Wedfy – Plataforma Web de *Freelancing* para Programadores

Luan E. de Queiroz[1], **André C. da Silva**[1]

[1]Grupo de Pesquisa Mobilidade e Novas Tecnologias de Interação
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de Ciência e Tecnologia de São Paulo (IFSP)
Avenida Thereza Ana Cecon Breda, N.º 1896,
Vila São Pedro – 13183-250 – Hortolândia – SP – Brasil

luan.q@aluno.ifsp.edu.br, andre.constantino@ifsp.edu.br

Abstract. *Despite the strong impact of the pandemic on the job market, which caused a peak in unemployment in the first half of 2021, the technology area was one of those that continued to grow and attract people, both those who are still starting their careers and those who are migrating from other areas. This project proposes the elaboration of a first version of a web application aimed at helping self-employed workers in the programming area to develop themselves technically and gain experience working on projects available on the platform. This project was developed in stages and for the elaboration of the prototype the agile Scrum methodology was chosen.*

Resumo. *Apesar do forte impacto da pandemia no mercado de trabalho que causou um pico de desemprego no primeiro semestre de 2021, a área de tecnologia foi uma das que continuou crescendo e atraindo pessoas, seja aquelas que ainda estão iniciando carreira quanto as que estão migrando de outras áreas. Este projeto propõe a elaboração da primeira versão de uma aplicação web voltada para ajudar os trabalhadores autônomos da área de programação a se desenvolver tecnicamente e conseguir experiência trabalhando em projetos disponibilizados na plataforma. Este projeto foi desenvolvido em etapas e para a elaboração do protótipo foi escolhida a metodologia ágil Scrum.*

1. Introdução

A pandemia de COVID-19 nos anos de 2020 a 2022 impactou a nossa realidade de diversas formas, e com o mercado de trabalho não seria diferente. Após o pico de desemprego que chegou durante o primeiro semestre de 2021 [Zanobia 2021], boa parte dos trabalhadores decidiram aderir ao modelo de trabalho autônomo no mesmo ano como uma oportunidade de renda complementar ou principal [PraCarreiras 2021].

Ao mesmo tempo temos o caso da área de tecnologia que, mesmo nesse período, continuou sendo uma das que mais atrai pessoas no mercado de trabalho, sejam os profissionais que ainda estão começando a sua carreira quanto aqueles que já possuem carreira em outras áreas [Albuquerque 2022]. Segundo o jornal Folha de S.Paulo, após a pandemia aquecer o setor de tecnologia, só no primeiro semestre de 2021 foram geradas 106.571 novas vagas [Yuri 2021].

Uma dificuldade que desenvolvedores autônomos podem encontrar é conseguir trabalhos para realizar na hora de entrar em contato com empresas e clientes para oferecer os seus serviços. Também é possível encontrar o cenário onde temos pessoas perdidas em meio às opções de tecnologias, ferramentas e metodologias sem saber quais deve aprender para chegar ao cargo que deseja por falta de direcionamento.

O objetivo deste trabalho é desenvolver a primeira versão de uma aplicação web de *freelancing* onde desenvolvedores podem ter acesso a trabalhos para ganhar experiência profissional e receber algum direcionamento por meio de recomendações de notícias da área dependendo do nicho que ele deseja adquirir mais conhecimento.

Neste artigo serão abordados diferentes tópicos relacionados ao desenvolvimento do projeto, desde a etapa de pesquisas sobre o tema até a análise dos requisitos e desenvolvimento do protótipo executável. Inicialmente serão apresentados na Seção Referencial Teórico alguns termos e conceitos importantes para o desenvolvimento do trabalho (Seção 2), em seguida serão mostrados alguns trabalhos relacionados ao tema de *freelancing* (Seção 3), a metodologia escolhida (Seção 4), como foi o desenvolvimento do projeto (Seção 5) e por fim o artigo se encerra com as conclusões (Seção 6).

2. Referencial Teórico

2.1. Engenharia Web

Engenharia Web é o processo utilizado para o desenvolvimento de aplicações web de alta qualidade, compartilhando muitos conceitos, princípios e métodos com a Engenharia de Software, mas incorporando em si diversas metodologias, ferramentas e técnicas para se adaptar às necessidades específicas das aplicações web [Breve 2002].

Essa engenharia também propõe uma estrutura de processo ágil de forma organizada para garantir a qualidade industrial dessas aplicações [Beder 2017] *apud* [Lowe and Pressman 2009].

2.2. Aplicação Web

Aplicação web é um software que é executado em um navegador da web, acessa serviços e troca informações de forma remota. Também permite acesso a funcionalidades complexas sem instalação ou configuração de softwares [Amazon 2023].

Esse tipo de aplicação possui algumas características e necessidades que, considerados em sua totalidade, o tornam bem diferente de sistemas tradicionais baseados em computador. Algumas dessas características que se encontram na maioria das aplicações web são [Lowe and Pressman 2009]:

Concorrência: Um grande volume de usuários podem acessar a aplicação de forma simultânea, e as vezes as ações de um usuário podem impactar nas ações e informações apresentadas para outros usuários.

Carga imprevisível: O volume de usuários que acessam a aplicação pode variar de um dia para o outro ou do horário.

Desempenho: Se a aplicação web demorar demais para fazer requisições para o servidor, permitir acesso do usuário ou realizar a exibição das informações no lado do cliente é possível que ele desista de usar a aplicação.

Disponibilidade: Normalmente é exigida de aplicações web uma disponibilidade de 100%, funcionando 24 horas por dia, 7 dias por semana, 365 dias por ano.

Orientada a dados: Uma das principais funções das aplicações web é trabalhar com hipermídia como textos, imagens, áudios, vídeos e gráficos, além de normalmente serem utilizadas para fazer acessar informações armazenadas em bancos de dados do lado do servidor.

Sensível ao conteúdo: A qualidade e estética de como a informação é apresentada ao usuário é um dos principais fatores utilizados para avaliar a qualidade de uma aplicação web.

Evolução contínua: Diferentemente dos *softwares* tradicionais, que normalmente recebem atualizações planejadas e cronologicamente espaçadas, as aplicações web podem receber atualizações de conteúdo em poucos minutos dependendo do serviço.

Segurança: Limitar a população de usuários finais de uma aplicação web é algo impossível por estarem disponíveis por acesso à rede.

Outra definição importante para a compreensão desse tipo de aplicação e como ela funciona é a arquitetura cliente servidor, um modelo de arquitetura amplamente utilizado em ambientes de TI [Net 2023]. Nessa abordagem os computadores são divididos em dois grupos: os servidores, aqueles que fornecem serviços e recursos, e os clientes, aqueles que acessam os recursos e serviços usando requisições. Nessa arquitetura os clientes acessam o *front-end*, ou lado cliente, das aplicações e de forma remota se comunicam com o *back-end* presente nos servidores, a parte da aplicação responsável pelas decisões lógicas e iterações com o banco de dados [Machado 2021].

2.3. Metodologia Ágil

São metodologias de desenvolvimento que respeitam os princípios do Manifesto para Desenvolvimento Ágil de Software, criado em 2001 por um grupo de renomados engenheiros de software conhecidos como a Aliança Ágil [Beck et al. 2001].

Elas contam com abordagens iterativas e incrementais para a especificação e desenvolvimento de softwares com o objetivo de apoiar o desenvolvimento de aplicações de negócios onde os requisitos mudam rapidamente [Beder 2017]. Um exemplo de Metodologia Ágil é o Scrum, que será detalhado na próxima seção.

2.4. Scrum

Criado por Jeff Sutherland e Ken Schwaber, o Scrum é um *framework* ágil que busca ajudar equipes e organizações a gerar valor através de soluções adaptativas para problemas complexos. Nessa metodologia a equipe de pessoas responsável pelas atividades relacionadas ao produto é chamada de Time Scrum, composta por um *Product Owner*, um *Scrum Master* e os *Developers* [Sutherland and Schwaber 2020].

