**Curso:** Tecnologia em Análise e Desenvolvimento de Sistemas (TDS)

**Equipe 3:** Fabricio, Kahuan, Fernando, Lucas, Giovana

MARINGÁ

2024

**Índice Analítico**

1. Introdução

2. Revisão de literatura

3. Metodologia

4. Funcionalidades

5. Arquitetura do sistema

6. Desenvolvimento

7. Testes

**1. Introdução**

Este projeto tem como objetivo desenvolver uma plataforma inovadora que conecte freelancers e empresas, facilitando o encontro de oportunidades de trabalho e permitindo que ambos os lados beneficiem-se de uma interface simples e eficaz. O aumento do trabalho remoto e das oportunidades para freelancers, principalmente em áreas tecnológicas e criativas, demanda soluções mais completas e eficientes para intermediar essas relações. Nosso sistema abordará não apenas o cadastro de freelancers e empresas, mas também oferecerá ferramentas de comunicação, avaliação e transações seguras entre as partes. Este documento detalha cada etapa do desenvolvimento, desde a concepção da ideia até a implementação final do projeto.

**2. Revisão de Literatura**

O mercado de freelancing tem crescido exponencialmente nos últimos anos, com plataformas como Upwork, Freelancer e Fiverr se destacando mundialmente. No entanto, essas plataformas enfrentam desafios como altas taxas e funcionalidades limitadas que nem sempre atendem às demandas de todos os usuários. Nesta seção, vamos explorar esses sistemas, suas principais características e limitações. Além disso, analisaremos como a nossa plataforma se diferencia, propondo uma solução focada em usabilidade, segurança e personalização. Serão abordadas as tendências tecnológicas que moldam o mercado atual, bem como a evolução das plataformas voltadas para freelancers.

**3. Metodologia**

Para o desenvolvimento da plataforma, adotaremos a metodologia ágil, que permite flexibilidade e constantes revisões durante o ciclo de desenvolvimento. A equipe trabalhará em sprints, com entregas semanais de funcionalidades incrementais. As tecnologias principais que serão utilizadas incluem Java com Spring Boot no backend, garantindo robustez e escalabilidade, e um sistema de frontend simples, utilizando HTML, CSS e JavaScript para criar uma interface de fácil utilização. Também utilizaremos APIs externas para integrar sistemas de pagamento e autenticação, e banco de dados relacional para armazenar as informações de usuários e serviços de maneira segura

.

**4. Funcionalidades**

A plataforma será composta por um conjunto de funcionalidades que visam otimizar a experiência tanto dos freelancers quanto das empresas. Entre elas:

* **Cadastro de freelancers e empresas**: Um sistema de registro que permitirá a criação de perfis detalhados.
* **Postagem de trabalhos**: Empresas poderão publicar oportunidades, especificando os requisitos e prazos.
* **Busca de trabalhos com filtros**: Freelancers poderão buscar projetos utilizando filtros como categoria, prazo e remuneração.
* **Sistema de mensagens**: Comunicação direta entre empresas e freelancers para alinhar detalhes do projeto.
* **Avaliação e feedback**: Após a conclusão de cada trabalho, ambas as partes poderão deixar avaliações que ajudarão a construir a reputação dos usuários na plataforma.

**5. Arquitetura do Sistema**

A arquitetura do sistema será monolítica, facilitando a manutenção e integração entre os componentes. O frontend será desenvolvido com tecnologias simples como HTML, CSS, JavaScript e o framework Bootstrap, enquanto o backend utilizará Java com Spring Boot, gerenciando toda a lógica de negócios e a interação com o banco de dados. O banco de dados será implementado em MySQL, garantindo persistência e eficiência no armazenamento dos dados. A aplicação será hospedada em nuvem, utilizando serviços simples para garantir alta disponibilidade.

