

## DOCUMENTAÇÃO JHEICANAMA

Matutino – ANAD2BM1

## JHEICAFREE SISTEMA WEB

Professor: Geovane da Costa Oliveira

ADS: 2 Semestre

Matéria: Projeto Integrado de Certificação Scrum

Integrantes: Andreza Cassandra

Ícaro Lins

Jheffiny Gervazio

Julia Ferreira

Maria Clara

Natália Rodrigues

Matutino – ANAD2BM1

## Resumo

O presente projeto tem como objetivo o desenvolvimento de uma solução tecnológica voltada à mediação de serviços temporários, abrangendo trabalhos freelance, por diárias e de curta duração. A plataforma funcionará como intermediadora entre contratantes e prestadores de serviços, permitindo a publicação de vagas com informações detalhadas sobre requisitos, condições e valores. Paralelamente, os profissionais terão acesso a mecanismos de busca e filtragem que possibilitarão a identificação de oportunidades compatíveis com suas habilidades e disponibilidade. A proposta busca unir usabilidade, praticidade e eficiência, garantindo que a interação entre contratante e contratado ocorra em um ambiente seguro, centralizado e transparente. Para o desenvolvimento da aplicação, serão empregadas as linguagens HTML, CSS e JavaScript, com foco em responsividade, navegabilidade e acessibilidade.

## Sumário

1. Introdução .....	6
2. Revisão Literária .....	7
3. Objetivo Específico .....	8
3.1. Objetivo Geral .....	9
4. Justificativa .....	10
5. Metodologia .....	11
5.1. Metodologias Scrum.....	12
5.2. Papéis do Scrum.....	13
5.3. Cerimônias do Scrum .....	14
6. Cronograma .....	15
7. Quadro de Funções da Equipe .....	16
8. Lista de backlogs .....	17
8.1. Lista de prioridades produto backlogs .....	18
9. Quadro Kanban .....	19
10. Relatos das reuniões diárias.....	20
11. Entregas realizadas.....	23
11.1.Gráfico burndown Gráfico Burnup .....	24
12. Desenvolvimento.....	25
12.1. Camada de Serviço .....	26
12.2. Configuração de Segurança.....	27
12.3. Repositório de Dados .....	28
12.4. Candidato.....	29
12.5. Camada de Gerenciamento .....	30
12.6. Telas do sistema em funcionamento .....	32
12.7. Prototipagem .....	33
12.8. Login do usuário .....	34
12.9. Login do contratante .....	35
12.10. Cadastro do contratado .....	36
12.11. Cadastro do contratante.....	37
12.12. Desenvolvimento final .....	38
13. Resultados esperados .....	39

<b>14. Desafios enfrentados .....</b>	<b>40</b>
<b>15. Conclusão .....</b>	<b>41</b>
<b>16. Referências bibliográficas .....</b>	<b>42</b>

## 1. Introdução

Com o avanço da tecnologia e a crescente digitalização dos processos de trabalho, novas formas de contratação vêm ganhando espaço no mercado, especialmente no que se refere a serviços temporários e trabalhos autônomos. Nesse cenário, este projeto propõe o desenvolvimento de uma plataforma digital destinada à intermediação de serviços freelance e por diárias, voltada tanto para indivíduos que buscam oportunidades de renda complementar quanto para aqueles que necessitam contratar profissionais para demandas pontuais. A proposta consiste em oferecer um ambiente centralizado, acessível e confiável, que permita a divulgação e a busca de oportunidades de forma prática e eficiente. Dessa maneira, objetiva-se reduzir barreiras de comunicação, otimizar o processo de contratação e ampliar a visibilidade de prestadores de serviços, contribuindo para o fortalecimento do mercado de trabalho informal e para a geração de novas oportunidades.

## 2 Revisão Literária

O estudo da linguagem Java é amplamente consolidado na literatura técnica. Deitel (2016) apresentam uma abordagem abrangente da programação orientada a objetos, ressaltando conceitos fundamentais, boas práticas de codificação e exemplos aplicados, sendo referência indispensável para estudantes e profissionais da área de desenvolvimento.

De modo complementar, a documentação oficial da Oracle (2025) constitui fonte primária de consulta, oferecendo atualizações constantes, especificações técnicas e exemplos de aplicação. Esse material assegura que o desenvolvimento do projeto seja conduzido conforme os padrões atuais do mercado e com base em informações oficiais da linguagem.

Além disso, os cursos disponibilizados pela Digital Innovation One (DIO, 2025) demonstram a importância da formação prática, ao aproximar teoria e aplicação real em projetos. Recursos audiovisuais, como o curso de Java publicado por Santos (2023) no YouTube, também contribuem para a disseminação do conhecimento, permitindo uma aprendizagem acessível e dinâmica que complementa a literatura tradicional.

No âmbito da gestão de projetos, Schwaber e Sutherland (2020) apresentam o Guia do Scrum, no qual descrevem um framework ágil que organiza o desenvolvimento em sprints, possibilitando maior adaptabilidade e transparência no processo. Esse modelo de gerenciamento é amplamente utilizado em projetos de software e se mostra adequado para iniciativas que exigem entregas rápidas e contínuas.

Portanto, observa-se que a integração entre fundamentos teóricos (DEITEL; DEITEL, 2016), documentação oficial (ORACLE, 2025), recursos de ensino prático (DIO, 2025; SANTOS, 2023) e metodologias ágeis (SCHWABER; SUTHERLAND, 2020) oferece base consistente para o desenvolvimento do website de freelancers proposto, unindo aspectos técnicos, práticos e gerenciais.

### **3. Objetivo Específico**

Desenvolver um website responsivo e intuitivo utilizando as linguagens HTML, CSS e Java Script;

Disponibilizar funcionalidades que permitam ao contratante cadastrar e divulgar vagas com informações detalhadas sobre requisitos, valores e condições;

Implementar mecanismos de busca e filtragem que possibilitem ao prestador de serviços localizar oportunidades compatíveis com suas habilidades e disponibilidade;

Garantir a usabilidade e a navegabilidade do sistema, assegurando uma experiência satisfatória para os usuários;

Promover maior visibilidade para profissionais autônomos, ampliando suas possibilidades de inserção no mercado de trabalho;

Estimular a formalização e a organização dos processos de contratação, reduzindo a informalidade nas negociações atuais.

### **3.1 Objetivo Geral**

Desenvolver uma plataforma digital inovadora voltada para a intermediação de serviços temporários, abrangendo tanto trabalhos de natureza freelance quanto atividades realizadas por diárias, de forma a criar um ambiente virtual seguro, acessível e eficiente. O projeto busca atender às necessidades de dois públicos principais: os contratantes, que demandam profissionais qualificados e disponíveis de forma ágil, e os prestadores de serviços, que necessitam de uma ferramenta confiável para divulgar suas habilidades e conquistar novas oportunidades de renda.

Além de promover a aproximação entre essas partes, a plataforma pretende se diferenciar ao oferecer mecanismos de transparência e confiança, como avaliações mútuas, verificação de perfis e histórico de serviços prestados, possibilitando relações mais seguras e justas. Outro aspecto central é a preocupação com a inclusão e a democratização do acesso, garantindo que o sistema seja intuitivo, fácil de usar e adaptado a diferentes perfis de usuários, desde os mais experientes em tecnologia até aqueles com menor familiaridade digital.

