

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

HALLIDAY GAUSS COSTA DOS SANTOS
Orientador: Prof. Dr. Carlos Frederico Marcelo da Cunha Cavalcanti

**AUTÔNOMOS:
O APLICATIVO PARA TRABALHO FREELANCER**

Ouro Preto, MG
2023

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

HALLIDAY GAUSS COSTA DOS SANTOS

**AUTÔNOMOS:
O APlicativo para Trabalho Freelancer**

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Carlos Frederico Marcelo da Cunha Cavalcanti

Ouro Preto, MG
2023

Gauss Costa dos Santos, Halliday.

Autônomos:/ Halliday Gauss Costa dos Santos. –, 2023-
49 p. 1:il. (colors; grafs; tabs).

Orientador: Prof. Dr. Carlos Frederico Marcelo da Cunha Cavalcanti

Monografia – Universidade Federal de Ouro Preto,
Instituto de Ciências Exatas e Biológicas, Departamento de Computação, 2023.

1. *Freelancer*. 2. Trabalhadores Autônomos. 3. Emprego. 4. Smartphone. 5. Aplicativos móveis. 6. JavaScript. 7. React Native. 8. Android. 9. iOS. 10. Node.js. 11. MongoDB. 12. WebSocket. 13. Twilio. 14. Figma. I. Prof. Dr. Carlos Frederico Marcelo da Cunha Cavalcanti. II. Universidade Federal de Ouro Preto. III. Autônomos:

Halliday Gauss Costa dos Santos

**AUTÔNOMOS:
O APLICATIVO PARA TRABALHO FREELANCER**

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau em Bacharel em Ciência da Computação.

Aprovada em Ouro Preto, XX de mês de Ano.

Prof. Dr. Carlos Frederico Marcelo da Cunha Cavalcanti
Universidade Federal de Ouro Preto - UFOP
Orientador

Prof. M.Sc. Vinicius Antonio de Oliveira Martins
Universidade Federal de Ouro Preto - UFOP
Examinador

Prof. Dr. Membro da Banca 2
Universidade Federal de ... - UFXX
Examinador

Este trabalho é dedicado aos meus pais, Rangel Freitas e Ivone Patrícia, e a minha irmã Hallizy Gauss. Obrigado por todo o apoio e suporte, sem vocês nada disso seria possível.

Agradecimentos

Em primeiro lugar, agradeço à Deus, que não me deixou desistir e me guiou nessa longa jornada com muitas bençãos.

Aos meus pais, Rangel Freitas e Ivone Patrícia, pelo sacrifício que fizeram em todos esses anos de estudos, amo muito vocês.

A minha família, especialmente ao meu tio Elivaldo e minha tia Elivania por todo apoio e incentivo.

Aos meus amigos: Emanuel Jesus, William Gomes, Guilherme Augusto, Pedro Igor, Lucas Andrade, Vinicius de Paula, Vinicius Targa, Gabriel Bicalho, Marcos Ponte, Luiz Ravell e Kevin Willer pelo companheirismo, pela amizade incondicional, e pela excelente convivência e troca de experiência que me permitiu crescer como pessoa e profissionalmente.

À instituição de ensino UFOP, essencial no meu processo de formação profissional.

Aos professores, pela dedicação, paciência e amizade, e por todos os ensinamentos, correções e conselhos que certamente guiaram o meu aprendizado.

Ao meu orientador Prof. Carlos Frederico, conhecido como Prof. CFRED, pelo compromisso, confiança e por acreditar em mim e no sucesso deste trabalho. Muito obrigado a todos!

"Quando foi a última vez que você quis algo de verdade? Se você quer, você quer, então trace um plano, o que vai ser? É a dedicação máxima. E não tem uma pessoa na face da Terra que não conseguiu o que quis de verdade, isso segue invicto. Dedicação real, corta tudo que te atrasa e que não vai te fazer chegar lá."

- Douglas Viegas ([NINJA](#), 2020).

Resumo

De acordo com IBGE, em 2021, o número de trabalhadores autônomos no Brasil chegou a mais de 25 milhões, todavia, a condição desses trabalhadores está associada a incertezas quanto aos rendimentos, dependência da demanda da clientela e a vulnerabilidade a eventos que os impeçam, permanente ou temporariamente, de continuarem trabalhando e obterem algum rendimento. Ainda, segundo o IBGE mais de 80% dos domicílios do Brasil possuem internet e o smartphone é o principal aparato tecnológico utilizado para acessá-la. Portanto, no intuito de ajudar os *freelancers* a aumentarem suas rendas, seria viável um aplicativo que os permitisse encontrar mais propostas de emprego e divulgar seus serviços. Este trabalho apresenta uma possível solução para esse problema por meio do desenvolvimento de um aplicativo chamado AUTÔNOMOS, feito para as plataformas Android e iOS, utilizando JavaScript juntamente da biblioteca React Native. Neste projeto é abordado os principais trabalhos correlatos, assim como o diferencial que a aplicação construída terá em relação aos mesmos. Também é apresentado as tecnologias utilizadas para a construção e modelagem da aplicação, como: o React Native para o *front-end*, o Node.js para o *back-end*, o MongoDB para o armazenamento de dados, a Twilio para autenticação via SMS, o protocolo WebSocket para a troca de dados de forma síncrona e o Figma para a criação do protótipo e fluxo das telas.

Palavras-chave: *Freelancer*. Trabalhadores Autônomos. Emprego. Smartphone. Aplicativos móveis. JavaScript. React Native. Android. iOS. Node.js. MongoDB. WebSocket. Twilio. Figma.