**6. Desenvolvimento**

O desenvolvimento será dividido em três grandes fases: planejamento, implementação e testes. No planejamento, foram definidos os requisitos funcionais e as tecnologias a serem utilizadas. Na fase de implementação, cada módulo será desenvolvido de forma integrada, com foco na simplicidade e usabilidade. Durante o processo, enfrentamos desafios como a integração com sistemas de pagamento, que exigiu uma cuidadosa gestão de segurança. A implementação do sistema de comunicação entre freelancers e empresas também foi um ponto crítico para garantir a eficiência nas interações.

**7. Testes**

Serão realizados testes unitários e de integração para garantir que todas as funcionalidades estejam operando corretamente. Cada funcionalidade será testada individualmente para verificar seu comportamento isolado, e também em conjunto para garantir a integração entre os diferentes módulos. Testes de usabilidade serão conduzidos para validar a experiência do usuário, além de testes de desempenho para avaliar a escalabilidade do sistema em cenários de alta demanda. Todos os resultados serão documentados e corrigidos conforme necessário.

**Requisitos Funcionais:**

(RF01) Cadastrar de Usuário

(RF02) Login e Autenticação

(RF03) Publicar Serviços

(RF04) Enviar Currículo para Serviços

(RF05) Avaliar Freelancer e Serviços

(RF06) Criar de Ordens de Serviço

(RF07) Visualizar Histórico de Serviços

(RF08) Processamento de Pagamentos

(RF09) Gerenciar Currículos

(RF10) Filtrar Serviços por Categoria

(RF11) Validar Contratos

(RF12)Gerenciar Perfis

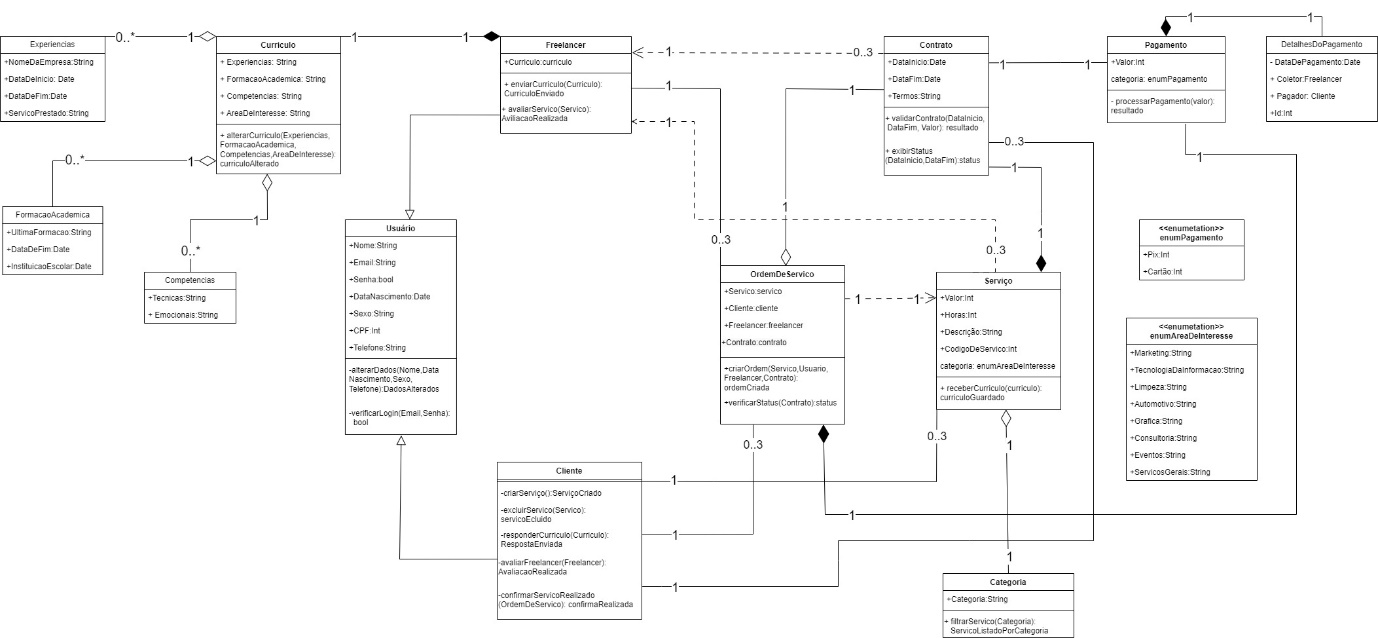
**Requisitos Não Funcionais:**

(RNF 01) Segurança e Criptografia

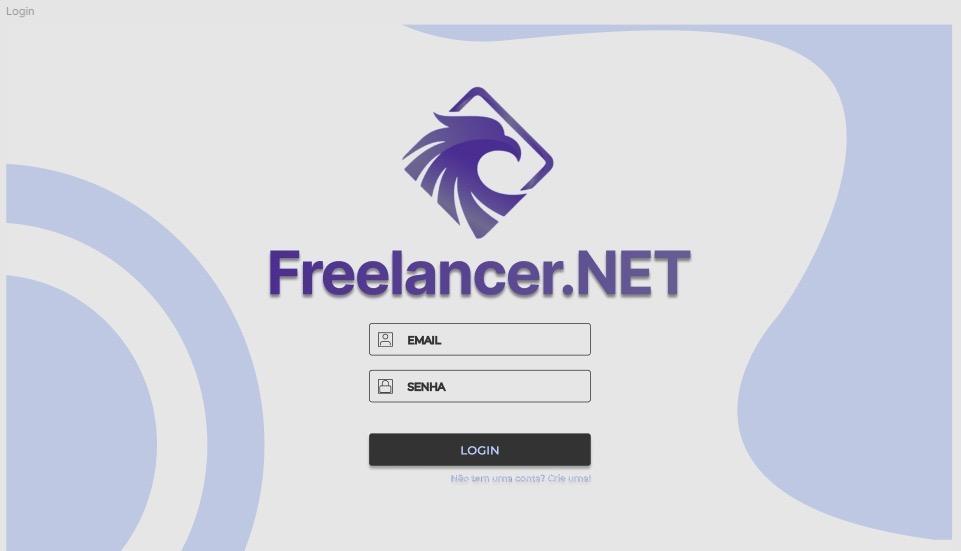
(RNF02) Compatibilidade com Múltiplos Dispositivos

