seguindo a metodologia *Scrum*, são formadas equipes de 4 a 6 pessoas, dentro das quais os integrantes assumem diferentes papéis. O primeiro papel é do *Product Owner* (P.O.) que é o maior nível dentro da hierarquia. Em geral, o P.O. solicita a criação das histórias, pois tem contato com o cliente e conhece o produto. O responsável por cuidar das atividades do time, e reportar ao P.O. o andamento das atividades é o *Scrum Master*, que conhece muito bem da metodologia, e fica responsável também por conduzir as cerimônias junto ao time. Os demais membros do time são chamados de *Team Members*, e são os desenvolvedores(as) do projeto.

Essa metodologia é muito adequada para a criação de um software, principalmente devido ao fato de ser iterativa. Ao final de cada *sprint* é feita uma pequena entrega do software para o P.O., mostrando para o cliente que o desenvolvimento está tendo uma evolução. O sistema em desenvolvimento fica mais adaptável a eventuais mudanças, uma vez que, ao final de cada ciclo, podem surgir novas ideias de funcionalidades e melhorias para os próximos ciclos.

Por conta disso essa foi a metodologia escolhida para o trabalho. Ao final de cada *sprint* foi realizada uma análise do nível de dificuldade da implementação de cada funcionalidade (“fácil” ou “difícil”), verificando-se também formas de melhorar o desenvolvimento para as próximas *sprints*, bem como propondo meta(s) para organizar o(s) ciclo(s) seguinte(s).

3.3 Tecnologias

3.3.1 Spring MVC

Spring MVC é um *framework* que implementa o padrão modelo visão controle, onde a visão é uma página HTML chamada de *view*, e o *backend* é escrito na linguagem Java, usando o *framework Spring Boot* (DEVMEDIA, 2014).

3.3.2 Hibernate

Hibernate é um *framework* que facilita a comunicação entre o banco de dados e a API, possui métodos que criam automaticamente a *query* no banco de dados, fazendo com que o desenvolvedor não precise criar o código (DEVMEDIA, 2009).

3.3.3 Maven

Maven é uma ferramenta que gerencia as dependências do projeto, faz a construção e importação de algum código a partir de um repositório central. Na linguagem Java, é utilizado o arquivo pom.xml, onde são incluídas todas as dependências do projeto. (APACHE MAVEN, 2020).

3.3.4 Postgres SQL

Postgres é um Sistema Gerenciador de Banco de Dados SQL *Open Source*, ele foi escolhido pois é o SGBD disponibilizado na plataforma Heroku, onde será feito o *deploy* da aplicação (PostgreSQL WebSite).

3.3.5 GitHub

GitHub é uma plataforma de gerenciamento de versão de código fonte, é *open source*, e é uma das ferramentas mais utilizadas para TI no mundo, ela foi utilizada, pela praticidade de salvar o código fonte, e para *deploy* no ambiente (HOSTINGER, 2019)

3.3.6 Bootstrap

Bootstrap é um *framework* utilizado para formatação CSS das páginas HTML. O *framework* possui declarações simples que formatam os *Inputs*, *Models* e *Labels,* permitindo que não precisem ser criadas manualmente as classes CSS (BOOTSTRAP Documentation, 2021).

3.3.7 Ajax

Ajax, acrônimo de *Asynchronous JavaScript and XML*, é uma técnica de desenvolvimento *Web* utilizada para comunicação da *front-end* com *back-end*, dessa forma é possível criar páginas que enviem requisições sem precisarem ser recarregadas, diminuindo o tempo de espera de carregamento da página (DEVMEDIA, 2007). Facilita a usabilidade do usuário, quando diminuímos o tempo de espera quando não precisamos recarregar as páginas, e fazer requisições em paralelo.

3.3.8 Javascript

Javascript é a linguagem interpretada e orientada a objetos, que roda como *client side* no navegador do usuário (MDN Web Docs, 2022). É uma linguagem mundialmente conhecida e utilizada, por ser simples, e ser leve para processamento. No caso foi utilizada para validação dos campos quando o usuário digitar algo, o Javascript válida se algum campo não foi preenchido corretamente, e colocando máscara em alguns casos.

3.4 Engenharia WEB

Atualmente, a busca pela qualidade do software é essencial para o sucesso dele, alguns autores, a partir da área da Engenharia de Software, criaram a Engenharia WEB, focada na qualidade do desenvolvimento de aplicações *Web*. Segundo Medeiros (2021), os princípios da Engenharia *Web* são: Definir os objetivos e requerimentos de forma clara, desenvolver a aplicação *web* sistematicamente em fases, planejar cuidadosamente essas fases, gerenciar todo o processo de desenvolvimento de forma contínua. O desenvolvimento da aplicação será feito com base nas

metodologias impostas pelos especialistas, e buscando ter boas características de funcionalidade, usabilidade, confiabilidade, eficiência e portabilidade (ISO, 1991).

3.5 Trabalhos correlatos

Essa seção aborda sistemas semelhantes às do sistema que se pretende desenvolver, e que possuem foco ou aplicação em TI.

O *site* **APinfo** possui muita relação com o que será desenvolvido nesse artigo, pois o mesmo é focado exclusivamente para mercado de TI, e tanto empresas podem buscar candidatos por filtro de perfil, e candidatos podem se candidatar nas vagas escolhidas.

A Figura 2 apresenta uma das telas do **APinfo**.

Figura 2: Página inicial do **APinfo**.

O *site* **Vagas.com**, cuja página inicial está representada na Figura 3, possui funcionalidades semelhantes às do **APinfo**, porém abrange diversas áreas, não sendo focado somente em TI.

Figura 3: Página inicial do aplicativo **Vagas.com**.

O *site* **Geekhunter** (Figura 4) é especializado exclusivamente em vagas de desenvolvedores. Pode-se notar que o *site* possui uma tela de cadastro das habilidades, as quais estão relacionadas com a área de desenvolvimento de software.