A iniciativa também se propõe a fomentar o empreendedorismo individual e a valorização do trabalho temporário como alternativa flexível no mercado, contribuindo para o fortalecimento da economia colaborativa. Para isso, serão incorporados recursos tecnológicos modernos, que facilitem a comunicação, o gerenciamento de propostas e pagamentos, bem como ferramentas de filtragem personalizadas, permitindo que cada usuário encontre oportunidades ou prestadores de serviços que atendam de forma precisa às suas necessidades.

Portanto, o objetivo geral é criar uma solução digital que não apenas simplifique a intermediação entre contratantes e prestadores, mas que também se torne um espaço confiável, dinâmico e sustentável, capaz de gerar impacto positivo tanto no âmbito profissional quanto social

#### **4. Justificativa**

A crescente procura por trabalhos temporários e oportunidades freelance reflete mudanças significativas no mercado de trabalho contemporâneo, caracterizado pela flexibilidade, pela busca por autonomia e pela necessidade de complementação de renda. Contudo, ainda se observam dificuldades no processo de divulgação e de acesso a essas oportunidades, uma vez que muitos profissionais dependem de redes sociais, grupos informais ou indicações pessoais, o que limita a visibilidade e a confiabilidade das ofertas. Nesse contexto, a criação de uma plataforma digital unificada justifica-se pela necessidade de proporcionar um espaço seguro, centralizado e eficiente para a intermediação entre contratantes e prestadores de serviços. Além de contribuir para a organização e acessibilidade das informações, a solução proposta amplia as chances de empregabilidade de trabalhadores informais, ao mesmo tempo em que facilita a contratação para pessoas físicas e jurídicas que necessitam de serviços pontuais.

## 5 Metodologia

A empresa Jheicanama optou pela linguagem de programação Java, escolhida por nossos desenvolvedores devido à sua robustez, portabilidade e ampla aceitação no mercado. Essa linguagem permite a utilização da programação orientada a objetos, além de oferecer recursos como classes, métodos e interfaces. É importante destacar que, por ser executada em uma máquina virtual, garante maior acessibilidade e flexibilidade. Tendo em vista esses fatores, utilizamos o Java para o desenvolvimento do backend (sistema que o usuário não vê) do nosso sistema web de freelance. Já no frontend (o que o usuário vê e com o que interage), utilizamos um conjunto de tecnologias: o HTML para a estrutura textual, o CSS para a parte estética e, por fim, o JavaScript para trazer interatividade, além de realizar a integração com o código do backend. Dessa forma, construímos um sistema web que oferece serviços de freelance, proporcionando maior acessibilidade ao nosso público-alvo.

## 5.1 Metodologia Scrum

Scrum não é exatamente uma metodologia, mas sim um framework — uma estrutura dentro da qual as pessoas podem resolver problemas complexos e adaptativos, enquanto entregam produtos do mais alto valor possível de forma produtiva e criativa.

Ele é baseado no empirismo, que afirma que o conhecimento vem da experiência e da tomada de decisões com base no que é conhecido. O Scrum emprega uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar os riscos.

O Scrum é fundamentado em três pilares essenciais:

**Transparência:** Todos os aspectos importantes do processo devem ser visíveis para todos os envolvidos. Isso garante que todos tenham um entendimento comum do que está sendo feito e dos desafios existentes.

**Inspeção:** Os diversos aspectos do Scrum devem ser inspecionados com frequência para detectar variações indesejadas. A inspeção é realizada através dos eventos do Scrum.

**Adaptação:** Se a inspeção revelar que algo se desvia dos limites aceitáveis e que o produto resultante será inaceitável, o processo ou o material que está sendo processado deve ser ajustado.

## 5.2 Papéis do Scrum

**Product Owner (PO):** É o responsável por maximizar o valor do produto resultante do trabalho da equipe de desenvolvimento. É a única pessoa que gerencia o Product Backlog, o que inclui criar, priorizar e comunicar claramente os itens, garantindo que a equipe entenda o que precisa ser feito e por quê. O PO representa os interesses dos stakeholders.

**Scrum Master:** É o responsável por garantir que o time siga as práticas e os valores do Scrum. Ele não é um gerente de projetos tradicional, mas um líder-servidor que facilita os eventos do Scrum, remove impedimentos que atrapalham a equipe, protege o time de interferências externas e atua como um coach de agilidade para a equipe e para a organização.

**Desenvolvedores:** São os responsáveis por entregar uma versão Pronta do produto ao final de cada Sprint. A equipe é multifuncional, possuindo todas as habilidades necessárias para criar o incremento do produto. Eles são responsáveis por planejar o trabalho da Sprint e garantir a qualidade da entrega.

## 5.3 Cerimônias do Scrum

Os eventos no Scrum são projetados para criar regularidade e minimizar a necessidade de reuniões não definidas no framework. Todos os eventos têm um tempo máximo de duração (time-box).

**Sprint:** É o coração do Scrum. Um período de tempo fixo (geralmente de 1 a 4 semanas) durante o qual um incremento de produto "Pronto", utilizável e potencialmente liberável, é criado. Uma nova Sprint começa imediatamente após a conclusão da anterior.

**Sprint Planning (Planejamento da Sprint):** Ocorre no início de cada Sprint. O time inteiro colabora para definir o que pode ser entregue na Sprint que se inicia e como esse trabalho será realizado. O resultado é a Meta da Sprint e o Sprint Backlog.

**Daily Scrum (Reunião Diária):** Realizada diariamente durante a Sprint, é uma reunião rápida (no máximo 15 minutos) para a equipe de desenvolvimento sincronizar as atividades e criar um plano para as próximas 24 horas, inspecionando o progresso em direção à Meta da Sprint.

**Sprint Review (Revisão da Sprint):** Acontece ao final da Sprint. O Time Scrum e os stakeholders se reúnem para inspecionar o incremento do produto que foi construído. O Product Owner apresenta o que foi "Pronto" e o time demonstra o trabalho realizado, sendo uma oportunidade para obter feedback e adaptar o Product Backlog.

**Sprint Retrospective (Retrospectiva da Sprint):** Ocorre após a Sprint Review e antes da próxima Sprint Planning. É uma reunião para o Time Scrum inspecionar a si mesmo e criar um plano de melhorias a serem implementadas na próxima Sprint, focando no que correu bem e no que pode ser melhorado.

## 6 Cronograma

O cronograma foi elaborado com o objetivo de organizar as etapas do desenvolvimento do sistema Jheicafree, distribuindo as atividades de forma equilibrada ao longo do projeto. Ele serviu como guia para o acompanhamento do progresso da equipe, possibilitando uma visão clara sobre prazos, responsabilidades e prioridades de cada sprint.

Através do cronograma, foi possível definir metas semanais, como a criação das telas de login e cadastro, o desenvolvimento do backend e a finalização da documentação. Além disso, o cronograma facilitou a coordenação entre os membros da equipe, garantindo que cada função, desde o desenvolvimento até a prototipagem e documentação fosse executada dentro do tempo planejado.

Essa ferramenta foi essencial para o sucesso do projeto, pois permitiu o controle contínuo das entregas e ajudou a prevenir atrasos, assegurando que todas as fases do sistema fossem concluídas de maneira organizada e eficiente.