O *Product Owner* é o responsável por maximizar o valor do produto resultante do trabalho do Time Scrum e realizar a gestão do *Product Backlog*, uma lista ordenada dos ajustes necessários para melhorar o produto. O *Scrum Master* é o responsável pela implementação do Scrum de forma correta e manter a eficácia do Time Scrum. E os *Developers* são os responsáveis por criar qualquer incremento utilizável em cada *Sprint*.

Na Figura 1 podemos ver como o processo Scrum funciona, seus ritos e artefatos. O primeiro rito de uma *Sprint* é a *Sprint Planning*, uma reunião onde a equipe Scrum escolhe quais itens do *Product Backlog* serão desenvolvidos ao longo da *Sprint* para desenvolver incrementos utilizáveis, gerando o *Sprint Backlog*.

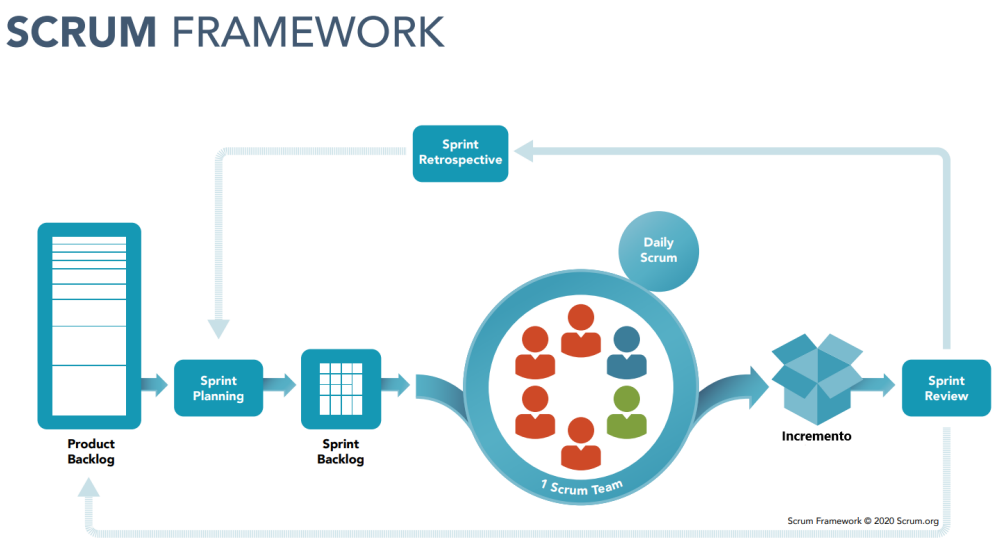

Figura 1. Processo Scrum

Diariamente durante a *Sprint* é realizada a *Daily Scrum*, uma reunião curta onde os *Developers* descrevem o andamento das suas atividades e se há impedimentos para seu desenvolvimento.

No final da *Sprint* ocorre a *Sprint Review*, onde a equipe se reúne para revisar o trabalho concluído e apresentá-lo às partes interessadas, e a *Sprint Retrospective*, onde a equipe Scrum discute sobre os pontos positivos e negativos da *Sprint* e assim gerar ideias para melhorar as próximas *Sprints*.

3. Trabalho Correlatos

Nesta seção são detalhados trabalhos relacionados a *freelance* que foram selecionados como trabalhos correlatos, sendo eles: FindProj, Freelancer e 99Freelas.

3.1. FindProj – Encontre o projeto certo para você

Produzido pela Érika Pereira Ferraz, Lucas Rocha e Rafael Tanaka [Ferraz et al. 2014], este é um Trabalho de Conclusão de Curso desenvolvido em 2014 na Universidade Federal do Paraná para o curso de Tecnologia em Análise e Desenvolvimento de Sistemas.

O objetivo do projeto foi criar um sistema onde as empresas podem ofertar vagas de trabalho e os *freelancers* podem publicar seus currículos, facilitando a comunicação entre eles. Desta forma ajudando o profissional autônomo na busca por serviços e a procura da empresa por mão de obra.

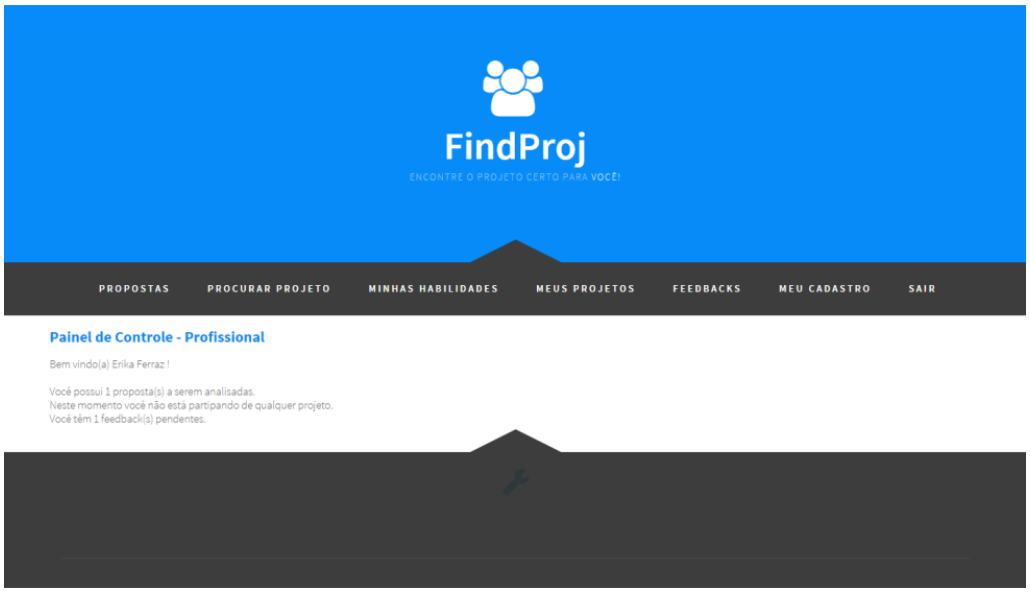

Figura 2. FindProj - tela de opções do profissional

3.2. Freelancer

Freelancer é uma plataforma de *freelancing* que atua no mercado desde 2012 trabalhando com profissionais de diferentes áreas de atuação e projetos de diferentes categorias. Nessa plataforma os profissionais podem, além de se candidatar para trabalhar com os projetos, indicar outros trabalhadores para projetos e criar redes para que possam se comunicar e colaborar. A plataforma também permite a troca de *feedbacks* entre trabalhador e cliente do projeto para que outros usuários possam consultar.

Figura 3. Freelancer - página inicial

3.3. 99Freelas

99Freelas é uma plataforma online brasileira fundada em 2014 por Wilker Iceri, ela reúne trabalhadores autônomos e projetos de diversas áreas. Nela são publicadas vagas para

tarefas específicas onde os *freelancers* podem se inscrever e ser avaliados por histórico de trabalho, *feedbacks* de clientes anteriores, portfólio e posição no sistema de ranqueamento do próprio site.

Figura 4. 99Freelas - página inicial

3.4. Análise comparativa dos trabalhos correlatos

Com base nas informações adquiridas nos trabalhos correlatos apresentados anteriormente, é possível realizar uma análise comparativas das funcionalidades desses trabalhos com o projeto a ser desenvolvido (Tabela 1). Logo de início podemos perceber que todas as aplicações tem como principal função buscar ajudar na comunicação entre trabalhador autônomo e contratante, permitindo cadastro de currículo, troca de *feedbacks* entre usuários e cadastro de habilidades dos trabalhadores. Também é possível notar que a maior parte dos projetos permitem o cadastro do portfólio dos profissionais, visualização de perfis e a possibilidade de aplicar filtros de pesquisa para projetos ou profissionais.

As aplicações de mercado são as únicas que permitem a criação de rede de comunicação entre usuários na plataforma e indicação de profissionais para projetos. O uso de todas as plataformas é gratuito, porém as aplicações Freelancer e 99Freelas disponibilizam assinaturas com recursos extras e removendo algumas limitações, como a quantidade de projetos que pode trabalhar simultaneamente.

Algumas das características encontradas em aplicações web citadas da Seção 2 que podemos identificar nos trabalhos correlatos são a disponibilidade, páginas orientadas a dados e sensíveis ao conteúdo, adaptáveis a evolução contínua além de entregarem um bom desempenho.