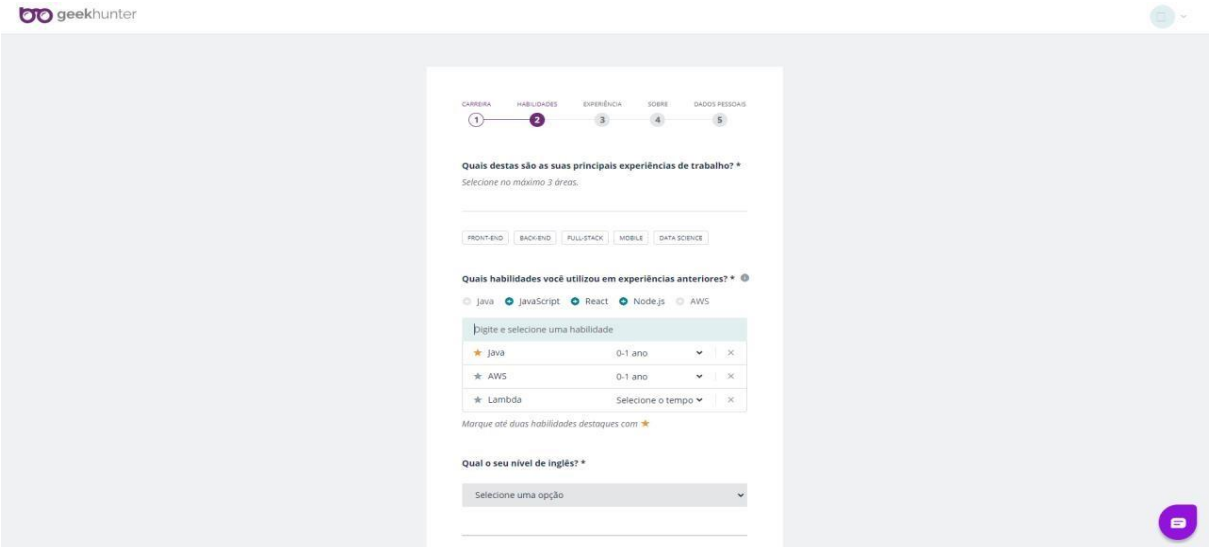

Figura 4: Página inicial de **Geekhunter**.

Após a pesquisa dos trabalhos correlatos, elaborou-se a Tabela 1, que resume algumas das principais características de cada sistema, facilitando assim a comparação entre os mesmos.

Tabela 1: Comparação de funcionalidades entre os sistemas correlatos.

Funcionalidade	APinfo	Vagas	Geekhunter
Filtro de vagas	Possui	Possui	Possui
Candidato se inscrever em vagas	Possui	Possui	Possui
Empresa publicar vagas	Possui	Possui	Possui
Filtro de candidatos	Possui	Possui	Possui
Currículo online	Possui	Possui	Possui
Área destinada às habilidades individuais	Não Possui	Não Possui	Possui
Desenvolvimento de software	Não Possui	Não Possui	Não Possui

Para a elaboração da Tabela 1, criou-se uma conta de teste em cada uma das plataformas, a fim de se analisar quais as informações necessárias para o cadastro, bem como as funcionalidades presentes de auxílio a(o) candidato(a).

Na área de "empresas" dos *sites* **Vagas.com** e **APinfo**, verificou-se as funcionalidades presentes do ponto de vista das empresas contratantes. Notou-se que ambos os *sites* possuem características semelhantes e, a partir dessa análise, foi observada a ausência das seguintes funcionalidades:

1) Campos para seleção de tecnologia(s) de preferência/domínio, nível de conhecimento dessa(s) tecnologia(s), *frameworks*, metodologias, quantidade de projetos entregues, especificação do(s) projeto(s) entregue(s), situações de desenvolvimento desafiadoras e testes de TI.

2) Uma área “técnica” a ser preenchida pelo candidato, que não seja apenas um texto livre;

3) Opções de outros cargos além do de “desenvolvedor”.

Tendo em vista essa possível “lacuna” de funcionalidades dos *sites* analisados, chegou-se à conclusão de que um *site* de vagas em T.I. deveria possuir, pelo menos, os campos descritos na comparação acima, a fim de prover uma melhor visão das habilidades do candidato para o possível contratante. Dentre os cargos disponíveis para a pretensão do candidato, poderiam ser inclusos os de “desenvolvedor”, “analista de qualidade”, “líder técnico”, dentre outros.

Outra característica percebida nos *sites* correlatos é que os mesmos envolvem contratos, cadastro das empresas que irão divulgar e pagamento para divulgação de vagas. Identificou-se a necessidade de um sistema mais simples e menos oneroso, e que cumpra os mesmos objetivos. Por essas razões, justifica-se o desenvolvimento do sistema proposto neste trabalho de conclusão de curso.

4. Metodologia

Nesta seção detalha-se a metodologia escolhida para o desenvolvimento do trabalho, que foi a Metodologia *Scrum*.

4.1 Metodologia Ágil Scrum

Para controle das funcionalidades entregues no sistema, o desenvolvimento foi feito com base na metodologia ágil *Scrum*, descrita na Seção 3.2 e que auxilia o time de desenvolvimento a resolver problemas mais complexos, e promover a satisfação na entrega do produto (KEN SCHWABER, 1990).

Primeiro será feita a priorização das partes do sistema, essa priorização será feita de acordo com o *backlog*. Quando possível, são feitas demonstrações das funcionalidades a serem entregues para o time de recursos humanos, a fim de que possam avaliar a satisfação do cliente com o sistema. Logicamente, considerando-se o desenvolvimento do TCC de forma individual, essa etapa será simplificada.

O desenvolvimento será dividido de 4 a 5 *sprints*, em que cada *sprint* deve ser feito um trecho do sistema. Entretanto, não haverá reuniões e cerimônias, novamente devido ao fato do trabalho ser realizado de forma individual.

5. Desenvolvimento

Nesta seção será descrita a etapa do desenvolvimento da Aplicação de Contratação dos Candidatos especializados em TI. O código fonte da aplicação será armazenado no GitHub (https://github.com/ThiCamargo18/vagas-tcc).

5.1 Levantamento de Requisitos

O ponto de partida para elaboração dos requisitos foi buscando software*s* similares, buscando adquirir uma diversidade de informações, para elaborar os requisitos necessários e melhorias. Dessa forma, utilizando a metodologia ágil, a partir de todos os requisitos que foram identificados, eles foram definidos como os épicos, que posteriormente serão quebrados em histórias e se necessário *sub-tasks.* Os requisitos identificados estão listados na Tabela 2.

Tabela 2: Requisitos Identificados para o Sistema.

Requisito	Tipo
Candidato realizar seu cadastro	Funcional
Candidato inscrever em vagas de TI	Funcional
Sistema possuir filtragem de vagas	Funcional
Empresa publicar vagas de TI	Funcional
Sistema possuir filtragem de candidatos	Funcional
Segurança, Usabilidade, Performance	Não Funcional

5.2 Casos de Uso

Prosseguindo com o desenvolvimento, a próxima etapa foi a identificação dos casos de uso para o sistema, os quais foram elencados em um diagrama, representado na Figura 5.

O Diagrama de Casos de Uso representa os atores do sistema – Candidato e Empresa, bem como as funcionalidades previstas para o mesmo, por meio dos seguintes casos de uso: Visualizar Vagas, Inscrever Vaga, Gerenciar Cadastro, Gerenciar Cadastro, Gerenciar Vagas, Selecionar Candidato e Filtrar Candidatos.