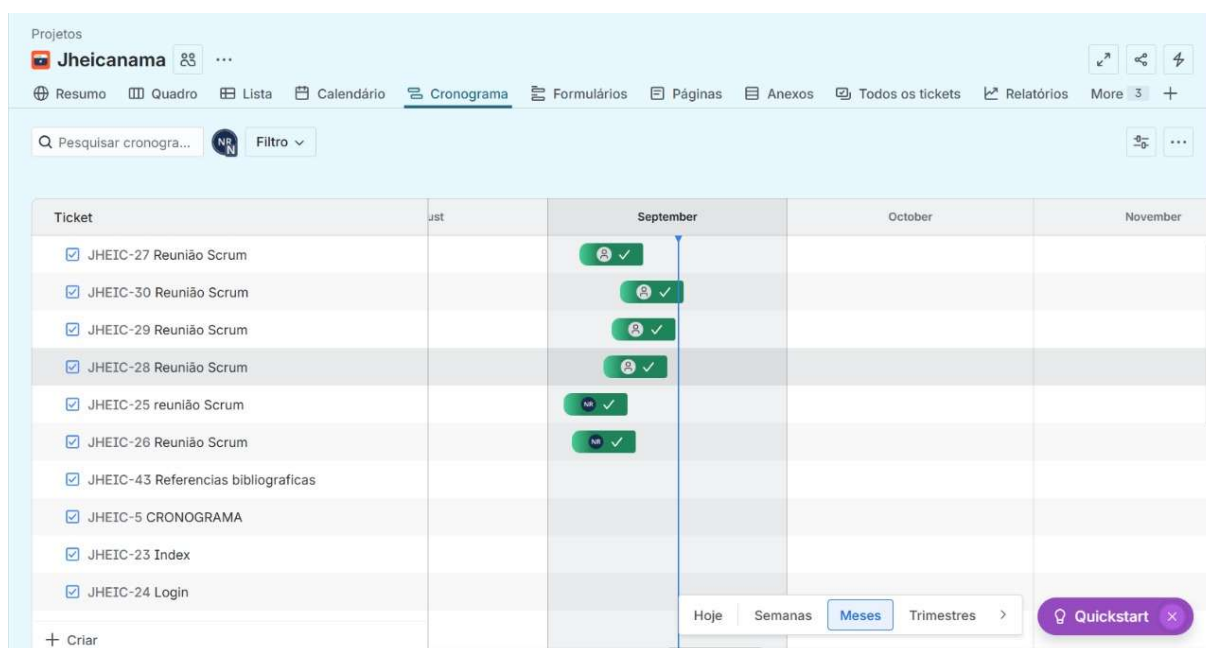


Figura 1 – Cronograma

## 7 Quadro de Funções da Equipe

Gerente De Projetos	Natalia Rodrigues	O gerente de projeto é quem planeja, organiza e acompanha todas as etapas do trabalho. Ele é responsável por garantir que os prazos sejam cumpridos, que os custos fiquem dentro do previsto e que a comunicação entre a equipe e o cliente aconteça de forma clara e eficiente.
Analista De Requisitos	Andreza Cassandra	O analista de requisitos é a pessoa que conversa com os usuários e clientes para entender suas necessidades, transformando essas informações em requisitos que servirão de guia para o desenvolvimento do sistema. Quando também exerce o papel de Scrum Master, atua como um facilitador, ajudando a equipe a seguir as práticas ágeis, promovendo reuniões, resolvendo obstáculos e mantendo o bom andamento do projeto.
Desenvolvedores	Ícaro Lins & Maria Clara	Os desenvolvedores são os profissionais que transformam os requisitos em realidade, criando o software por meio da programação. Eles constroem as funcionalidades, fazem testes básicos e garantem que o sistema atenda ao que foi solicitado, seja na parte visual, na lógica ou na integração com bancos de dados.
Documentadores	Jheffiny Gervazio & Julia Ferreira	O documentador, é quem registra todas as informações importantes do projeto. Ele elabora atas, relatórios, manuais e demais registros que organizam e documentam o processo, permitindo que a equipe e os clientes tenham acesso às informações de forma clara e estruturada.

## 8 Lista de backlogs

• Cadastro De Usuário
• Login De Usuário
• Logout Do Usuário
• Segurança
• Página Inicial
• Backend
• Frontend
• Banco De Dados
• Expirar Conta
• Prototipagem
• Documentação
• Criptografia
• Página Do Contratante
• Página De Login
• Tela Inicial
• Interface/Index

## 8.1 Lista de prioridades produtos backlogs

Categoria	Itens Prioritários
Backend	Login do usuário, cadastro, criptografia, página do contratante, página inicial, logout do usuário, expirar conta, esqueci minha senha.
Prototipagem	Página de login, tela inicial.
Banco de Dados	
Frontend	Página inicial, login do usuário, cadastro, logout, interface/index, expirar conta, esqueci minha senha.
Documentação	

## 9. Quadro Kanban

O quadro Kanban foi utilizado para organizar e visualizar o fluxo de trabalho da equipe, garantindo que todas as etapas do desenvolvimento fossem acompanhadas de forma clara e eficiente. As tarefas foram divididas em colunas representando os status “A Fazer”, “Em Progresso” e “Concluído”, o que facilitou o controle das entregas e a priorização dos backlogs. Essa ferramenta permitiu maior transparência entre os membros da equipe, otimizando a comunicação e ajudando a identificar erros. Por meio do Kanban, foi possível manter o ritmo de produção e garantir que cada sprint fosse concluída dentro do prazo estabelecido.