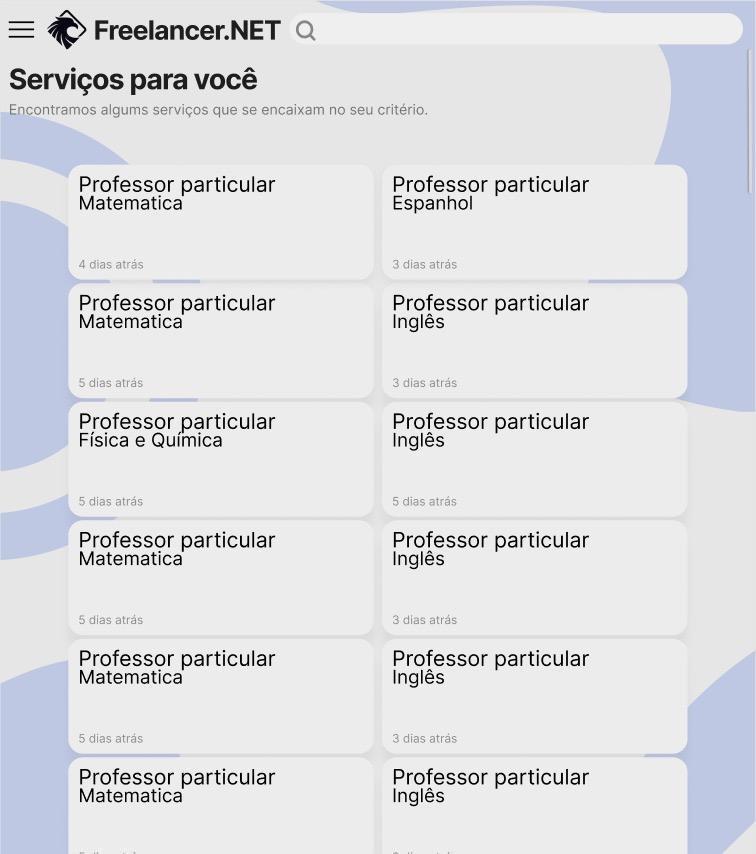
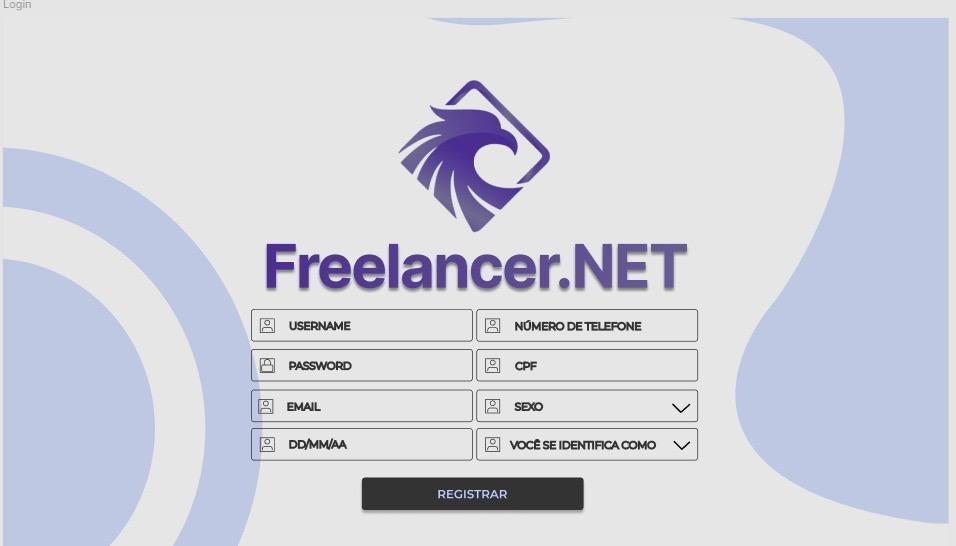
(RNF03) Manutenibilidade