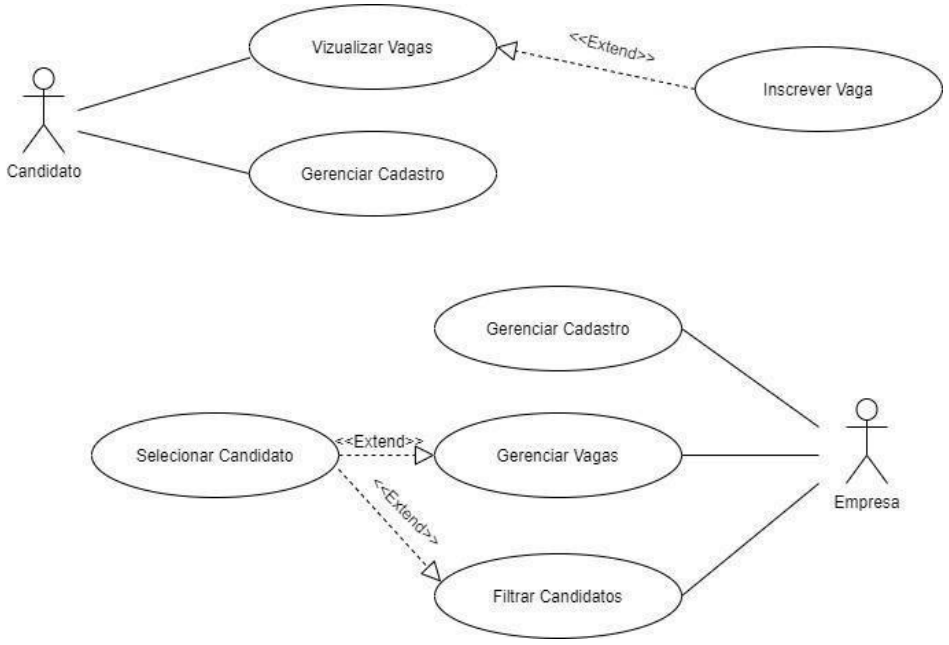

Figura 5: Diagrama de casos de uso dos sistema.

O caso de uso “Visualizar Vagas” pode ou não gerar a inscrição na vaga, presente no caso de uso “Inscrever Vaga”.

O caso de uso “Gerenciar Cadastro”, por parte do candidato, diz respeito a todo cadastro do perfil do usuário no sistema. Podendo criar, atualizar, listar o mesmo.

O caso de uso “Gerenciar Cadastro”, por parte da empresa, diz respeito a todo cadastro das informações da empresa no sistema. Podendo criar, atualizar, listar o mesmo.

O caso de uso “Gerenciar Vagas” diz respeito à criação de vagas, e listagem delas, e a partir desse caso de uso é possível selecionar um candidato, informado no caso de uso “Selecionar Candidato”.

O caso de uso “Filtrar Candidatos” diz respeito à funcionalidade do sistema em que a empresa - a partir de parâmetros informados - visualiza os candidatos resultantes da busca.

5.3 Diagrama Entidade-Relacionamento

A Figura 6 mostra o Diagrama Entidade-Relacionamento das tabelas criadas no banco de dados. O diagrama ficou relativamente complexo pela quantidade de tabelas, que em sua maioria dizem respeito ao cadastro do candidato, armazenando informações como: dadosPessoais, experiencia, dadosAdicionais, registroNacional, habilidades, projetos, tecnologia, *framework*, ferramenta e *role*. As demais tabelas dizem respeito às informações relativas à(s) empresa(s) contratante(s).

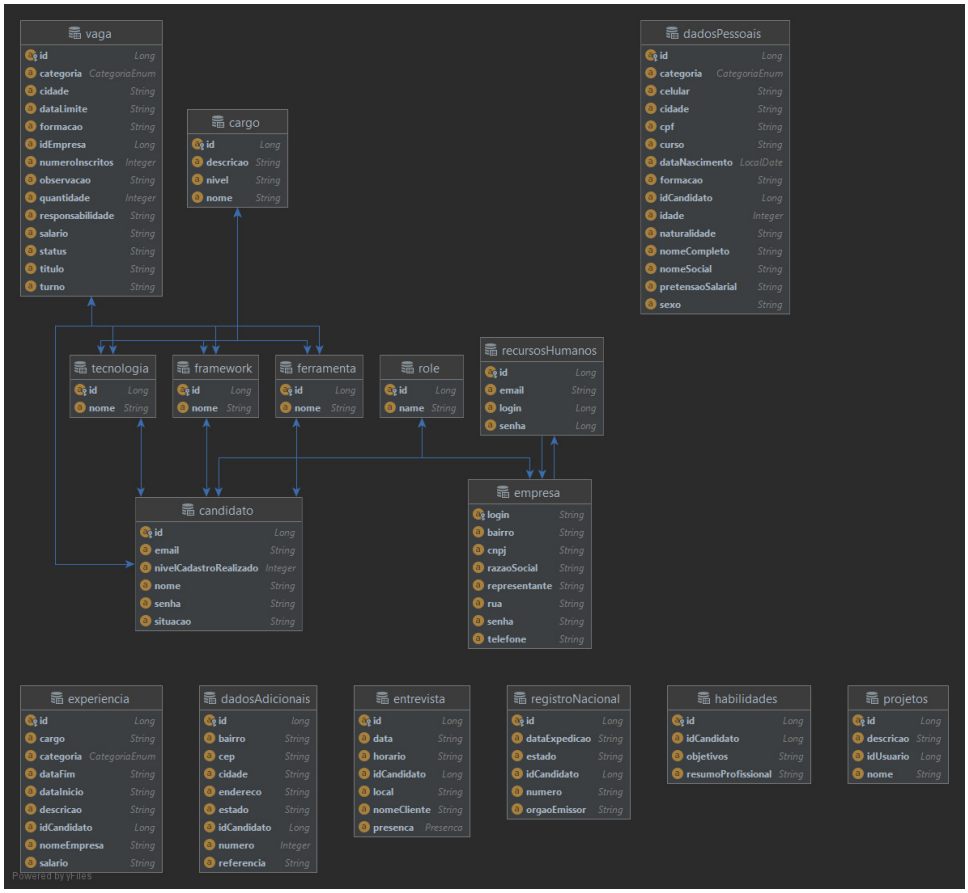

Figura 6: Diagrama Entidade-Relacionamento.

5.4 Planejamento do Desenvolvimento

Nessa etapa foi realizado o planejamento do desenvolvimento (Tabela 3), visto que foram criadas as histórias a partir dos requisitos identificados. Cada linha da tabela contém uma história e a identificação de qual *sprint* será realizada essa história, e qual o prazo para o desenvolvimento.

Ao fim de cada *sprint* foram realizados testes funcionais, buscando a validação de que aquela funcionalidade está funcionando como esperado.

Tabela 3: Planejamento de cada história.