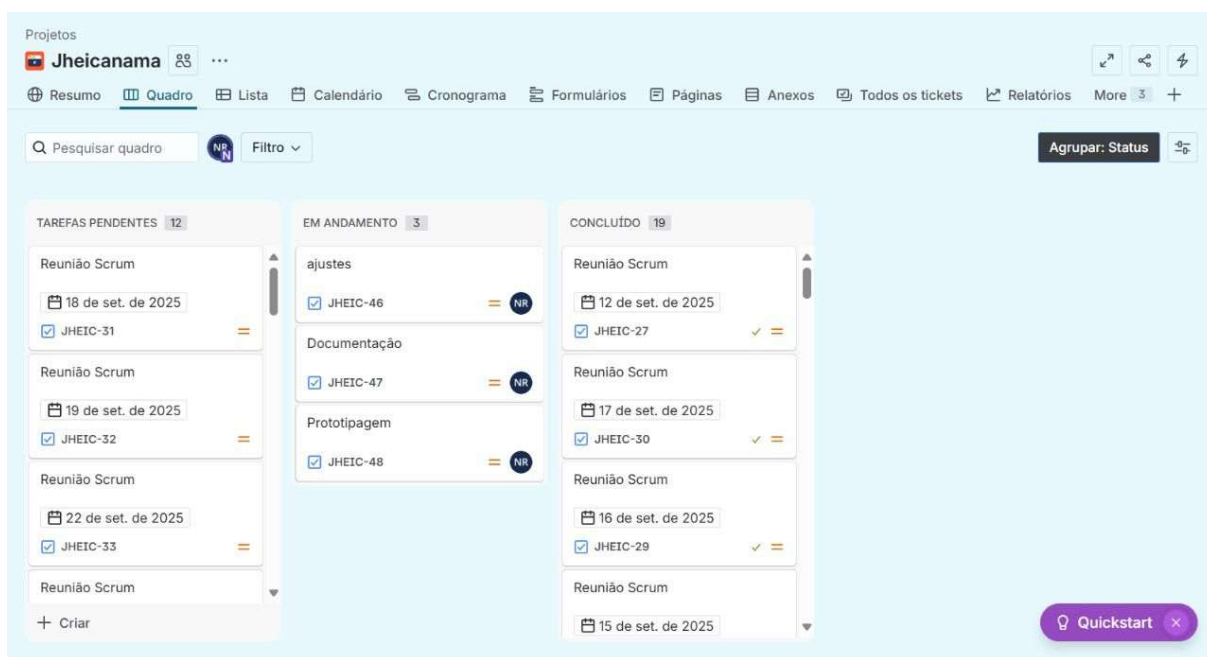


Figura 2 – Quadro Kanban

## **10. Relatos das reuniões diárias**

Ao longo das reuniões diárias, a equipe discutiu e acompanhou o andamento do projeto, com foco na entrega de backlogs, desenvolvimento do backend e prototipagem de telas. Inicialmente, o Scrum destacou a importância de alinhar a terminologia do sistema, definindo que o termo utilizado seria “Sistema Web” em vez de site. Nesta mesma reunião, identificaram uma pendência: a prototipação da tela de login não havia sido entregue, o que impactou o fluxo de atividades. Em questão ao Trello, foi indicado como ferramenta alternativa para gestão de tarefas em caso de falhas no GPA/GitHub.

No progresso do projeto, foram deliberadas entregas de artefatos, discutida a estratégia de segurança de dados por meio de tokens criptográficos e planejados os avanços futuros, em especial a implementação dos bancos de dados. Nessas ocasiões, a equipe também realizou demonstrações técnicas com amostras de códigos para comprovar o progresso.

Sobre o andamento do backend, a equipe reportou a conclusão integral do módulo de cadastro de contratante, além de ajustes estratégicos nas entregas. Ficou estabelecido que os desenvolvedores devem documentar metodologias utilizadas, explicar a linguagem de programação escolhida e incluir prints de tela do código para registro.

Em relação à implementação prática, a equipe iniciou o desenvolvimento da página principal (index), aplicando a metodologia de trabalho definida. Além disso, foram registrados novos requisitos, como a criação de uma tela de login e a inclusão de opção de gênero na plataforma, pontos destacados como atenção no cronograma de entregas.

Foram feitos, nas reuniões, ajustes de escopo, e a prioridade passou a ser a finalização do backend em substituição ao frontend do login, e o Scrum Master ficou responsável por ajustar o planejamento na ferramenta de gestão. Em paralelo, houve relatos sobre a necessidade de estruturar dois processos de cadastro distintos para

atender melhor às exigências do sistema, bem como o avanço na criação de um repositório de consulta de contratantes por CNPJ.

No decorrer das reuniões, também foram abordadas pendências de documentações, melhorias na transparência das entregas e a priorização de backlogs. A equipe demonstrou empenho em cumprir metas, resolvendo parte das tarefas planejadas, com destaque para a finalização do protótipo da tela de login e o backend do cadastro de contratante como pontos centrais do progresso.

A equipe de desenvolvimento deu início à fase de implementação do projeto. O backlog da sprint, com todas as atividades feitas, recebeu apenas alguns ajustes.

A equipe de desenvolvimento destacou que o sistema está funcionando com tudo feito. E a equipe da documentação está registrando tudo do início ao fim.

## 11. Entregas realizadas

- Funcionalidade de login (backend) – analisada e revisada pelo Product Owner.
- Backlogs entregues – foram apresentados e revisados.
- Documentação das metodologias utilizadas. Desenvolvimento (prints e exemplos de códigos) – responsabilidades atribuídas e execução.
- Seção de referência teórica – está sob responsabilidade da gerente de projetos (entrega parcial, mas com necessidade de aprimoramento).
- Prototipação da tela de login do usuário.
- Referência teórica.
- Alinhamento da prototipagem com a página de login.
- Decisões tomadas e ações definidas.
- Uso do termo correto: o projeto deve ser chamado de “SISTEMA WEB”, não de “site”.

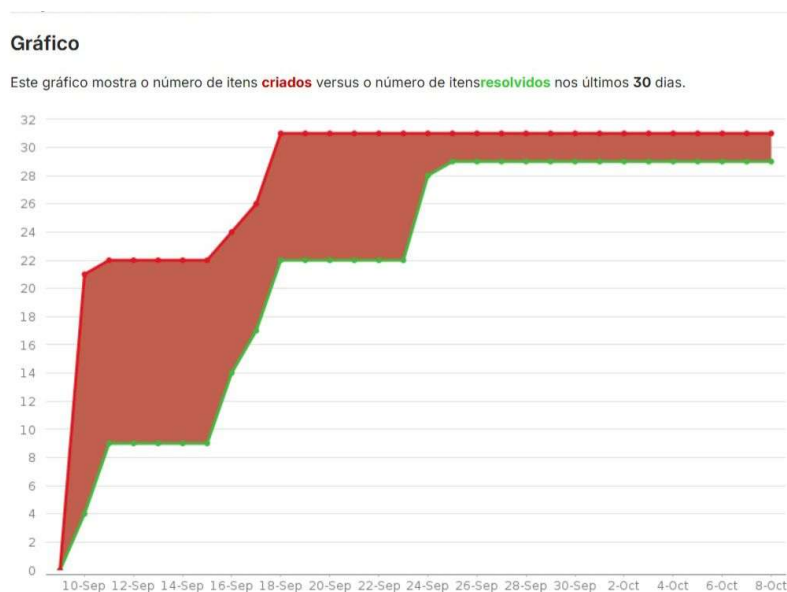
## 11.1. Gráfico Burndown e Gráfico Burnup

O gráfico Burndown foi empregado para acompanhar o ritmo de execução das tarefas durante cada sprint, representando a quantidade de trabalho restante ao longo do tempo. Com ele, a equipe pôde identificar se o progresso estava de acordo com o cronograma e ajustar o planejamento quando necessário. No caso do Jheicafree, o Burndown evidenciou um bom desempenho coletivo, com redução gradual das pendências até a conclusão das atividades planejadas. Essa ferramenta foi essencial para manter a equipe focada, cumprir os prazos e assegurar que as entregas fossem realizadas dentro do tempo previsto.

O gráfico Burnup foi utilizado para monitorar o progresso acumulado do projeto em relação às metas estabelecidas no Product Backlog. Ele mostra como as entregas evoluíram ao longo do tempo, evidenciando a quantidade de trabalho concluído e a meta total de tarefas a serem finalizadas.

Durante o desenvolvimento do Jheicafree, o gráfico demonstrou crescimento contínuo na quantidade de entregas concluídas, refletindo a produtividade da equipe e o cumprimento das metas de cada sprint. Essa visualização ajudou o grupo a compreender o avanço do projeto de forma positiva, além de reforçar o alinhamento entre o planejamento e as entregas realizadas.