4. Metodologia

O Scrum foi adaptado para o desenvolvimento deste projeto de forma que eu estarei responsável pela gestão do *Product Backlog* e *Sprint Backlog*, além de desenvolver os incrementos utilizáveis das *Sprints* atuando como a Equipe Scrum. As reuniões de *Sprint Review* também foram adaptadas, elas foram utilizadas para repassar ao professor orientador

Tabela 1. Comparação das principais funcionalidades dos trabalhos correlatos

	FindProj	Freelancer	99Freelas	Wedfy
Oferece uma forma de ajudar na construção da base técnica do usuário				X
Ajuda na comunicação profissional x contratante	X	X	X	X
Cadastro de portfólio		X	X	X
Cadastro de currículo	X	X	X	X
Criar rede de profissionais		X		
Fazer recomendação de profissionais		X	X	
Enviar feedbacks	X	X	X	X
Visualizar perfis de outros usuários	X	X	X	X
Cadastrar habilidades	X	X	X	X
Aplicar filtros na pesquisa de projetos ou profissionais		X	X	X

os incrementos utilizáveis gerados ao longo das *Sprints* e como está o desenvolvimento do projeto.

A metodologia escolhida para o desenvolvimento do projeto foi o processo de desenvolvimento de software constituído por quatro etapas básicas: fase de diagnóstico, levantamento e análise de requisitos, fase de desenvolvimento e etapa de implantação [Júnior 2021].

Fase de diagnóstico: Nessa etapa o time de desenvolvimento foca em conhecer o cliente a fundo, detalhar e explicar o problema em questão para que o software atenda a todas as necessidades. No caso desse projeto, a fase de diagnóstico será a etapa destinada às pesquisas mais aprofundadas sobre o tema e outras plataformas de *freelance*.

Levantamento e análise de requisitos: Nesta etapa começamos a pensar em alternativas de soluções para o problema baseadas no diagnóstico feito na etapa anterior. Após o levantamento é feita a análise, onde é feita a definição das atividades e estipulação dos prazos, por fim é feita uma breve validação, onde é avaliada a eficiência e relevância dos requisitos levantados. Para o desenvolvimento do projeto proposto essa etapa será utilizada para levantar os requisitos da aplicação e elaborar diagramas que irão auxiliar a etapa de desenvolvimento.

Fase de desenvolvimento: É a etapa onde o código da aplicação começa a ser de fato desenvolvido com base nas atividades definidas na etapa anterior. Essa etapa será utilizada para realizar o desenvolvimento do protótipo executável da aplicação proposta.

Etapa de implantação: Entrega do projeto e onde são requisitados o *feedback* do usuário para melhorar o software. No caso desse projeto a etapa de implantação foi adaptada para ser a etapa onde o projeto é apresentado para o professor orientador e a banca, realizando a coleta dos *feedbacks* para realizar os ajustes finais no projeto e artigo.

O *front-end* da aplicação será desenvolvido utilizando o *framework* React e C# no *back-end*, enquanto será adotado o sistema gerenciador de banco de dados MySQL.

A escolha do React para o *front-end* se dá por ser um dos principais *frameworks* usados no momento, na frente do Vue e Angular, além de possuir diversas bibliotecas que auxiliam no desenvolvimento, outra vantagem é que o React permite que os trechos de código sejam separados e utilizados como componentes reutilizáveis.

O C# foi escolhido como tecnologia do *back-end* por já ser uma linguagem madura com diversas bibliotecas para auxiliar no desenvolvimento, além de ser a linguagem mais usada para programar com o *framework* ASP.NET Core, o segundo maior *framework back-end* na internet.

A alta performance, alta disponibilidade e outras diversas funcionalidades sendo gratuito foram os fatores que tornaram o MySQL ser o escolhido para o desenvolvimento deste projeto. Outro fator que também foi levado em conta foram os conhecimentos que já tenho sobre essas tecnologias e que gostaria de aprimorar durante o desenvolvimento.

5. Desenvolvimento do Trabalho

5.1. Fase de diagnóstico

A primeira etapa do desenvolvimento do trabalho foi voltada para pesquisas exploratórias com base em plataformas de *freelance* já existentes, buscando analisar como elas fornecem aos usuários a opção de procurar por projetos e classificá-los, além de como é trabalhada a apresentação dos perfis dos profissionais autônomos cadastrados e suas capacidades técnicas.

Durante essa primeira etapa foi possível identificar ao menos dois atores para o sistema, sendo eles o cliente que anuncia trabalhos na plataforma e o programador que busca trabalhos para desenvolver. Pensando no contexto do projeto proposto também foi identificado um terceiro ator, o administrador do sistema que terá a função de cadastrar novas recomendações de notícias para os programadores autônomos terem acesso.

5.2. Levantamento e análise de requisitos

Com base nas informações adquiridas durante a etapa anterior, foi elaborado um diagrama de casos de uso para organizar os requisitos funcionais do projeto pensando nos atores Programador, Cliente e Administrador, como pode ser visualizado na Figura 5. Ao longo desta etapa foi realizado o acompanhamento e validação dos diagramas desenvolvidos com o professor orientador como *Product Owner*.

Inicialmente foi criado um ator Usuário com casos de uso relacionados a login, cadastro de perfil, visualização de perfil e interação com *feedbacks* que irá passar essas funções básicas do sistema para os outros atores por meio de herança.

O Programador é o ator que se relaciona com mais casos de uso, sendo capaz de fazer a pesquisa de projetos, aplicar filtros de pesquisa, ver detalhes de projetos, demonstrar interesse neles e poder cadastrar seu portfólio para aparecer no seu perfil. Esse ator também tem acesso a uma página com recomendações de notícias sobre a área de tecnologia da informação.

O ator Cliente é capaz de fazer a administração dos seus projetos, podendo publicar, excluir e finalizar projetos. Também pode aceitar ou recusar as solicitações de interesse de Programadores relacionadas aos seus projetos, notificando eles sobre a sua decisão.

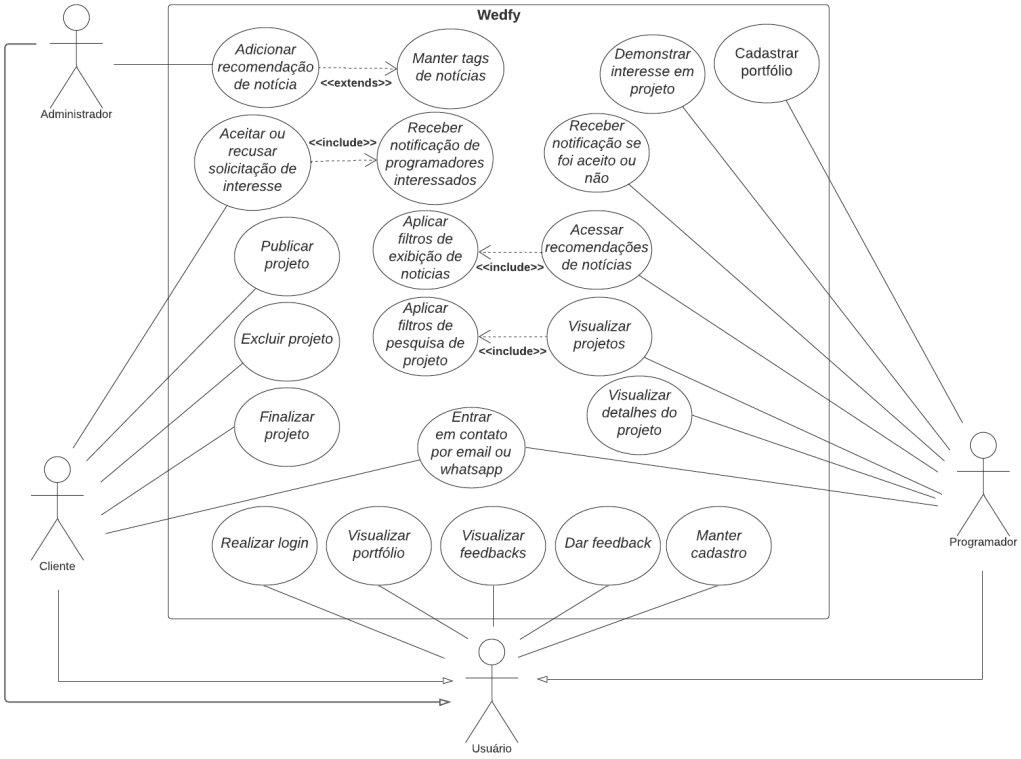

Figura 5. Diagrama de caso de uso para a aplicação Wedfy

Por fim, o Administrador é o responsável por fazer a inclusão de recomendações de notícias que o Programador será capaz de acessar posteriormente, podendo adicionar *tags* para que seja possível filtrar as notícias por categorias. O caso de uso relacionado a cadastro de conta funciona de forma diferente para o Administrador, de forma que ele não seja capaz de realizar o cadastro pela aplicação, somente de forma direta pelo banco de dados. O Cliente e o Programador também podem entrar em contato entre si por WhatsApp ou e-mail através de atalhos em seus perfis para que possam conversar sobre o projeto a ser desenvolvido.