História	Sprint	Prazo
Cadastro Candidato	1	23/08 até 03/09
Cadastro Empresa		
Tela Inicial Empresa	2	06/09 até 17/09
Tela Inicial Candidato		
Candidato Visualizar Vagas	3	20/09 até 01/10
Empresa Criar Vaga		
Candidato inscrever-se em uma vaga		
Empresa verificar inscritos nas vagas	4	04/10 até 15/10
Empresa filtrar inscritos		

5.4.1 Entrega da *Sprint* 1

Durante a realização do desenvolvimento da *Sprint* 1, foi realizada a etapa de análise, para documentar os atributos necessários, depois o desenvolvimento e os testes manuais para validar o desenvolvimento da funcionalidade almejada.

As Figuras 7 e 10 representam as interfaces gráficas desenvolvidas relativas a esta *Sprint*. Nesta *Sprint* foram criadas duas telas, bem como o código do *backend* responsável pelo tratamento das ações realizadas em ambas as telas. O objetivo destas telas é a realização do "Cadastro da Empresa" (Figura 7) e "Cadastro do Candidato" (Figura 10).

Figura 7: Tela de "**Cadastro da Empresa**".

O código de *backend* responsável pelo tratamento de cada uma das interfaces gráficas faz a comunicação com o Sistema Gerenciador de Banco de Dados, realizando o armazenamento das informações coletadas nas respectivas telas.

Os métodos das classes do *backend* são denominados "*endpoints*". Doravante será utilizada essa terminologia para referir-se aos métodos das classes de controle, uma vez que esta é a forma com que os mesmos são referenciados no mercado.

Para a tela de “Cadastro da Empresa”, foram criados três *endpoints* que estão na classe *controller* (Figura 8), a fim de receber os dados enviados pela *view* e salvar no banco de dados.

O primeiro *endpoint,* denominado “**/registrarEmpresa**”, é o *endpoint* que fica sem segurança para que qualquer empresa possa realizar o seu cadastro.

Tal *endpoint* apenas valida se o navegador do cliente já está autenticado a partir do método “securityService.isAuthenticated()” e, se não estiver autorizado, retorna a página HTML “/admin/login/registrar.html” para o navegador do cliente.

O *endpoint* “**/registrarEmpresa**”, com verbo POST, serve para receber os dados digitados pelo usuário e salvar no banco de dados através da classe *EmpresaAutenticacaoService*.

O último *endpoint* dessa classe, denominado “**/loginEmpresa**”, diz respeito à página de *Login* da empresa.

```
@GetMapping("/registrarEmpresa")
public String registration(Model model) {
    if (securityService.isAuthenticated()) {
        return "redirect:/admin";
    }

    model.addAttribute( attributeName: "userForm", new EmpresaEntrada());

    return "/admin/login/registrar";
}

@PostMapping("/registrarEmpresa")
public String registration(@ModelAttribute("userForm") EmpresaEntrada empresaEntrada, HttpServletRequest request) {
    empresaEntrada.setRoles(Collections.singletonList(new RoleEntity("EMPRESA")));

    EmpresaEntity empresaEntity = empresaAutenticacaoService.save(empresaEntrada);

    securityService.autoLogin(empresaEntrada.getLogin(), empresaEntrada.getSenha());

    EmpresaSessao empresaSessao = new EmpresaSessao(empresaEntity.getLogin(), empresaEntity.getRazaoSocial());

    HttpSession session = request.getSession();
    session.setAttribute( name: "empresaLogada", empresaSessao);

    return "redirect:/admin";
}

@GetMapping("/loginEmpresa")
public String login(Model model, String error, String logout) {
    if (securityService.isAuthenticated()) {
        return "redirect:/";
    }

    if (error != null)
        model.addAttribute( attributeName: "error",  attributeValue: "Usuario ou senha invalidos.");

    if (logout != null)
        model.addAttribute( attributeName: "message",  attributeValue: "Loggout realizado com sucesso.");

    return "/admin/login/login";
}
```

Figura 8: Classe *Controller* associada à tela de “**Cadastro da Empresa**”.

A Figura 9 apresenta o código-fonte da classe ***EmpresaAutenticacaoService***, a qual é responsável pela regra de negócio de fazer a conexão com o banco de dados.

Esta conexão, por sua vez, é realizada por meio do *JPA repository,* localizado na classe ***EmpresaRepository***.

```java
@Service
public class EmpresaAutenticacaoServiceImpl implements EmpresaAutenticacaoService {
    @Autowired
    private EmpresaRepository empresaRepository;
    @Qualifier("senha2")
    @Autowired
    private BCryptPasswordEncoder passwordEncoder;

    @Override
    public EmpresaEntity findByUsername(String id) {
        Optional<EmpresaEntity> empresaEntity = empresaRepository.findById(id);

        if (empresaEntity.isEmpty())
            return null;
        else
            return empresaEntity.get();
    }

    @Override
    public EmpresaEntity save(EmpresaEntrada empresaEntrada) {
        EmpresaEntity empresaEntity = EmpresaMapper.INSTANCE.mapToEntity(empresaEntrada);

        empresaEntity.setSenha(passwordEncoder.encode(empresaEntrada.getSenha()));

        return empresaRepository.save(empresaEntity);
    }
}
```

Figura 9: Cadastro da empresa - Classe ***EmpresaAutenticacaoService***.

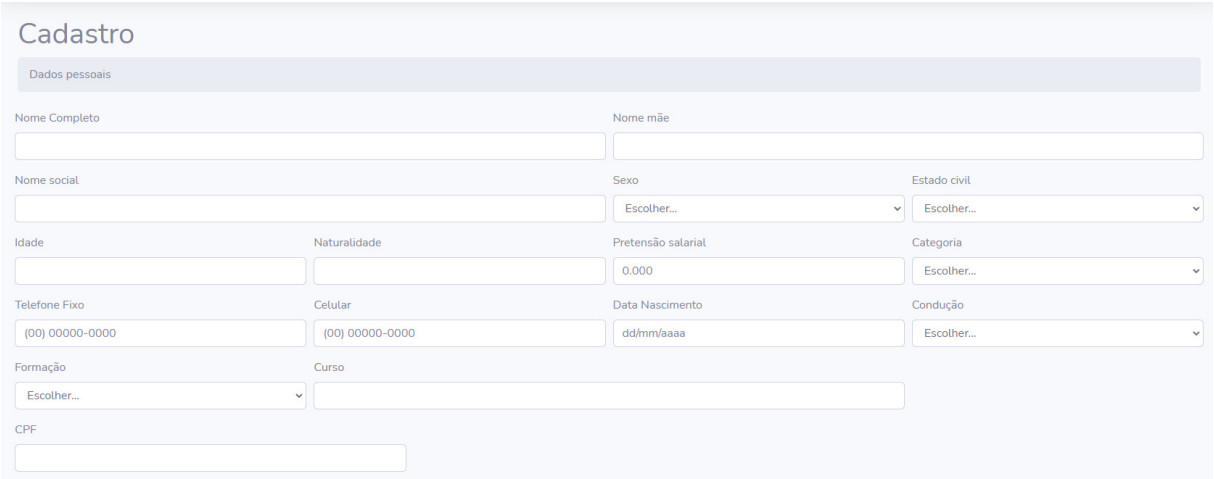

Figura 10: Tela de "**Cadastro do Candidato**".