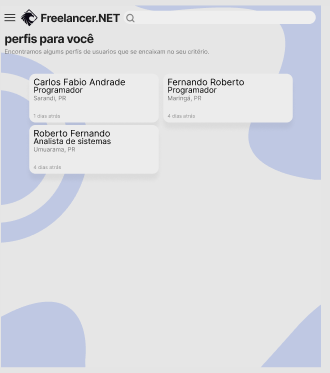
***Diagrama de classes:***



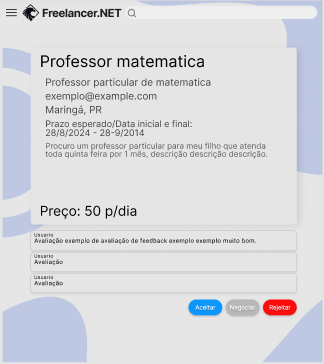
***Prototipagem:***

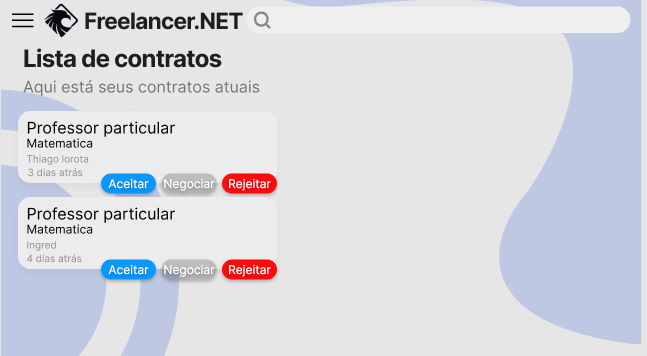


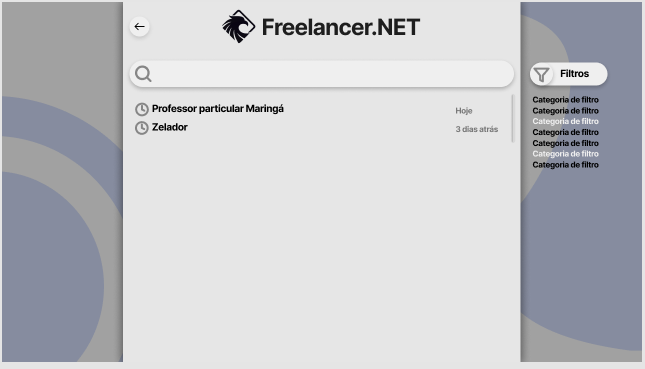


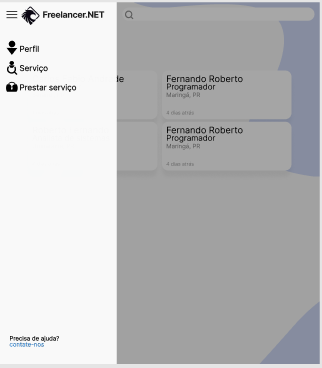


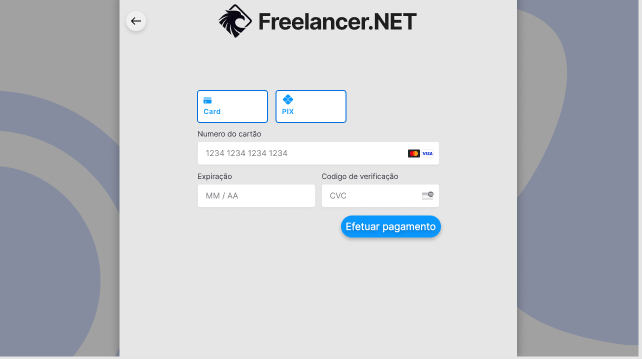


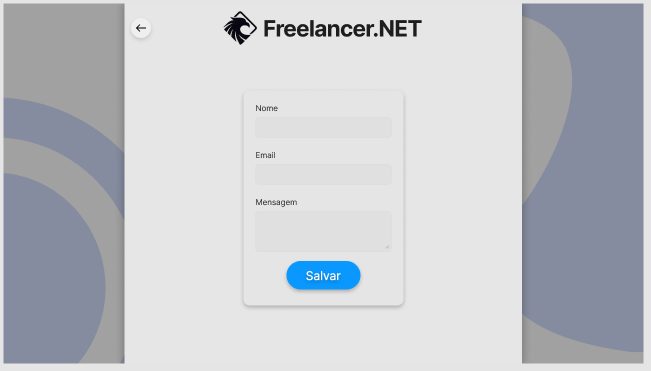


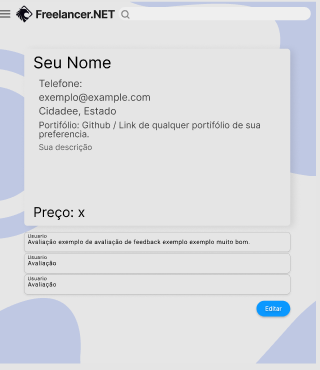


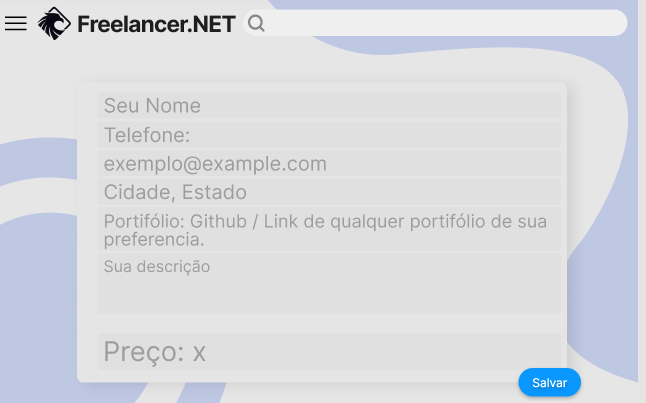


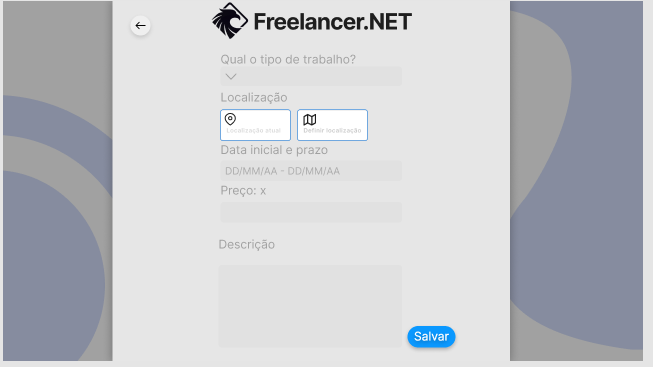




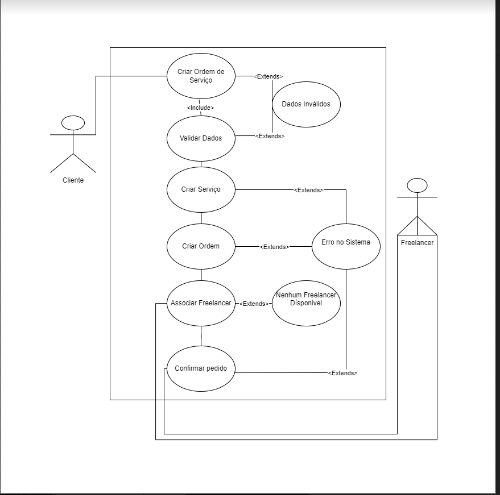








**Caso de uso:**



**Especificação do caso de uso:**

**Especificação do Caso de Uso: FreelancerNet**

**Descrição:**

Esse caso de uso descreve o processo pelo qual um cliente solicita a criação de uma ordem de serviço em um sistema, incluindo a validação de dados, criação de serviço, associação de um freelancer, e confirmação da ordem. O fluxo também prevê possíveis erros, como dados inválidos ou indisponibilidade de freelancers.

**Atores Principais:**

**Cliente:** Solicita a criação da ordem de serviço.

**Freelancer:** Associado ao pedido de serviço.

**Fluxo Principal:**

O Cliente acessa o sistema e inicia o processo para Criar Ordem de Serviço.

O sistema executa a operação de Validar Dados, verificando se as informações fornecidas (detalhes do serviço, prazo, etc.) são válidas.

Se os dados forem válidos, o sistema prossegue para Criar Serviço, instanciando os parâmetros do serviço solicitado pelo cliente.