O diagrama referente a Figura 6 tem como objetivo ilustrar as entidades e relacionamentos para definir como as tabelas do banco de dados do projeto vão se relacionar. Desenvolvido pensando nos requisitos funcionais apontados no diagrama de caso de uso (Figura 5), o modelo final conta com uma tabela para cada objeto diferente identificado no projeto, com exceção da tabela "usuario"destinada para o armazenamento das informações pessoais dos três atores do sistema.

5.3. Fase de desenvolvimento

Levando em conta todos os conceitos e requisitos apresentados nas seções anteriores, nesta Subseção são detalhadas as informações sobre o processo de elaboração dos protótipos do projeto e desenvolvimento das suas funcionalidades.

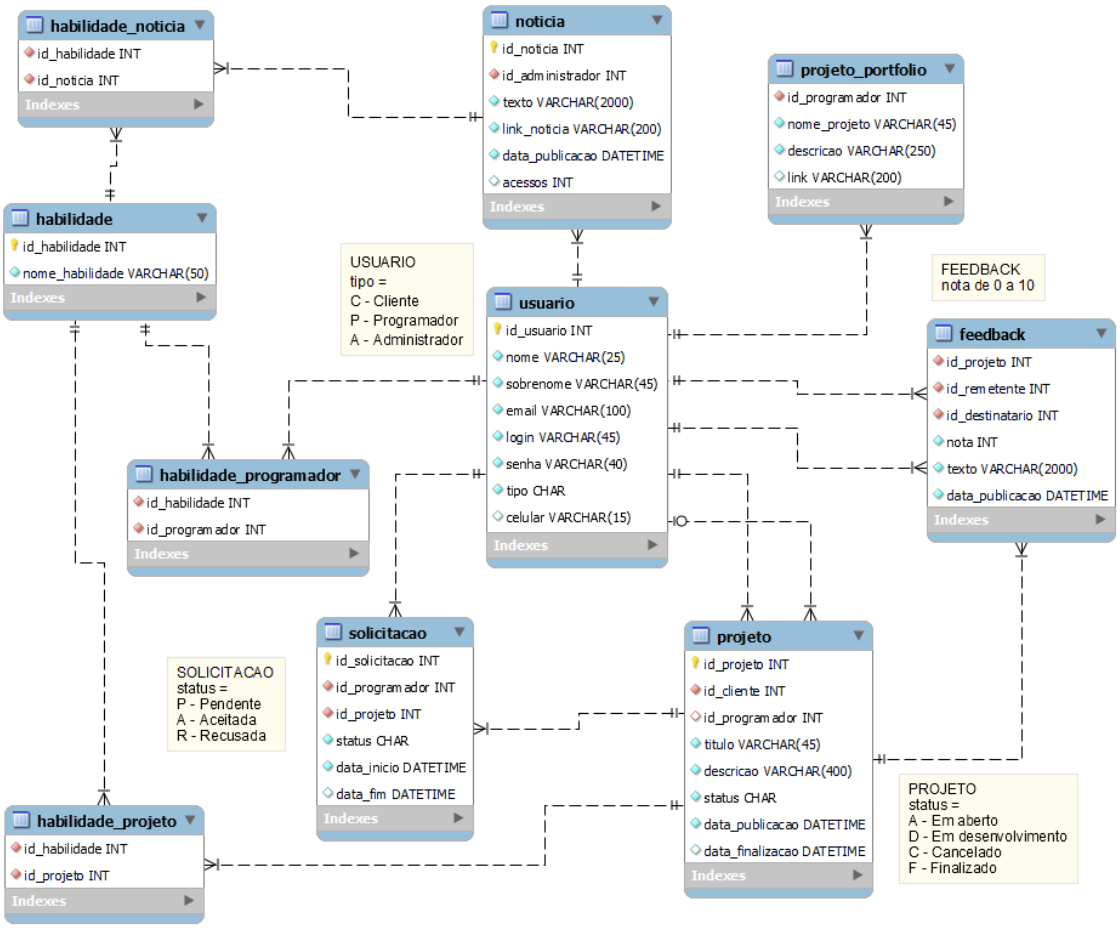

Figura 6. Diagrama lógico para a base de dados do Wedfy

5.3.1. Prototipação das telas

A fase de desenvolvimento foi iniciada com a prototipação das telas da aplicação, buscando estabelecer como serão trabalhadas as principais funcionalidades da aplicação apontadas na Figura 5 e como exibidas aos diferentes atores do sistema. Nessa subseção são apresentadas as principais telas do sistema relacionadas ao fluxo de desenvolvimento dos projetos e acesso as notícias, desenvolvidas ao longo das duas *sprints* do mês de Agosto com validação e acompanhamento do professor orientador como *Product Owner*.

O desenvolvimento dos protótipos das telas foi iniciado pensando na divisão da aplicação em três partes distintas para cada tipo de usuário, principalmente para o Administrador que interage com o sistema de forma bem diferente do Cliente e Programador, tendo como única função realizar o cadastro de novas notícias para os usuários Programadores.

A tela inicial do projeto é uma *landing page* dividida em duas partes, a primeira onde é exibida a logo do site junto de botões que levam aos formulários de cadastro de usuário e de login, e a segunda onde é possível visualizar dois cartões que apresentam as ações que cada usuário é capaz de executar, como demonstrado na Figura 7.

A tela de login exibida na Figura 8 conta com uma barra de menu na parte superior

Figura 7. Protótipo da Landing page

da página que possui a logo do site e um *link* para a *landing page* apresentada anteriormente, além do formulário de login com os campos login e senha, botão de confirmação e outros dois botões que direcionam o usuário para as telas de cadastro de usuário cliente e programador, respectivamente.

As telas de cadastro dos usuários cliente e programador possuem aparências bem semelhantes, ambas contendo um menu na parte superior da página com um *link* que leva para a tela de login, o formulário de cadastro posicionado no lado esquerdo do corpo da página acompanhado de uma imagem relacionada a função escolhida ao lado direito. A estrutura da página de cadastro do cliente é exibida na Figura 9.

Na Figura 10 é apresentada a tela de busca de projetos acessada por um usuário Programador, contando com uma barra de menu na parte superior da página, um campo de texto para auxiliar na busca de projetos, um campo de seleção para aplicar filtros de pesquisa e um espaço destinado para a exibição de cartões com as principais informações

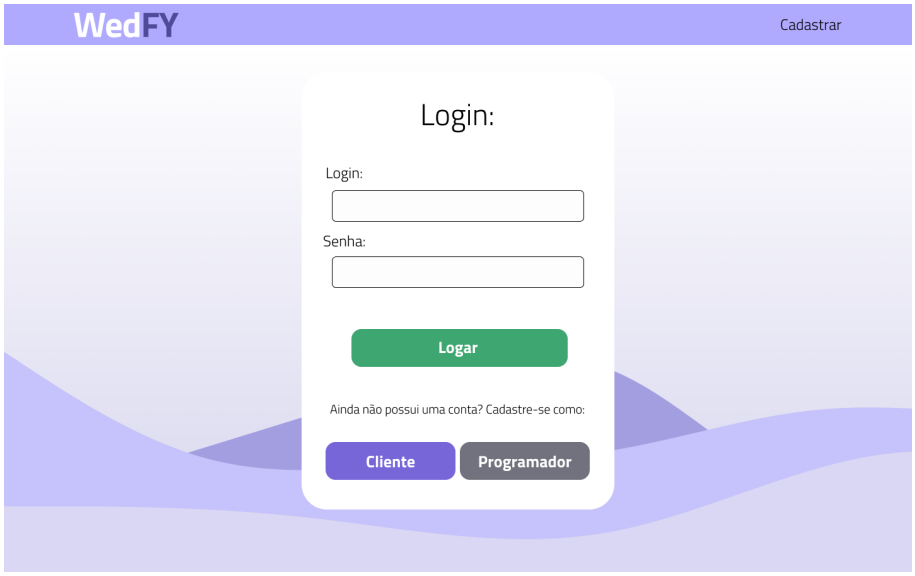

Figura 8. Protótipo da tela de login

dos projetos.