A interface de usuário "Cadastro do Candidato" segue o mesmo padrão da tela "Cadastro da Empresa". A diferença ocorre nos *endpoints* e nos parâmetros de suas funções.

A classe ***LoginCandidatoController***, responsável pelo processamento dos dados do cadastro do candidato, está representada na Figura 11.

```java
@GetMapping("/registrar")
public String registration(Model model) {
    if (securityService.isAuthenticated()) {
        return "redirect:/";
    }

    model.addAttribute( attributeName: "userForm", new CandidatoEntrada());

    return "/candidato/login/registrar";
}

@PostMapping("/registrar")
public String registration(@ModelAttribute("userForm") CandidatoEntrada userForm, HttpServletRequest request) {
    userForm.setRoles(Collections.singletonList(new RoleEntity("CANDIDATO")));

    CandidatoEntity save = userService.save(userForm);

    securityService.autoLogin(userForm.getEmail(), userForm.getSenha());

    CandidatoSessao candidatoSessao = new CandidatoSessao();
    candidatoSessao.setId(save.getId());
    candidatoSessao.setNome(userForm.getNome());

    HttpSession session = request.getSession();
    session.setAttribute( name: "usuarioLogado", candidatoSessao);

    return "redirect:/";
}

@GetMapping("/login")
public String login(Model model, String error, String logout) {
    if (securityService.isAuthenticated()) return "redirect:/";

    if (error != null)
        model.addAttribute( attributeName: "error",  attributeValue: "Usuario ou senha invalidos.");

    if (logout != null)
        model.addAttribute( attributeName: "message",  attributeValue: "Loggout realizado com sucesso.");

    return "/candidato/login/login";
}
```

Figura 11: Código-fonte do *Controller* responsável pelo **Cadastro do Candidato.**

Como o desenvolvimento do sistema ocorreu com apenas uma pessoa, não foi necessária a realização de reuniões diárias de acompanhamento do projeto - denominadas "*daily"* na metodologia *Scrum*.

Em relação à retrospectiva do conhecimento adquirido durante a realização da *sprint*, pode-se mencionar que, para elaborar os atributos necessários do cadastro do usuário, muitos *sites* de vagas acabam seguindo o mesmo padrão de cadastro. Devido a isso, acabou sendo simples identificar os atributos necessários para os cadastros de usuário e empresa.

O que acabou se tornando um pouco mais complexo, foi a identificação dos atributos necessários para o cadastro do perfil de T.I. do candidato.

5.4.2 Entrega da *Sprint* 2

Durante o desenvolvimento da *Sprint* 2, após ter feito o cadastro do candidato e da empresa, foi necessário construir a página de início de cada um desses atores do sistema.

Após o login/cadastro ser realizado, o **Candidato** e a **Empresa** terão acesso às páginas HTML representadas nas Figuras 12 e 13, respectivamente.

Cada uma das telas oferece um menu de funcionalidades disponíveis em cada caso pelo sistema.

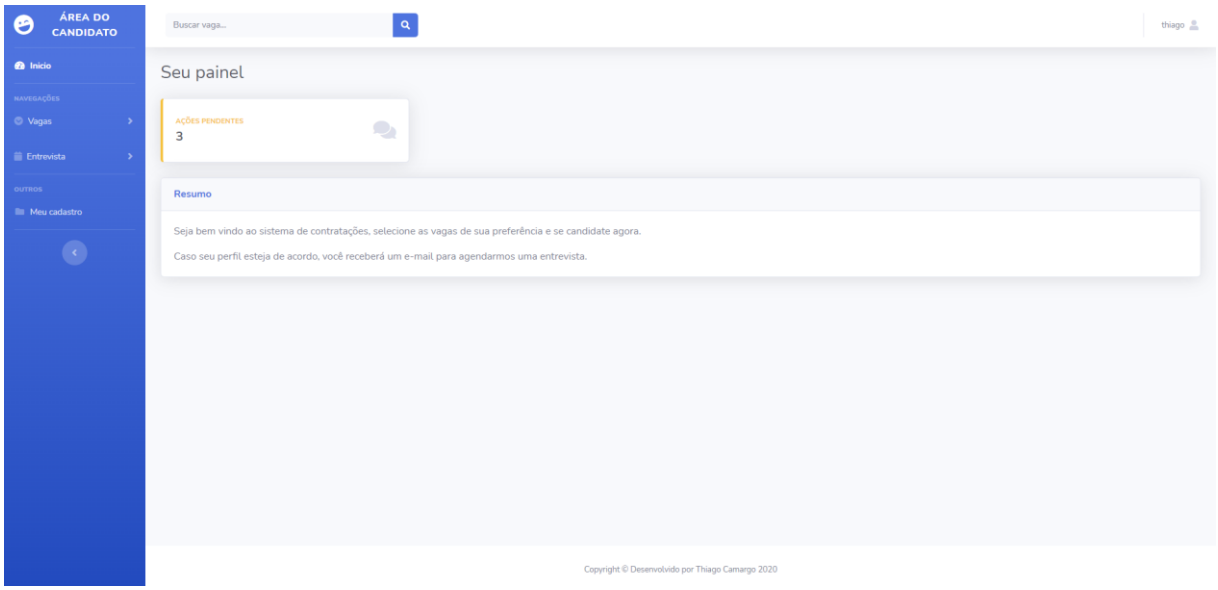

Figura 12: Tela inicial do **Candidato**.

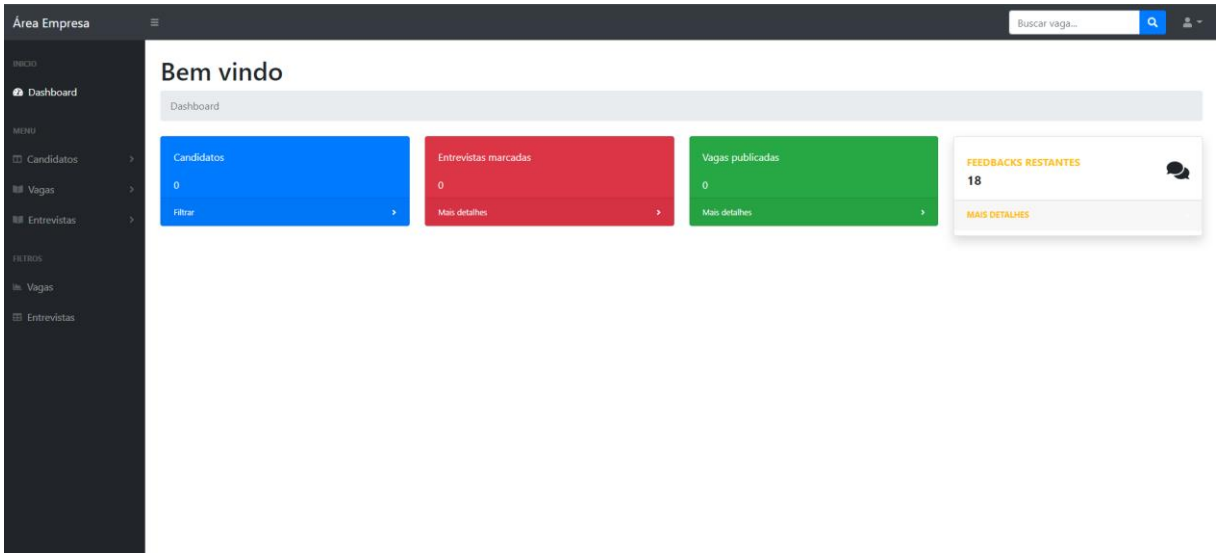

Figura 13: Tela inicial da **Empresa**.