Figura 3 – Gráfico Burnup

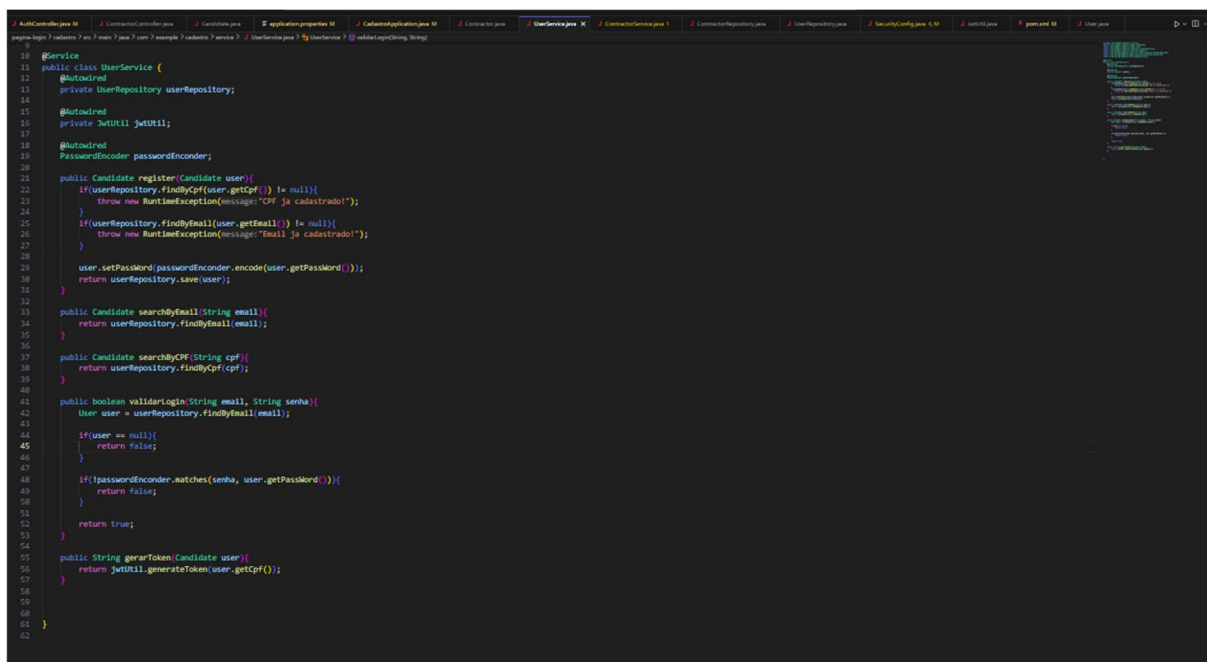


## 12. Desenvolvimento

O projeto foi desenvolvido em Java utilizando o framework Spring Boot, que é bastante usado no meio corporativo por simplificar a criação de sistemas. Esse framework já traz configurações automáticas, ferramentas de segurança, conexão com banco de dados e até um servidor embutido, o que permite que a aplicação funcione de forma independente e sem complicações extras. O coração da aplicação está na classe principal, chamada CadastroApplication, que funciona como o botão de Iniciar do sistema. Quando essa classe é executada, o Spring Boot inicia todo o ciclo de vida da aplicação, carregando os componentes necessários, como entidades, serviços e repositórios, além de disponibilizar um servidor pronto para receber requisições.

## 12.1. Camada de Serviço

As regras de funcionamento do sistema ficam concentradas nas chamadas “camadas de serviço”. Nelas temos, por exemplo, a *UserService*, responsável pela lógica de negócios relacionada aos candidatos, e a *ContractorService*, que faz o mesmo trabalho, mas para os contratantes. Essas classes cuidam de operações como registro, login e validação de dados, além de garantir a segurança das informações através da criptografia de senhas e da geração de tokens de autenticação. Assim, elas funcionam como o cérebro do sistema, organizando e controlando a forma como os usuários interagem com a aplicação.



```
10 @Service
11 public class UserService {
12     @Autowired
13     private UserRepository userRepository;
14
15     @Autowired
16     private JwtUtil jwtUtil;
17
18     @Autowired
19     PasswordEncoder passwordEncoder;
20
21     public Candidate register(Candidate user) {
22         if (userRepository.findByCpf(user.getCpf()) != null) {
23             throw new RuntimeException(message: "CPF já cadastrado!");
24         }
25         if (userRepository.findByEmail(user.getEmail()) != null) {
26             throw new RuntimeException(message: "Email já cadastrado!");
27         }
28         user.setPassword(passwordEncoder.encode(user.getPassword()));
29         return userRepository.save(user);
30     }
31
32     public Candidate searchByEmail(String email) {
33         return userRepository.findByEmail(email);
34     }
35
36     public Candidate searchByCPF(String cpf) {
37         return userRepository.findByCpf(cpf);
38     }
39
40     public boolean validateLogin(String email, String senha) {
41         User user = userRepository.findByEmail(email);
42         if (user == null) {
43             return false;
44         }
45         if (!passwordEncoder.matches(senha, user.getPassword())) {
46             return false;
47         }
48         return true;
49     }
50
51     public String gerarToken(Candidate user) {
52         return jwtUtil.generateToken(user.getCpf());
53     }
54 }
55
56
57
58
59
60
61
62 }
```

Figura 4 – Camada de Serviço

## 12.2. Configuração da Segurança

Outro ponto central é a segurança, configurada pela classe `SecurityConfig`. É nela que se define quais partes do sistema podem ser acessadas livremente e quais exigem login. Rotas como as de cadastro e login ficam abertas para qualquer pessoa, enquanto todas as demais só podem ser acessadas após autenticação. Além disso, essa configuração assegura que as senhas nunca sejam guardadas em texto simples no banco de dados, aumentando a proteção contra invasões e vazamentos de informação.

```
pagina-login > cadastro > src > main > java > com > example > cadastro > security > SecurityConfig.java > ...
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
6 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
7 import org.springframework.security.crypto.password.PasswordEncoder;
8 import org.springframework.security.web.SecurityFilterChain;
9
10 @Configuration
11 public class SecurityConfig {
12     @Bean
13     public PasswordEncoder passwordEncoder(){
14         return new BCryptPasswordEncoder();
15     }
16
17     @Bean
18     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
19         http
20             .csrf().disable()
21             .authorizeHttpRequests()
22                 .requestMatchers(...patterns: "/usuarios/register", "/usuarios/login").permitAll()
23                 .anyRequest().authenticated()
24             .and()
25             .httpBasic(); // só para testes, depois você troca por JWT
26         return http.build();
27     }
28 }
29
30
31
```

Figura 5 - Configuração da Segurança

## 12.3. Repositório de Dados

O acesso aos dados é feito pelos repositórios, que são interfaces que conversam diretamente com o banco de dados. O UserRepository é responsável pelos candidatos e o ContractorRepository pelos contratantes. Graças ao Spring Data JPA, esses repositórios já vêm com métodos prontos para operações comuns, como salvar, atualizar, buscar ou deletar informações, além de permitir a criação de consultas personalizadas, como procurar usuários pelo e-mail, CPF ou CNPJ.

```
pagina-login > cadastro > src > main > java > com > example > cadastro > repository > ContractorRepository.java > ContractorRepository
1 package com.example.cadastro.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.example.cadastro.model.Contractor;
7
8 @Repository
9 public interface ContractorRepository extends JpaRepository<Contractor, Long>{
10     Contractor findByEmail(String email);
11     Contractor findByCnpj(String cnpj);
12 }
13
```