Junto desses cartões também é exibido um botão que permite que o usuário acesse os detalhes do projeto. Esta página também pode ser acessada pelo usuário Cliente, porém sem a opção de acessar a página de notícias no menu superior por se tratar de um recurso único do Programador.

Na Figura 11 é possível ver como foi pensada a página de exibição de detalhes dos projetos no ponto de vista do programador, onde além do nome, dono do projeto, *tags* relacionadas e data de publicação já exibidos anteriormente também é possível visualizar uma breve descrição do projeto, seu status atual e a opção de demonstrar interesse em desenvolvê-lo. Essa página também pode ser acessada pelo usuário cliente dono do projeto, aqui ele pode alterar o *status* de desenvolvimento do projeto ou cancelá-lo.

Dando continuidade ao processo de solicitações de interesse, na Figura 12 é exibida como o usuário Cliente é capaz de visualizar as solicitações recebidas de programadores para cada um de seus projetos publicados na plataforma, também sendo capaz de realizar o aceite ou recusa dessas solicitações além de acessar o perfil dos programadores para consultar possíveis *feedbacks*, projetos anteriores e seu portfólio.

Na Figura 13 é possível visualizar como é exibido o perfil de usuários programadores para os clientes, informando o nome do profissional, suas principais habilidades ou tecnologias que domina, projetos anteriores que ele trabalhou na plataforma, projetos de portfólio que ele tenha cadastrado e *feedbacks* de clientes que trabalharam com ele anteriormente acompanhados de uma avaliação.

Outra tela que é considerada uma das principais do sistema é a de consulta de recomendações de notícias, mantida pelo Administrador acessada somente pelos usuários Programadores. Na Figura 14 é exibido o protótipo da tela de pesquisa de notícias, onde as notícias são exibidas do lado direito da tela enquanto do lado esquerdo estão localizados opções de como o conteúdo deve ser organizado e uma barra de pesquisa, caso o usuário queira buscar um tópico específico.

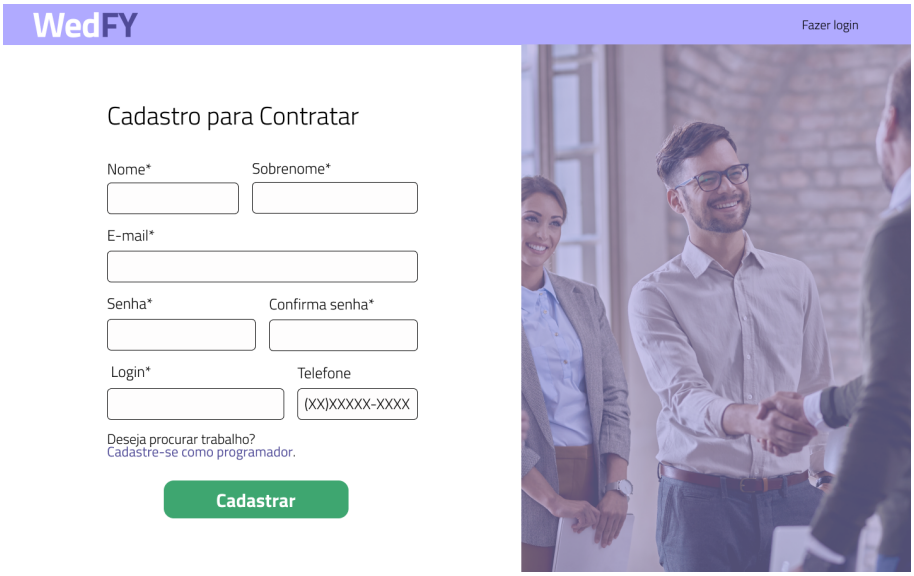

Figura 9. Protótipo da tela de cadastro do usuário cliente

5.3.2. Desenvolvimento do banco de dados

A etapa de desenvolvimento do banco de dados foi possivelmente a mais curta do processo graças ao diagrama lógico de dados desenvolvido anteriormente. Com auxílio da ferramenta de design de banco de dados MySQL Workbench foi possível gerar a estrutura do banco de dados com todas as tabelas e relações necessárias utilizando a opção de "engenharia reversa"a partir do diagrama lógico, onde a ferramenta gera todo o código para a criação da base de dados usando o modelo como referência.

5.3.3. Desenvolvimento do *back-end*

Inicialmente, o *back-end* do projeto foi concebido para ser desenvolvido em uma arquitetura em quatro camadas, a saber: *Controllers*, *Interfaces*, *Models* e *Services*. A camada de *Controllers* seria responsável por conter os arquivos da API com rotas que podem ser utilizadas por outras aplicações para acessá-la e fazer requisições, como a rota de consulta de Notícias apresentada na Figura 15.

A camada de *Interfaces* teria como função conter os arquivos que funcionam como "contratos"para as outras camadas, definindo quais informações elas devem conter e quais funções devem ser capazes de executar. Na camada de *Models* são encontrados os arquivos da aplicação que representam os objetos utilizados durante o funcionamento da aplicação. Esses arquivos contêm os campos que determinam quais informações cada elemento deve armazenar. Como exemplo, a classe Notícia, conforme ilustrada na Figura 16. Por fim, a camada de *Services* seria responsável por armazenar os arquivos com as funções que seriam chamadas pela camada de *Controllers* para se comunicar com o banco de dados e retornar informações.

Devido a atrasos no desenvolvimento da arquitetura planejada inicialmente, foi necessário realizar uma mudança nos planos no final da última *Sprint* destinada ao desen-