Durante o desenvolvimento da *Sprint* 2, a maior dificuldade foi pensar em uma tela para o usuário final, buscando seguir o padrão dos *sites* mais famosos.

Para manter a qualidade de usabilidade, por exemplo, o botão de gerenciamento do perfil geralmente fica no canto superior direito - padrão identificado na maioria dos *sites*.

Todo *design* do *site* foi feito com base na tecnologia *Bootstrap*, que faz a formatação do HTML e do CSS.

A Figura 14 representa o código HTML correspondente à Tela de Entrada do Candidato - representada na Figura 12. Nesta listagem da Figura 14 é possível ver as classes utilizadas

pelo *Bootstrap* para formatar o CSS - como exemplo a classe "*container-fluid*" que faz com que o código fique centralizado na tela, colocando margem nos cantos da página e melhorando, portanto, a sua visibilidade.

```
    <!-- Topbar -->
    <div th:replace="candidato/fragments/fragment :: topbar"></div>
    <!-- End of Topbar -->

    <div class="container-fluid">

        <div class="d-sm-flex align-items-center justify-content-between mb-4">
            <h1 class="h3 mb-0 text-gray-800">Seu painel</h1>
        </div>

        <div class="row">
            <div class="col-xl-3 col-md-6 mb-4">
                <div class="card border-left-warning shadow h-100 py-2">
                    <div class="card-body">
                        <div class="row no-gutters align-items-center">
                            <div class="col mr-2">
                                <div class="text-xs font-weight-bold text-warning text-uppercase mb-1">Ações pendentes</div>
                                <div class="h5 mb-0 font-weight-bold text-gray-800">3</div>
                            </div>
                            <div class="col-auto">
                                <i class="fas fa-comments fa-2x text-gray-300"></i>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>

        <!-- Approach -->
        <div class="card shadow mb-4">
            <div class="card-header py-3">
                <h6 class="m-0 font-weight-bold text-primary">Resumo</h6>
            </div>
            <div class="card-body">
                <p>Seja bem vindo ao sistema de contratações, selecione as vagas de sua preferência e se candidate agora.</p>
                <p class="mb-0">Caso seu perfil esteja de acordo, você receberá um e-mail para agendarmos uma entrevista.</p>
            </div>
        </div>
    </div>
</div>
```

Figura 14: Implementação usando o ***Bootstrap***.

Em relação aos conhecimentos adquiridos na *Sprint*, e à entrega realizada, a maior dificuldade foi a criação de tela(s) de fácil utilização, na(s) qual(is) que o usuário encontre os botões de forma natural.

5.4.3 Entrega da *Sprint* 3

Durante o desenvolvimento da *Sprint* 3, foi realizada a criação das telas de "Cadastro de Vagas" (Figura 15), "Visualização das Vagas" (Figura 16), bem como de uma tela de "Inscrição para Vaga" (Figura 17).

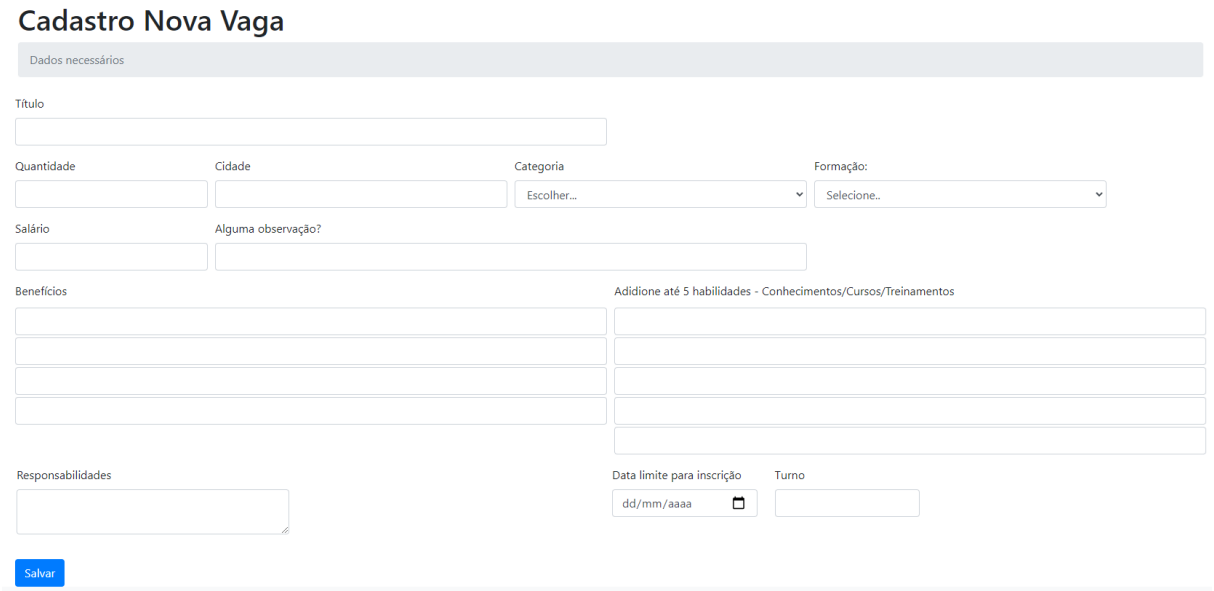

Figura 15: Tela de "**Cadastro de Vaga**".