Figura 6 - Repositório de Dados

## 12.4. Candidato

A forma como os dados são organizados está definida nos modelos da aplicação. A classe `User` funciona como uma base, reunindo atributos comuns, como nome, e-mail e senha, e evitando repetição de código. A partir dela, nascem classes mais específicas: `Contractor`, que adiciona informações de contratantes, como CNPJ, empresa e telefone, e `Candidate`, que acrescenta CPF e data de nascimento. Esses modelos também possuem regras de validação para garantir que os dados sejam corretos antes mesmo de serem salvos no banco, como verificar se o CPF tem 11 dígitos ou se a data de nascimento é realmente no passado.

```
pagina-login > cadastro > src > main > java > com > example > cadastro > model > Candidate.java > Candidate
1 package com.example.cadastro.model;
2 import java.time.LocalDate;
3
4 import jakarta.persistence.Entity;
5 import jakarta.validation.constraints.NotBlank;
6 import jakarta.validation.constraints.Past;
7 import jakarta.validation.constraints.Size;
8
9 @Entity
10 public class Candidate extends User {
11     @NotBlank(message = "CPF é obrigatório")
12     @Size(min = 11, max = 11, message = "CPF deve ter 11 caracteres")
13     private String cpf;
14
15     @Past(message = "Data de nascimento deve ser no passado")
16     private LocalDate dataNascimento;
17
18     public String getCpf() {
19         return cpf;
20     }
21
22     public void setCpf(String cpf) {
23         this.cpf = cpf;
24     }
25
26     public LocalDate getDataNascimento() {
27         return dataNascimento;
28     }
29
30     public void setDataNascimento(LocalDate dataNascimento) {
31         this.dataNascimento = dataNascimento;
32     }
33
34 }
```

Figura 7 - Candidato

## 12.5. Camada de Gerenciamento

Por fim, existem os controladores, que são a porta de entrada da aplicação. O ContractorController e o AuthController recebem as requisições dos usuários, como pedidos de cadastro ou login, e encaminham essas informações para as classes de serviço. Eles também cuidam do retorno das respostas, incluindo mensagens de erro ou a geração de tokens de autenticação quando o login é bem-sucedido. Em alguns casos, ainda permitem que o usuário recupere seus próprios dados de forma segura, usando o token como chave de acesso.

```
pagina-login > cadastro > src > main > java > com > example > cadastro > controller > ContractorController.java > ...
15
16 @RestController
17 @RequestMapping("/contratantes")
18 @CrossOrigin(origins="*")
19 public class ContractorController {
20
21     @Autowired
22     private ContractorService contractorService;
23
24     @Autowired
25     private JwtUtil jwtUtil;
26
27     @PostMapping("/register")
28     public ResponseEntity<> cadastrar(@RequestBody Contractor user){
29         try{
30             Contractor novo = contractorService.register(user);
31             return ResponseEntity.ok(novo);
32         } catch (RuntimeException e){
33             return ResponseEntity.badRequest().body(e.getMessage());
34         }
35     }
36
37     @PostMapping("/login")
38     public ResponseEntity<> login(@RequestParam String email, @RequestParam String senha){
39         boolean valido = contractorService.validarLogin(email, senha);
40
41         if(valido){
42             Contractor user = contractorService.searchByEmail(email);
43             String token = jwtUtil.generateToken(user.getEmail());
44             return ResponseEntity.ok(token);
45         }
46
47         return ResponseEntity.status(status:401).body(body:"Email/Cnpj ou senha incorreta");
48     }
49 }
50
```

Figura 8 - Camada de Gerenciamento

```

10 @RestController
11 @RequestMapping("/usuarios")
12 @CrossOrigin(origins = "*")
13 public class AuthController {
14
15     @Autowired
16     private UserService userService;
17
18     @Autowired
19     private JwtUtil jwtUtil;
20
21     @PostMapping("/register")
22     public ResponseEntity<> cadastrar(@RequestBody Candidate usuario){
23         try{
24             Candidate novo = userService.register(usuario);
25             return ResponseEntity.ok(novo);
26         } catch (RuntimeException e) {
27             return ResponseEntity.badRequest().body(e.getMessage());
28         }
29     }
30
31     @PostMapping("/login")
32     public ResponseEntity<> login(@RequestParam String email, @RequestParam String senha) {
33         boolean valido = userService.validarLogin(email, senha);
34
35         if (valido) {
36             Candidate user = userService.searchByEmail(email);
37             String token = jwtUtil.generateToken(user.getCpf());
38             return ResponseEntity.ok(token);
39         }
40         return ResponseEntity.status(401).body("Email ou senha inválidos");
41     }
42
43     @GetMapping("/{cpf}")
44     public ResponseEntity<> searchByCpf(@PathVariable String cpf){
45         Candidate user = userService.searchByCpf(cpf);
46         if (user != null){
47             return ResponseEntity.ok(user);
48         }
49         return ResponseEntity.notFound().build();
50     }
51
52     @GetMapping("/me")
53     public ResponseEntity<> getMeData(@RequestHeader("Authorization") String token){
54         if (token.startsWith("Bearer ")){
55             token = token.substring(token.indexOf(" ") + 1);
56         }
57         if (!jwtUtil.validateToken(token)){
58             return ResponseEntity.status(401).body("Token inválido ou expirado");
59         }
60         String cpf = jwtUtil.extractCpf(token);
61         User user = userService.searchByCpf(cpf);
62         return ResponseEntity.ok(user);
63     }
64 }

```

Figura 9 – Camada de Gerenciamento

## 12.6 Telas do Sistema em Funcionamento

Nesta seção, são apresentadas e analisadas as principais telas que compõem a porta de entrada do sistema 'jheicafree'. O foco recai sobre os fluxos de autenticação (login) e registro (cadastro), que são cruciais para a segurança e para a segmentação dos usuários. Serão demonstradas as funcionalidades de cada interface e como a interação foi projetada para ser clara e eficiente para os dois perfis de usuário da plataforma: o Freelancer, que busca oportunidades, e o Contratante, que oferece as vagas. O objetivo é ilustrar visualmente como o sistema acolhe e direciona cada tipo de usuário desde o seu primeiro acesso, validando a implementação dos requisitos funcionais do projeto.

## 12.7 Prototipagem

A prototipagem foi uma etapa essencial no desenvolvimento do projeto, pois permitiu visualizar e testar a aparência e o funcionamento do sistema antes da implementação final. Por meio dela, foi possível definir o layout das páginas, a navegação entre telas e a disposição dos elementos visuais, garantindo uma melhor experiência ao usuário.

Durante essa fase, foram criados protótipos das principais interfaces, como a página de login, tela inicial e página do contratante, representando o fluxo de interação do usuário dentro do sistema. Essa abordagem ajudou a identificar melhorias de usabilidade e a alinhar o design com os requisitos funcionais do projeto.

A prototipagem serviu, portanto, como uma ponte entre o planejamento e o desenvolvimento, facilitando a comunicação entre a equipe e reduzindo o retrabalho nas etapas seguintes.