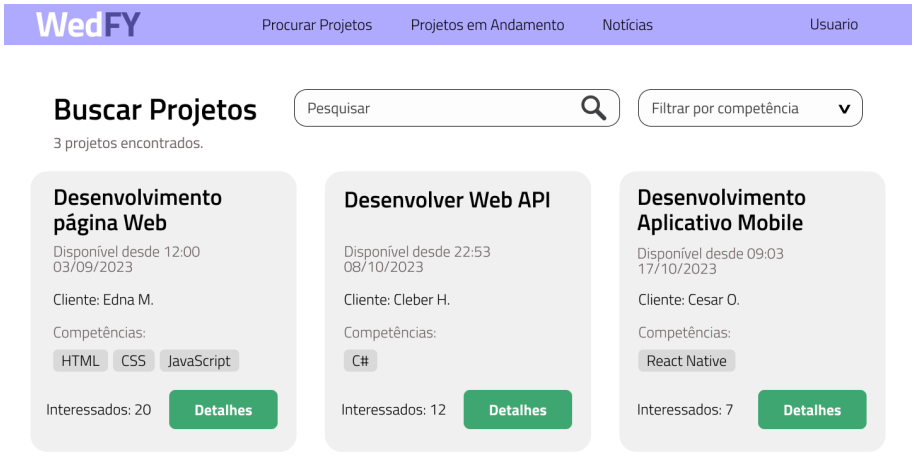

Figura 10. Protótipo da tela de busca de projetos

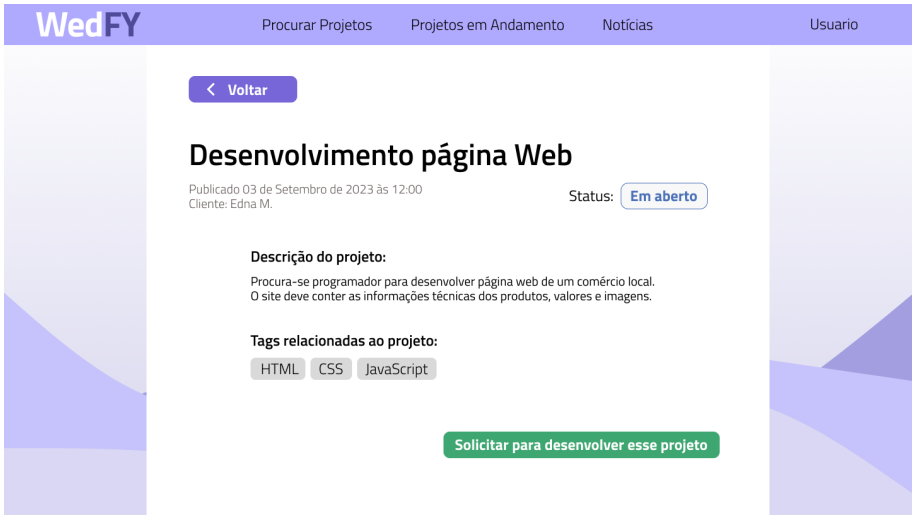

Figura 11. Protótipo da tela de detalhes de projeto

volvimento do *back-end*. Optei por uma abordagem menos complexa, com uma arquitetura de apenas três camadas, desenvolvida com o auxílio da ferramenta *Entity Framework Core*, uma versão mais leve e multiplataforma da tecnologia de acesso a dados *Entity Framework*. Esta tecnologia permite que seja possível trabalhar com um banco de dados usando objetos .NET, eliminando a necessidade de maior parte do código que normalmente seria necessário e sendo compatível com vários mecanismos de bancos de dados [Microsoft 2023].

As três camadas que constituem a API desenvolvida com suporte do *Entity Framework Core* são: *Models*, *Data* e *Controllers*. As camadas *Controllers* e *Models* funcionam da mesma forma descrita anteriormente, enquanto a camada de *Data* é a responsável por realizar a conversão das classes presentes na *Models* em objetos .NET capazes de se comunicar com o banco de dados. Na Figura 17 é possível visualizar como é feita a conversão da classe de Notícia em um objeto .NET chamado de Entidade, associando os campos da classe com os presentes na tabela do banco de dados e aplicando suas propri-

Figura 12. Protótipo da tela de solicitações pendentes

edades.

5.3.4. Desenvolvimento do *front-end*

A implementação do *front-end* da aplicação não obteve o andamento desejado por conta da redução de tempo destinado a ele devido os atrasos no desenvolvimento da API, contudo ainda foi possível realizar a implementação das telas de login e de cadastro de usuário de forma estilizada e capazes de realizar a comunicação com a banco de dados através das rotas do *back-end*, utilizando de recursos de criptografia para realizar envio de informações como as senhas de usuário para a API de forma mais segura.

Como um projeto React padrão, a sua estrutura é constituída por três pastas principais, sendo elas a node_modules, responsável por armazenar informações como bibliotecas ou configurações de ambiente utilizadas no desenvolvimento da aplicação, a *public* que possui arquivos estáticos como a logo, o *index* da aplicação e suas configurações, e por fim a *src* que armazena os arquivos que constituem a aplicação em si como seus componentes, imagens, arquivos de estilização e sistema de rotas. Para obter maior nível de organização, dentro da pasta *src* foram criadas quatro pastas menores para que seja possível realizar a separação dos arquivos da aplicação por tipo, como exibido na Figura 18.

A pasta *components* é responsável por armazenar os arquivos com trechos de código que representam "partes"da página web, contendo tanto a lógica quanto a estrutura da parte que ele representa. Na Figura 19, é exibido o código do componente utilizado como cabeçalho nas telas de cadastro de usuário. Nela, é possível observar que a estrutura básica dos componentes começa com as importações das bibliotecas, imagens e arquivos de estilização necessários para seu funcionamento, seguidas de uma função principal que retorna a estrutura dos elementos do componente e, por fim, a exportação dessa função, permitindo que outros arquivos possam importar e utilizar esse componente posteriormente.

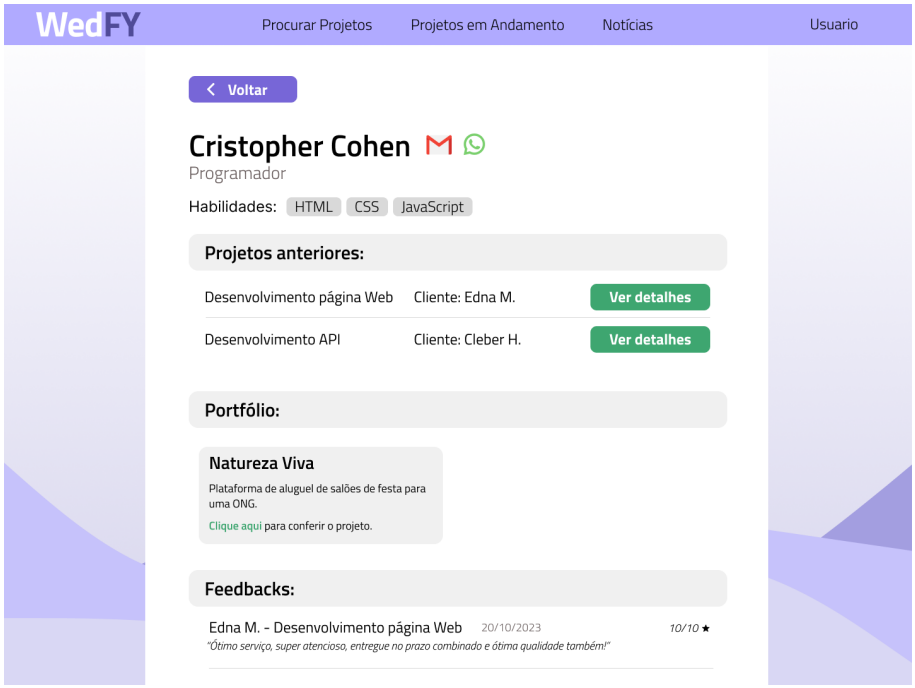

Figura 13. Protótipo da tela de exibição de perfil de usuário programador

Em *images* é onde permanecem armazenados os arquivos em formado de imagem utilizados nos componentes. Na pasta *styles* se encontram os arquivos CSS da aplicação que estilizam os textos, imagens e demais elementos presentes nos componentes, separados em pastas menores dependendo da página que os utiliza. Por fim, a pasta *pages* é a responsável por conter os arquivos que organizam como as páginas devem ser exibidas para o usuário, importando os componentes que a constituem e posicionando-os na sequência correta, como exemplificado na Figura 20 com o código da página de cadastro de usuário.

Para que seja possível realizar a navegação entre diferentes páginas dentro do projeto foi utilizada a biblioteca *react-router-dom*. Esta biblioteca permite que seja possível realizar o mapeamento das rotas da aplicação e vincular estas rotas com páginas diferentes, como apresentado no trecho de código presente na Figura 21, onde após realizar as importações dos objetos *RouterProvider* e *createBrowserRouter* é criado uma variável constante com o nome *router* que possui inicialmente dois caminhos, cada um atrelado a um elemento diferente. Após a criação desse objeto *router* ele é utilizado pelo *RouterProvider* como um parâmetro de rota e é renderizado, dessa forma ele já é capaz de interpretar a mudança das rotas e alterar o conteúdo da página de forma dinâmica.

Para auxiliar na comunicação entre o *front-end* e a API da aplicação foi escolhido o Axios, uma biblioteca que torna mais simples o processo de realizar requisições e receber respostas do servidor. Conforme exemplificado na função utilizada para autenticação do usuário exibida na Figura 22, o Axios recebe a rota que deve ser acessada, o tipo de requisição que é feita e quais informações devem ser fornecidas para o servidor retornar o resultado, logo em seguida já sendo possível realizar o tratamento desse resultado para apresentá-lo de forma visual ao usuário através de alguma mensagem ou redirecionamento para outra página.