Figura 16: Tela de **"Visualização de Vagas".**

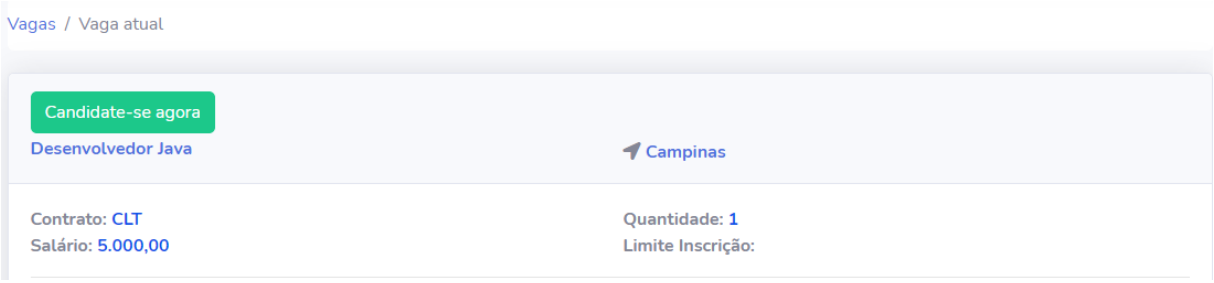

Figura 17: Tela de "**Inscrição para Vaga**".

O botão "**Candidate-se agora**", localizado na tela de "Inscrição para Vaga" (Figura 17), envia uma requisição para o *backend* passando como parâmetro o ***id*** (identificador) da vaga que o candidato está se candidatando.

Após isso a classe *controller* busca o ***id*** (identificador) do candidato – o qual foi armazenado na sessão após a realização do *login,* conforme ilustrado na Figura 18.

```java
@GetMapping("/inscrever/{idVaga}")
public String inscrever(@PathVariable Long idVaga, HttpServletRequest request) throws Exception {
    int nivelCadastro = candidatoService.cadastroBasicoRealizado(CandidatoSessao.getId(request));

    if (nivelCadastro < 3)
        return "redirect:cadastro/gerenciar";

    vagaCandidatoService.inscrever(CandidatoSessao.getId(request),idVaga);

    return "redirect:/";
}
```

Figura 18: Classe *Controller* de **Vagas**.

A classe **VagaService**, representada na Figura 19, é responsável pelas seguintes regras de negócio: a) buscar a respectiva vaga no banco de dados; b) validar a existência da vaga; c) buscar os candidatos daquela vaga; d) validar se o candidato já está inscrito; e) incluir o candidato na lista de inscritos daquela vaga e, por fim, f) fazer o *update* (atualização) no banco de dados.

```java
public void inscrever(Long idUsuario, Long idVaga) throws Exception {
    Optional<VagaEntity> vagaEntityOptional = vagaRepository.findById(idVaga);

    if (vagaEntityOptional.isEmpty()) {
        throw new Exception("Vaga não encontrada");
    }

    CandidatoEntity candidatoEntity = candidatoService.buscarPorId(idUsuario);
    VagaEntity vagaEntity = vagaEntityOptional.get();

    List<CandidatoEntity> clientesCadastrados = vagaEntity.getClientes();

    for (CandidatoEntity candidato : clientesCadastrados) {
        if (candidato.getId().equals(candidatoEntity.getId())) {
            throw new Exception("Você já se inscreveu nessa vaga!");
        }
    }

    clientesCadastrados.add(candidatoEntity);

    vagaEntity.setClientes(clientesCadastrados);
    vagaEntity.setNumeroInscritos(vagaEntity.getNumeroInscritos() + 1);

    vagaRepository.save(vagaEntity);

    VagaMapper.INSTANCE.mapToSaida(vagaEntity);
}
```

Figura 19: Código-fonte da Classe ***VagaService***.

O método *save* do JPA *repository* fica responsável por criar as *queries* do banco de dados, no caso o *update* (atualização) da entidade **Vagas**.

5.4.4 Entrega da *Sprint* 4

Durante a *Sprint* 4 foi realizada a entrega das telas nas quais se pode consultar os inscritos de uma determinada vaga (Figura 20).

Cada vaga possui um *link* que redireciona para a página em que é possível ver os inscritos daquela vaga específica, conforme representado na Figura 21.

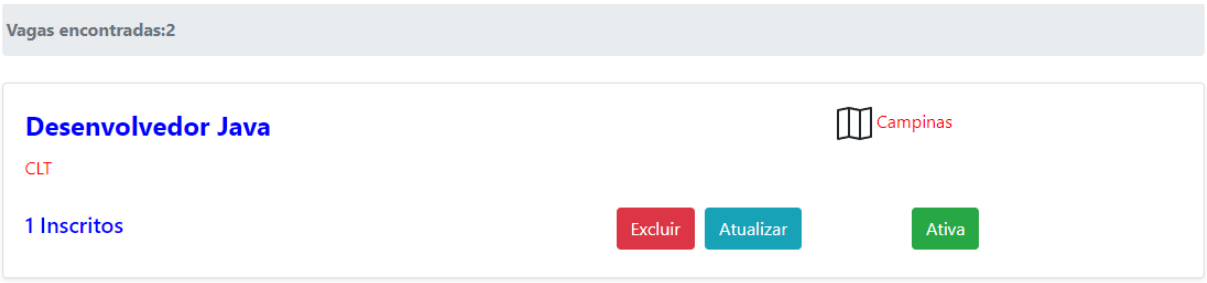

Figura 20: Tela de "**Consulta de Inscrição em Vagas**".

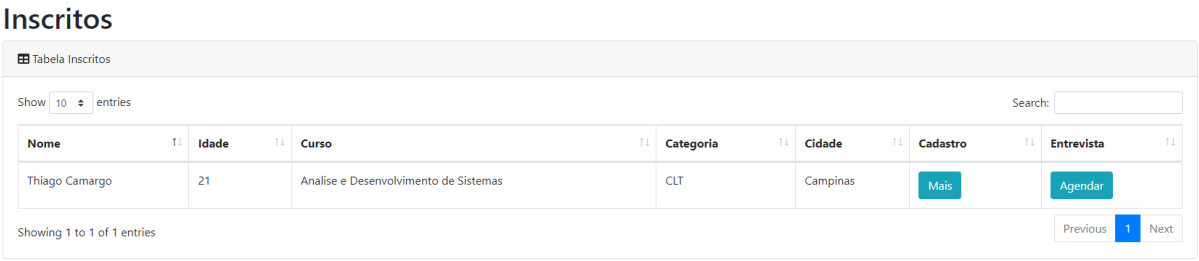

Figura 21: Tela de "**Inscritos em Vaga**".

Nessa *Sprint* foi concluído o desenvolvimento das funcionalidades previstas para o sistema, sendo possível ao candidato ver as vagas oferecidas pelas empresas, se candidatar-se, assim como as empresas poderem publicar suas vagas e verem os candidatos inscritos em cada vaga oferecida.

6. Resultados

Nessa seção são abordados os resultados obtidos no desenvolvimento do projeto, em relação às funcionalidades que foram previstas no início do projeto e as funcionalidades entregues ao fim do trabalho.

Tabela 4: Resultado das funcionalidades oferecidas.