Figura 10 – prototipagem

## 12.8. Login do usuário

Esta é a tela principal de autenticação, projetada para o acesso de usuários prestadores de serviço (freelancers). O layout é dividido em duas partes: à esquerda encontra-se a seção de boas-vindas, que apresenta o slogan *“Aqui começa a sua nova oportunidade”*, acompanhado de um texto motivacional com o objetivo de atrair e engajar os usuários, enquanto à direita está localizado o formulário de login.

O formulário de login é composto pelo campo CPF/EMAIL, no qual o usuário pode optar por entrar utilizando seu CPF ou o endereço de e-mail cadastrado, e pelo campo “SENHA”, destinado à inserção da senha secreta, com caracteres mascarados para garantir a segurança. Além disso, há o botão Logar, que valida as credenciais e, caso estejam corretas, redireciona o usuário para seu painel de controle.

Outros elementos complementam a interface, como o botão Cadastra-se, que direciona para a tela de cadastro de freelancer, voltado para aqueles que ainda não possuem conta; o botão Contratante, que funciona como um seletor e alterna o formulário para a versão de login do contratante; e o link Esqueci minha senha, responsável por iniciar o fluxo de recuperação de senha.



Figura 11 – Login do usuário

## 12.9 Login do Contratante

Esta tela é a porta de entrada para usuários que desejam contratar serviços. O design mantém a consistência visual com a tela de login do freelancer, porém o formulário é adaptado para as credenciais de uma pessoa jurídica. A principal diferença está no campo **CNPJ/EMAIL**, que solicita o CNPJ como identificação padrão para empresas, funcionando também como alternativa ao e-mail. O campo **SENHA** preserva a mesma funcionalidade de inserção segura da senha. Os botões e links disponíveis seguem a mesma lógica da tela anterior, mas direcionam o usuário para as áreas específicas destinadas ao contratante.



The screenshot shows a login interface for 'jheicafree'. On the left, there is a motivational message: 'Aqui começa a sua nova oportunidade' (Here begins your new opportunity), followed by 'Venha trabalhar conosco: praticidade e dinheiro mais perto de você. Empregos rápidos e com maior portabilidade. Entre para nossa comunidade!!' (Come work with us: convenience and money closer to you. Quick jobs and with greater portability. Join our community!!). Below this are social media handles '@jheicanabfreelance' and '/jheicanabfreelance'. On the right, a 'LOGIN' box contains a 'CNPJ/EMAIL' input field, a 'SENHA' input field, a 'cadastra-se' (sign up) button, and a 'Contratante' (contractor) button. A 'Logar' (login) button is also present. A link 'Esqueci minha senha' (I forgot my password) is located below the login box. The background features a stylized illustration of people working in an office environment.

Figura 12 – Login do contratante

## 12.10. Cadastro do contratado

Este formulário é destinado ao registro de novos prestadores de serviço, solicitando as informações essenciais para a criação de um perfil de pessoa física na plataforma. Entre os campos disponíveis estão **“Nome completo”**, **“E-mail”**, **“CPF”**, **“Crie uma senha”** e o campo de data no formato **“dd/mm/aaaa”**.

Além disso, a interface apresenta três botões de ação. O botão **“FINALIZAR”** é responsável por validar os dados e, caso estejam corretos, criar a conta do freelancer. O botão **“Voltar”** permite ao usuário cancelar o cadastro e retornar à tela anterior. Já o botão **“Contratante”** funciona como um seletor, alternando para o formulário de cadastro destinado aos contratantes.

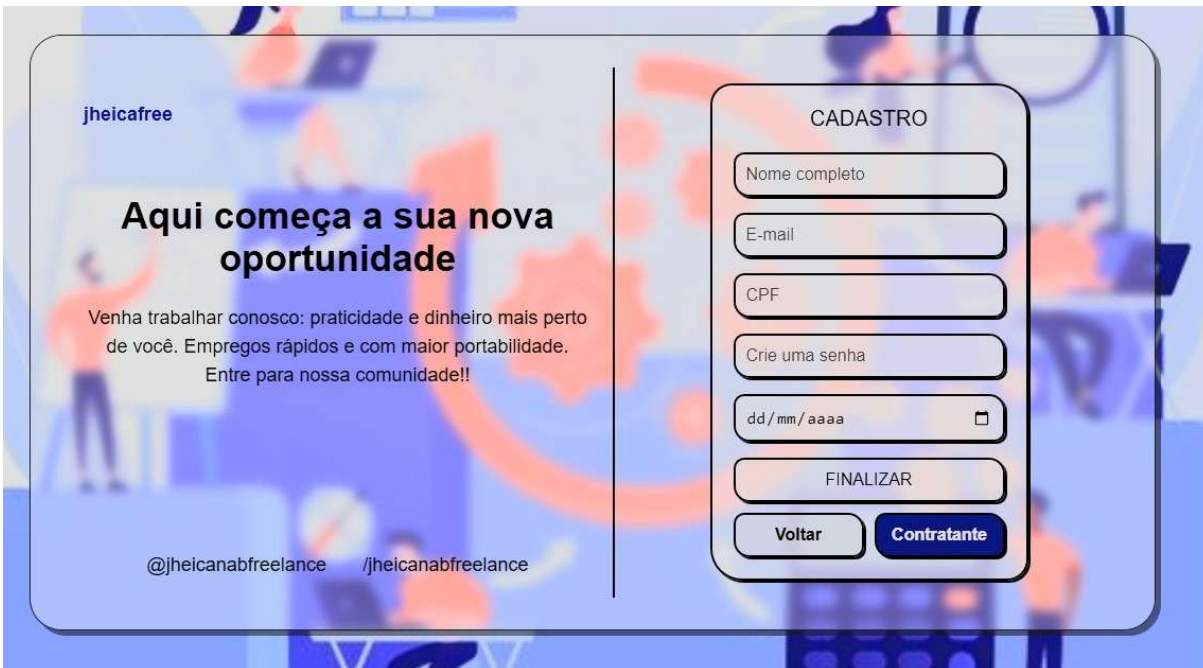
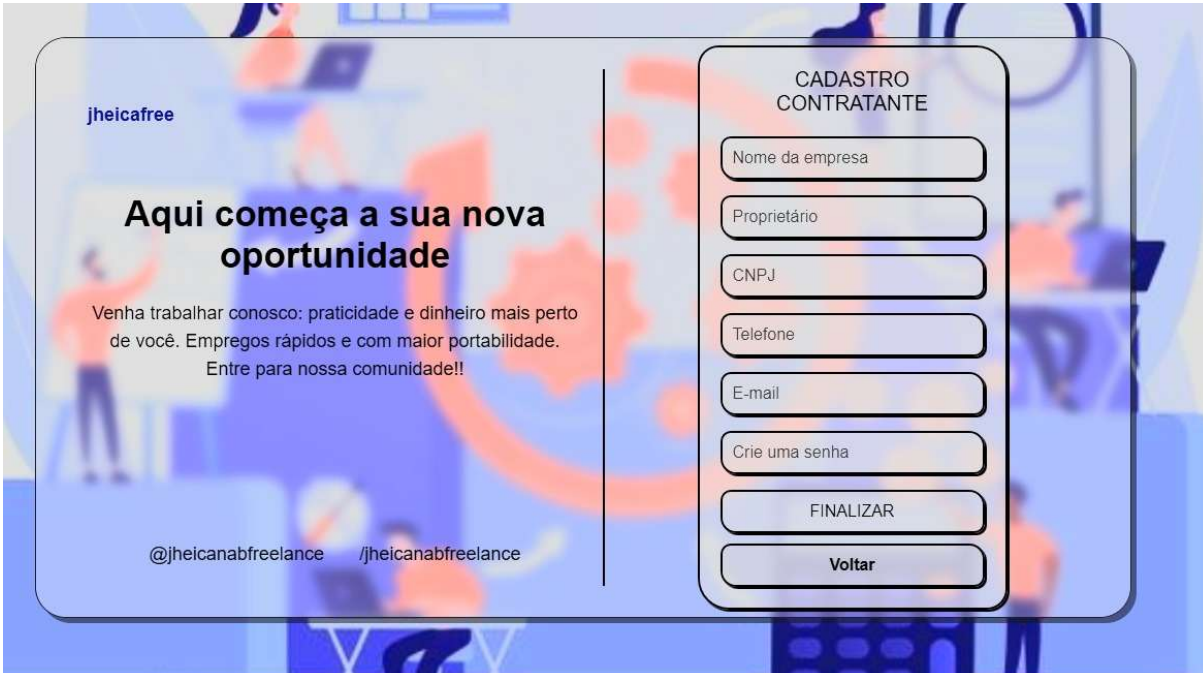