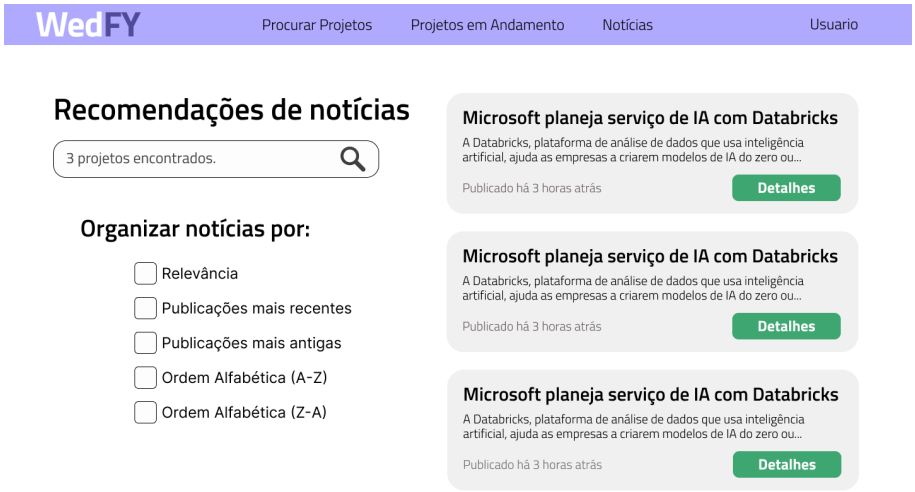

Figura 14. Protótipo da tela de exibição de recomendações de notícias

```
[HttpGet]
0 referências
public async Task<ActionResult<IEnumerable<Noticia>>> GetNoticia()
{
  if (_context.Noticia == null)
  {
      return NotFound();
  }
    return await _context.Noticia.ToListAsync();
}
```

Figura 15. Rota de consulta de notícias

Apesar dos atrasos citados anteriormente, foi possível realizar a implementação das telas de login e cadastro de cliente de forma fiel aos protótipos apresentados na seção de prototipação das telas, como pode ser visto nas Figuras 23 e 24, respectivamente.

6. Conclusão

Como apresentado no início deste artigo, o propósito do presente trabalho foi realizar o desenvolvimento da primeira versão de uma aplicação web de *freelancing* direcionada a programadores autônomos em inicio de carreira para contribuir com sua capacitação e facilitar a comunicação entre eles e possíveis contratantes de seus serviços.

Ao longo do desenvolvimento do projeto foram identificados diversos desafios como o atraso no desenvolvimento do *back-end* da aplicação relacionado a limitações técnicas ou os erros na tela de login, onde algumas vezes ela não conseguia realizar a requisição a API e exibir o resultado esperado durante os testes funcionais. Estes desafios contribuíram para a fixação do conhecimento adquirido durante o curso, além de servir como oportunidade para realizar a busca de aprendizado envolvendo tecnologias de mercado não apresentadas ao longo do curso, como o React para o desenvolvimento de interfaces de usuário em páginas web e o ASP.NET Core para criar aplicativos web conectados à internet.

Algumas das principais disciplinas do curso de Análise e Desenvolvimento de

```
namespace APIWedfy.Models;

10 referências
public partial class Noticia
{
    5 referências
    public int IdNoticia { get; set; }
    3 referências
    public int IdAdministrador { get; set; }
    1 referência
    public string Texto { get; set; } = null!;
    1 referência
    public string LinkNoticia { get; set; } = null!;
    1 referência
    public DateTime DataPublicacao { get; set; }
    1 referência
    public int? Acessos { get; set; }
    1 referência
    public virtual Usuario IdAdministradorNavigation { get; set; } = null!;
}
```

Figura 16. Classe referente ao objeto Notícia

```
modelBuilder.Entity<Noticia>(entity =>
{
    entity.HasKey(e => e.IdNoticia).HasName("PRIMARY");

    entity.ToTable("noticia");

    entity.HasIndex(e => e.IdAdministrador, "fk_noticia_usuario1_idx");

    entity.Property(e => e.IdNoticia).HasColumnName("id_noticia");
    entity.Property(e => e.Acessos).HasColumnName("acessos");
    entity.Property(e => e.DataPublicacao)
        .HasColumnType("datetime")
        .HasColumnName("data_publicacao");
    entity.Property(e => e.IdAdministrador).HasColumnName("id_administrador");
    entity.Property(e => e.LinkNoticia)
        .HasMaxLength(200)
        .HasColumnName("link_noticia");
    entity.Property(e => e.Texto)
        .HasMaxLength(2000)
        .HasColumnName("texto");

    entity.HasOne(d => d.IdAdministradorNavigation).WithMany(p => p.Noticia)
        .HasForeignKey(d => d.IdAdministrador)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("fk_noticia_usuario1");
});
```

Figura 17. Criação do modelo referente a entidade Notícia

Sistemas relacionadas com os conhecimentos articulados para a realização do trabalho incluem Interação Humano-Computador, por conta do cuidado presente na escolha das cores utilizadas nas telas para obter o nível recomendado de contraste e torná-las mais acessíveis; Desenvolvimento Web e Desenvolvimento de Sistemas Web para a implementação da aplicação web utilizando as tecnologias HTML, CSS e JavaScript, integração com banco de dados e comunicação com API externa; Banco de dados I e II para compreensão e elaboração do modelo e base de dados utilizados; Análise Orientada a Objetos, por conta da elaboração do diagrama de caso de uso.

O projeto foi realizado ao longo de dezesseis *Sprints* com duração de duas semanas cada, sendo as oito primeiras *Sprints* destinadas a escrita das seções iniciais do artigo e elaboração dos diagramas e modelos apresentados anteriormente enquanto as restantes permaneceram focadas no desenvolvimento da aplicação, seus protótipos e na documentação deste processo. A metodologia *Scrum* se mostrou mais presente durante o

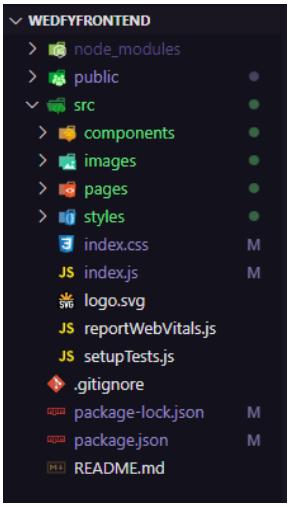

Figura 18. Estrutura de arquivos do *front-end*

```
import { Link } from 'react-router-dom';
import logo from '../../images/WedFY.png';
import '../../styles/register/RegisterHeader.css';

function RegisterHeader() {
    return (
        <header className="RegisterHeader">
            <img src={logo} className='Wedfy-logo' alt='logo Wedfy' />

            <Link to='/'>Fazer Login</Link>
        </header>
    );
}

export default RegisterHeader;
```

Figura 19. Código do cabeçalho das telas de cadastro de usuário

desenvolvimento da aplicação para que fosse possível realizar de forma mais organizada a separação do tempo destinada para a parte prática do projeto, apesar do atraso presente no desenvolvimento do *back-end*.

Para trabalhos futuros certamente poderiam ser implementadas as demais funcionalidades e telas do sistema relacionadas a manipulação de notícias, projetos e *feedbacks* que, infelizmente, não puderam ser desenvolvidas por conta do atraso citado anteriormente. Também devem ser considerados os requisitos não-funcionais que não foram trabalhados ao longo do desenvolvimento, como a sensibilidade ao conteúdo ou formas de lidar com a concorrência de acessos e carga imprevisível de usuários simultâneos.