Funcionalidade Prevista	**Funcionalidade Oferecida**
Candidato realizar seu cadastro	Sim
Empresa publicar vagas de TI	Sim
Sistema possuir filtragem de vagas	Sim
Candidato inscrever em vagas de TI	Sim
Sistema possuir filtragem de candidatos	Sim

1) **Candidato realizar seu cadastro:** Referente a essa funcionalidade, foi realizada a criação do cadastro em 3 etapas, em que a primeira parte diz respeito ao cadastro das informações pessoais do candidato (endereço, data de nascimento); a segunda parte é referente ao cadastro da experiência profissional do candidato (projetos realizados, experiências em outras empresas); por fim, a última parte refere-se às habilidades tecnológicas do candidato, verificando qual(ais) tecnologias/*frameworks*/ferramentas o mesmo conhece.

2) **Empresa publicar vagas de TI:** Conforme citado na proposta do projeto, o objetivo é focar apenas em vagas de TI, portanto a(s) empresa(s) pode(m) criar vagas de emprego, focando apenas em cargos de TI.

3) **Sistema possuir filtragem de vagas:** Para facilitar a busca de vagas do interesse do candidato foram criados alguns filtros, tais como: tipo de formação requerida na vaga (ensino médio, ensino superior completo/incompleto), categoria (CLT, estágio) e tecnologias (Java, Python).

4) **Candidato inscrever em vagas de TI:** O que diz respeito a essa funcionalidade é que é possível o candidato fazer sua inscrição em cada vaga, não permitindo se candidatar mais de uma vez na mesma vaga.

5) **Sistema possuir filtragem de candidatos:** Sobre essa funcionalidade, foi criada uma série de filtros para a(s) empresa(s) buscar o candidato ideal para ela(s). Os filtros criados foram referentes à formação do candidato, categoria de vaga, cidade em que o candidato mora, tecnologias que ele informou em seu cadastro, *frameworks* e ferramentas, informações essas coletadas durante o cadastro.

Sobre os testes realizados para validar se a funcionalidade atende os requisitos propostos, todos os testes foram executados de forma manual. Simulou-se a utilização do sistema por parte do usuário final, ou seja, para cada funcionalidade foi feito o preenchimento dos dados nos formulários e a validação posterior, verificando-se se todos os dados foram inseridos corretamente.

7. Conclusões

Como resultado do presente trabalho, chegou-se a uma aplicação *Web* que permite que candidatos e empresas se conheçam, através da plataforma de divulgação de vagas, plataforma está focada majoritariamente no mercado de TI. Em decorrência disso, pode-se concluir que o principal objetivo do trabalho de conclusão de curso foi alcançado. Além disso, destaca-se que o software desenvolvido é funcional, e cumpre com os requisitos básicos propostos no momento de sua concepção.

A arquitetura utilizada facilitou o desenvolvimento da aplicação, visto que o *backend* e *frontend* estão localizados na mesma aplicação, facilitando a transmissão dos dados via protocolo HTTP.

A metodologia utilizada (*Scrum*), fez com que o desenvolvimento fosse bem organizado, isto é, ao início de cada *sprint* já se sabe o que será feito naquele momento, tendo uma meta de prazo a ser seguida. Ao final de cada *sprint* a cerimônia de retrospectiva/demonstração - mesmo o trabalho sendo feito por uma única pessoa - ajuda a avaliar se a entrega da funcionalidade atende ao esperado no início do desenvolvimento. Assim evita-se o retrabalho, à medida que as demais partes do sistema vão se conectando.

Durante o desenvolvimento houve diversos momentos em que foram necessárias alterações no código-fonte e, em todos os casos, a metodologia empregada foi útil, justamente pelo fato de dividir o desenvolvimento em pequenas etapas contemplando funcionalidades específicas.

Dos conhecimentos adquiridos no curso de Análise e Desenvolvimento de Sistemas, foram utilizados, principalmente, os seguintes: lógica de programação, conhecimento em linguagem *Web*, Engenharia de Software, Linguagem de Banco de Dados, Interação Humano-computador (usabilidade) e Metodologia de Pesquisa Científica.

Como trabalhos futuros, podem-se acrescentar funcionalidades como: a) Preenchimento automático do currículo do usuário, por exemplo fazendo uso do Facebook ou Gmail para buscar os dados do candidato; b) Incorporação de alguma API que preencha endereço a partir do CEP; e c) Melhorias na usabilidade do usuário.

Além disso, algumas interfaces gráficas do sistema, ao serem acessadas por meio de dispositivo móvel (*smartphone*), apresentaram problemas de formatação - os *labels* e *inputs* ficaram desorganizados. Nesse caso seria desejável um trabalho posterior de melhoria da responsividade em diferentes dispositivos, o que potencialmente melhorará a experiência dos usuários.

A aplicação não contou com a criação de testes unitários. Por conta disso, e pensando em uma evolução futura do sistema, será interessante criá-los. Por último, outro trabalho futuro diz respeito à inclusão de *logs* nas funcionalidades, a fim de avaliar o comportamento dinâmico do sistema e facilitar eventuais manutenções.

8. Referências Bibliográficas

BRASSCOM. "Mercado de TI tem grande demanda e déficit de novos profissionais" Disponível em <https://brasscom.org.br/mercado-de-ti-tem-grande-demanda-edeficit-de-novosprofissionais/>. Acesso em 22 julho de 2021.

DEVMEDIA. "Técnicas Para Levantamento de Requisitos". Disponível em: <https://www.devmedia.com.br/tecnicas-para-levantamento-de-requisitos/9151>. Acesso em: 20 julho de 2021.

GETBOOTSTRAP. "*Documentation*". Disponível em: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>. Acesso em: 30 de julho de 2021.

GITHUB. "*GitHub*". Disponível em: < https://docs.github.com/pt/github>. Acesso em: 30 de julho de 2021.

ISO/IEC. "ISO/IEC 9126 Standard for Information Technology", "Software Product Evaluation – Quality Characteristics and Guidelines for their use". Geneve, 1991.

MDN WEB DOCS. "*JavaScript*". Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 30 de julho de 2021.

MEDEIROS, Higor. "Engenharia WEB Baseada na UML". DEVMEDIA. Disponível em: <https://www.devmedia.com.br/engenharia-web-baseada-na-uml/31001>. Acesso em 21 de julho de 2021.

SPRING. *Spring Framework*. Disponível em: <https://docs.spring.io/springframework/docs/current/reference/html/ > Acesso em 21 julho de 2021

SCRUM. "About Us". Disponível em: <https://www.scrum.org/about/ >. Acesso em: 20 de julho de 2021.

Documento Digitalizado Público

Artigo final de TCC do aluno Thiago Camargo da Silva

Assunto: Artigo final de TCC do aluno Thiago Camargo da Silva
Assinado por: Leandro Ledel
Tipo do Documento: Outro
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Leandro Camara Ledel**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 21/03/2022 10:38:36.

Este documento foi armazenado no SUAP em 21/03/2022. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 921015
Código de Autenticação: 0bd7469fde