Figura 13 – Cadastro do contratado

## 12.11. Cadastro do Contratante

Esta tela permite que empresas e empregadores se cadastrem na plataforma, sendo especificamente desenhada para coletar informações de pessoa jurídica. O formulário solicita dados como nome da empresa, proprietário, CNPJ, telefone, e-mail e a criação de uma senha. O botão "FINALIZAR" valida e submete os dados inseridos, criando a conta do contratante, enquanto o botão "Voltar" cancela a operação e retorna à tela anterior.



**jheicafree**

**Aqui começa a sua nova oportunidade**

Venha trabalhar conosco: praticidade e dinheiro mais perto de você. Empregos rápidos e com maior portabilidade. Entre para nossa comunidade!!

@jheicanabfreelance /jheicanabfreelance

**CADASTRO CONTRATANTE**

Nome da empresa

Proprietário

CNPJ

Telefone

E-mail

Crie uma senha

FINALIZAR

Voltar

Figura 14 – cadastro do contratante

## **12.12. Desenvolvimento final**

O sistema foi construído seguindo uma arquitetura em camadas bem definida: a classe principal inicia tudo, os serviços aplicam as regras de negócio, os repositórios cuidam do banco de dados, os modelos estruturam as informações e os controladores fazem a ligação entre o usuário e o restante do sistema. Essa organização torna a aplicação mais clara, segura e fácil de manter, além de garantir que cada parte tenha uma responsabilidade.

### **13. Resultados esperados**

Disponibilizar uma plataforma digital funcional e responsiva, desenvolvida em HTML, CSS e JavaScript, que possibilite a intermediação entre contratantes e prestadores de serviços temporários.

Garantir que a aplicação ofereça interface intuitiva e de fácil navegação, permitindo que usuários sem conhecimentos técnicos consigam utilizá-la de forma autônoma.

Proporcionar maior eficiência no processo de contratação, reduzindo o tempo de busca por profissionais ou oportunidades de trabalho freelance.

Assegurar segurança e confiabilidade na interação entre as partes, por meio de cadastro de usuários e sistema de login protegido.

Contribuir para a geração de renda complementar de prestadores de serviços, oferecendo um ambiente que centralize a divulgação e a busca por vagas temporárias.

Estimular a formalização de relações de trabalho temporárias, atuando como intermediador transparente e confiável entre contratante e contratado.

## 14. Desafios enfrentados

Garantir usabilidade e acessibilidade: desenvolver uma interface que seja simples e intuitiva para diferentes perfis de usuários, desde contratantes experientes em tecnologia até prestadores com menor familiaridade digital.

Segurança de dados: implementar mecanismos básicos de proteção de informações (cadastro, login e senhas), reduzindo riscos de uso indevido da plataforma.

Gerenciamento das interações entre usuários: criar um fluxo eficiente de publicação de vagas, candidaturas e comunicação, evitando falhas que comprometam a experiência.

Limitações técnicas: trabalhar apenas com HTML, CSS e JavaScript, sem recursos avançados de backend, o que pode restringir funcionalidades como sistemas de pagamento ou armazenamento em larga escala.

Adesão dos usuários: superar a dificuldade inicial de atrair contratantes e prestadores para utilizarem a plataforma, essencial para validar sua efetividade.

Responsividade: garantir que o site funcione corretamente em diferentes dispositivos, mantendo desempenho satisfatório.

## 15. Conclusão

O desenvolvimento do sistema web "Jheicafree" atingiu com sucesso o objetivo de criar uma plataforma robusta e funcional para a intermediação de serviços temporários. Ao longo do projeto, foi construída uma solução que atende diretamente à crescente demanda por um ambiente digital seguro e centralizado, capaz de conectar contratantes e prestadores de serviços de forma eficiente.

A metodologia adotada, que combinou o uso de Java com o framework Spring Boot para o backend e tecnologias consolidadas como HTML, CSS e JavaScript para o frontend, provou-se uma escolha acertada. Essa abordagem permitiu a criação de uma arquitetura em camadas bem definida, onde cada componente possui uma responsabilidade clara, desde os controladores que gerenciam as requisições até os repositórios que fazem a interface com o banco de dados. O resultado é um sistema organizado, seguro e de fácil manutenção.

O projeto não apenas entrega as funcionalidades essenciais propostas, como o cadastro de usuários e a publicação de vagas, mas também estabelece uma base sólida para futuras expansões. A implementação de recursos de segurança, como a criptografia de senhas e a autenticação via tokens, garante a proteção dos dados e a confiabilidade da plataforma.

Dessa forma, este trabalho representa uma contribuição significativa para o mercado de trabalho autônomo, oferecendo uma ferramenta que otimiza o processo de contratação, amplia a visibilidade de profissionais e fomenta a economia colaborativa. O sistema "Jheicafree" está, portanto, preparado para se tornar um ponto de encontro valioso entre a oferta e a demanda por serviços freelance.

## 16. Referências bibliográficas

GLOBADEV. *Curso de JavaScript para iniciantes*. [recurso eletrônico]. YouTube, 2021. Disponível em: <https://youtu.be/nPEpaft1y1k?si=oowiVYgnNvms9Fzw>.

FILIFE DESCHAMPS. *Curso de HTML e CSS: estrutura básica e boas práticas*. [recurso eletrônico]. YouTube, 2020. Disponível em: <https://youtu.be/McKNP3g6VBA?si=SQ9zMRZ8l1ycwmsh>.

PROGRAMADOR BR. *Curso de JavaScript: conceitos e fundamentos essenciais*. [recurso eletrônico]. YouTube, 2019. Disponível em: <https://youtu.be/w1J6gY40yMo?si=Qo27XGQyKaF5RftL>.

ORACLE. *Java Documentation*. Oracle, 2025. Disponível em: <https://docs.oracle.com/en/java/>

DIO – Digital Innovation One. *Bootcamps e Cursos de Java*. Disponível em: <https://www.dio.me/>. Acesso em: 17 set. 2025.

DEITEL, H. M.; DEITEL, P. J. *Java: como programar*. 10. ed. São Paulo: Pearson, 2016.