Outras propostas são o desenvolvimento de mais funcionalidades envolvendo o ator Administrador para que ele possa interagir mais com a plataforma e os outros usuários, obtendo mais ferramentas para desempenhar a sua função monitorando e garantindo o funcionamento da aplicação, mais opções de status para acompanhamento dos projetos, como um estado de "suspenso"para projetos com falta de verba, por exemplo. E, por fim, o desenvolvimento de um *chat* para realizar a comunicação entre cliente e programador, dessa forma não seria necessário recorrer a *e-mails* e números de telefone para que um

```
import RegisterHeader from "../components/register/registerHeader";
import ClientRegisterContent from "../components/register/clientRegisterContent";

function ClientRegister() {
    return (
        <>
            <RegisterHeader />
            <ClientRegisterContent />
        </>
    );
}

export default ClientRegister;
```

Figura 20. Código do cabeçalho das telas de cadastro de usuário

```
import React from 'react';
import './index.css';
import { createRoot } from 'react-dom/client';
import {
  RouterProvider,
  createBrowserRouter
} from "react-router-dom";
import reportWebVitals from './reportWebVitals';

import Login from './pages/Login';
import ClientRegister from './pages/ClientRegister';

const router = createBrowserRouter([
  {
    path: "/",
    element: <Login />
  },
  {
    path: "/cadastroCliente",
    element: <ClientRegister />
  }
])

createRoot(document.getElementById("root")).render(
  <RouterProvider router={router} />
)
```

Figura 21. Código do cabeçalho das telas de cadastro de usuário

possa contatar o outro fora da aplicação.

Referências

Albuquerque, K. (2022). Estudo aponta que 78% dos profissionais querem migrar para tecnologia. https://olhardigital.com.br/2022/08/08/pro/estudo-profissionais-migrar-tecnologia/. [Online; Acessado em 15 de mar. de 2023].

Amazon (2023). O que é uma aplicação web? https://aws.amazon.com/pt/what-is/web-application/. [Online; Acessado em 30 de mar. de 2023].

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). *Manifesto for Agile Software Development*.

```
const logar = () => {
    if (login !== '' && senha !== '') {
        const data = {
            Login: login,
            Senha: btoa(senha),
        }

        const url = `https://localhost:7183/api/Usuarios/${data.Login},${data.Senha}`;

        axios.post(url).then((result) => {
            if(result.data === true) {
                alert("Usuário encontrado!");
            } else {
                alert("Usuário não encontrado! Login e/ou senha inválidos.");
            }
        })
    }
}
```

Figura 22. Código da função de login

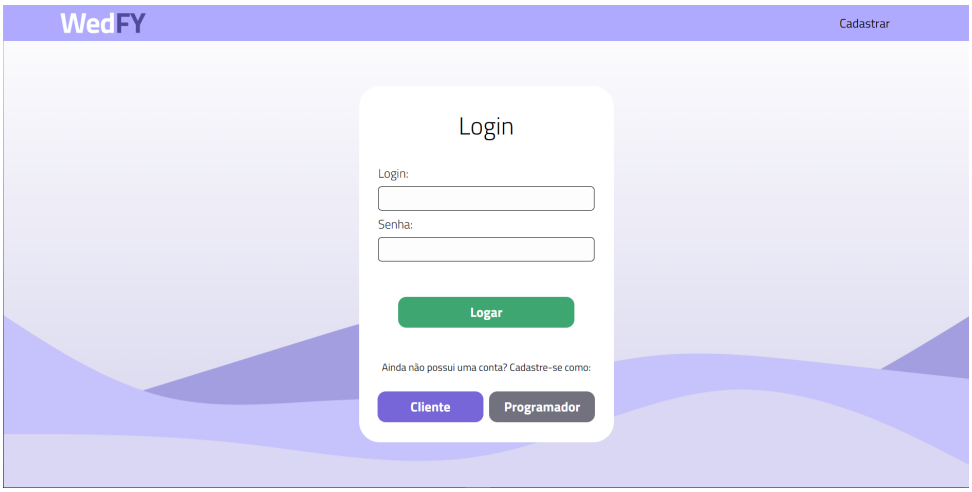

Figura 23. Tela final de login

Beder, D. M. (2017). *Engenharia Web: uma abordagem sistemática para o desenvolvimento de aplicações web*. EdUFSCar.

Breve, F. (2002). Engenharia para a web. Technical report, Departamento de Computação - Universidade Federal de São Carlos.

Ferraz, É. P., Rocha, L., and Tanaka, R. (2014). Findproj: encontre o projeto certo para você. Trabalho de Conclusão de Curso (Graduação em Análise e Desenvolvimento de Sistemas) - Universidade Federal do Paraná.

Júnior, I. (2021). Desenvolvimento de software: Conheça as 4 etapas que toda empresa não cansa de seguir. `https://www.ejemackenzie.com.br/desenvolvimento-de-softwares-4-etapas/`. [Online; Acessado em 30 de mar. de 2023].

Lowe, D. and Pressman, R. S. (2009). *Engenharia Web*. LTC.

Machado, A. (2021). Qual a diferença entre front-end e back-end? `https://tecnoblog.net/responde/qual-a-diferenca-entre-front-end-e-back-end/`. [Online; Acessado em 22 de jun. de 2023].

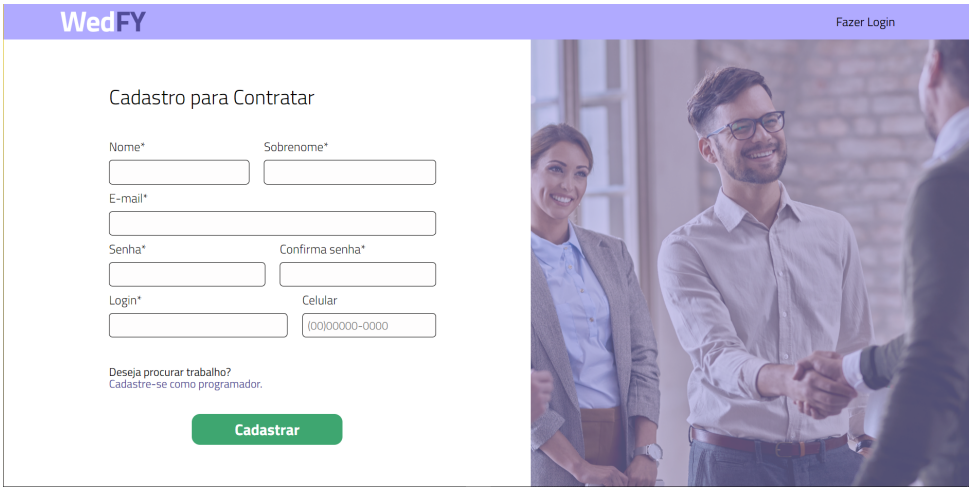

Figura 24. Tela final de cadastro do usuário cliente

Microsoft (2023). Entity framework core. https://learn.microsoft.com/pt-br/ef/core/. [Online; Acessado em 21 de out. de 2023].

Net, C. (2023). Cliente-servidor, uma estrutura lógica para a computação centralizada. https://www.controle.net/faq/cliente-servidor-uma-estrutura-para-a-computacao-centralizada. [Online; Acessado em 17 de jun. de 2023].

PraCarreiras (2021). Mais de 60% dos freelancers no brasil aderiram ao modelo após a pandemia. https://pracarreiras.com.br/freelancer-no-brasil/. [Online; Acessado em 15 de mar. de 2023].

Sutherland, J. and Schwaber, K. (2020). *The Scrum Guide*.

Yuri, D. (2021). Com mais vagas que profissionais, área de TI atrai quem quer mudar de carreira. https://www1.folha.uol.com.br/sobretudo/carreiras/2021/09/com-mais-vagas-que-profissionais-area-de-ti-atrai-quem-quer-mudar-de-carreira.shtml. [Online; Acessado em 17 de jun. de 2023].

Zanobia, L. (2021). Ibge: Desemprego durante a pandemia foi maior que o estimado. https://veja.abril.com.br/economia/ibge-desemprego-durante-a-pandemia-foi-maior-que-o-estimado/. [Online; Acessado em 17 de nov. de 2022].

Documento Digitalizado Público

TCC - Anexo I

Assunto: TCC - Anexo I
Assinado por: Andre Constantino
Tipo do Documento: Relatório
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Andre Constantino da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/03/2024 16:09:50.

Este documento foi armazenado no SUAP em 10/03/2024. 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: 1605987
Código de Autenticação: 69b2c2d2a7