Após isso, o sistema tenta Criar Ordem, vinculando o serviço aos dados do cliente.

Em seguida, o sistema tenta Associar Freelancer ao serviço.

Finalmente, o sistema Confirma o Pedido e notifica o cliente sobre a conclusão do processo.

**Fluxo Alternativo/Exceções:**

Se houver Dados Inválidos durante a validação, o sistema retornará um erro e solicitará correções ao cliente antes de prosseguir.

Caso haja uma falha no processo de Criar Ordem (por exemplo, Erro no Sistema), o cliente será notificado e a ordem não será gerada.

Se nenhum freelancer estiver disponível no momento da associação, o sistema informará que Nenhum Freelancer Disponível, e o cliente terá que esperar ou tentar novamente mais tarde.

**Pré-condições:**

O cliente deve estar autenticado no sistema.

Todos os dados inseridos pelo cliente devem ser válidos e completos**.**

**Pós-condições:**

A ordem de serviço é criada com sucesso e um freelancer é associado.

O cliente recebe uma confirmação da criação da ordem de serviço.

**Exceções:**

**Dados inválidos:** O sistema notifica o cliente sobre a necessidade de correção dos dados.

**Erro no sistema:** Caso ocorra um erro no processo de criação da ordem, o cliente é notificado e o processo é interrompido.

**Falta de freelancers disponíveis:** Se não houver freelancers disponíveis, o cliente será informado, e o processo poderá ser refeito posteriormente**.**