Abstract

According to IBGE, in 2021, the number of self-employed workers in Brazil reached more than 25 million, however, the condition of these workers is associated with uncertainties regarding the income, dependence on customer demand and vulnerability to events that prevent them, permanently or temporarily, to continue working and earn some income. Also, according to the IBGE, more than 80% of households in Brazil have internet and smartphones. It is the main technological apparatus used to access it. Therefore, in order to help freelancers to increase their income, an application that allows them to find more job offers and publicize your services. This work presents a possible solution to this problem through the development of an application called Autonomous, made for Android and iOS platforms, using JavaScript together with the React Native library. This project addresses the main related works, as well as the differential that built application will have in relation to them. It also presents the technologies used for building and modeling the application, such as: React Native for the front-end, Node.js for the backend, MongoDB for the datastore, Twilio for authentication via SMS, the WebSocket protocol for synchronous data exchange and Figma for creating of the prototype and flow of the screens.

Keywords: Freelancer. Autonomous Workers. Job. Smartphone. Mobile apps. JavaScript. React Native. Android. iOS. Node.js. MongoDB. WebSocket. Twilio. Figma.

Lista de Ilustrações

Figura 1.1 – Variedade de profissões autônomas.	2
Figura 1.2 – Porcentagem de domicílios no Brasil que utilizam à Internet.	3
Figura 1.3 – Equipamentos utilizados para acessar à Internet por pessoas de 10 anos ou mais de idade.	4
Figura 2.1 – Exemplo da definição de <i>front-end</i> e <i>back-end</i>	6
Figura 2.2 – Representação do OpenID Connect.	8
Figura 2.3 – Tela inicial do Canva após login.	10
Figura 2.4 – Alguns serviços oferecidos pela plataforma GetNinjas.	12
Figura 2.5 – Telas de solicitação de um pedido de aulas para o Ensino Superior no GetNinjas.	13
Figura 2.6 – Tela de login do GetNinjas.	14
Figura 2.7 – Tela de contatar clientes do GetNinjas.	15
Figura 2.8 – Tela de solicitação de serviço de aulas para Desenvolvimento Web no GetNinjas.	16
Figura 2.9 – Tela de notificações de serviços da aplicação GetNinjas.	17
Figura 2.10–Tela de compra de moedas e tela de liberação do contato de um cliente no GetNinjas.	18
Figura 2.11–Página de login e página de propostas para um trabalhador autônomo da Workana.	19
Figura 2.12–Página inicial do site da aplicação Fiverr.	20
Figura 2.13–Página inicial do site da aplicação 99Freelas.	20
Figura 3.1 – Logomarca do aplicativo AUTÔNOMOS.	25
Figura 3.2 – Tela de carregamento da aplicação.	26
Figura 3.3 – Tela inicial da aplicação.	26
Figura 3.4 – Logando no aplicativo AUTÔNOMOS.	27
Figura 3.5 – Acessando a tela de cadastro.	27
Figura 3.6 – Cadastrando usuário na plataforma.	28
Figura 3.7 – Tela inicial do cliente.	28
Figura 3.8 – Solicitando um serviço de pintura.	29
Figura 3.9 – Removendo um serviço e verificando serviços solicitados.	29
Figura 3.10–Conversando com um pintor autônomo dentro da plataforma.	30
Figura 3.11–Tela inicial do autônomo.	30
Figura 3.12–Usuário autônomo indicando que atuará como Desenvolvedor e Pintor dentro da plataforma.	31
Figura 3.13–Tela de Verificação de Serviços Disponíveis.	31
Figura 3.14–Filtrando serviços de pintura presenciais e mais próximos.	32
Figura 3.15–Desbloqueando o contato de um cliente para um serviço de pintura.	32
Figura 3.16–Atualizando a tela de serviços disponíveis para um autônomo.	33

Figura 3.17–Conversando com um cliente dentro da plataforma.	33
Figura 3.18–Tela de notificações.	34
Figura 3.19–Diretório geral da aplicação contendo dois outros diretórios: "Autonomos"(<i>front-end</i>) e " <i>back-end</i> ".	34
Figura 3.20–Organização geral do <i>front-end</i> do projeto.	35
Figura 3.21–Subdiretório "assets"e ícone que representa um serviço online.	35
Figura 3.22–Diretório "screens"dentro do <i>front-end</i>	36
Figura 3.23–Tela inicial do cliente: interface gráfica e código.	37
Figura 3.24–Código principal da aplicação.	37
Figura 3.25–API utilizada no <i>front-end</i> para fazer requisições ao servidor.	38
Figura 3.26–Organização geral do <i>back-end</i> do projeto.	38
Figura 3.27–Arquivo de configurações do <i>back-end</i>	39
Figura 3.28–Criação do <i>schema</i> de conversas do <i>back-end</i>	39
Figura 3.29–Estabelecendo rotas e <i>endpoints</i> das conversas, e utilizando o <i>endpoint</i> "/cliente"para obter conversas de um cliente consultando o banco de dados.	40
Figura 3.30–Código principal do <i>back-end</i>	40
Figura 3.31–Utilizando a API da Twilio para enviar código de verificação via SMS.	41
Figura 3.32–Criando um servidor <i>http</i> , vinculando o mesmo a aplicação e a um WebSocket, e iniciando o servidor.	41
Figura 3.33–Código para emitir um sinal ao chegar uma mensagem nova, e aguardar um sinal na aplicação.	42
Figura 4.1 – Aplicativo AUTÔNOMOS em execução no Android e iOS.	43
Figura 4.2 – Protótipo na comunidade do Figma e repositório da aplicação no GitHub.	44

Lista de Tabelas

Tabela 3.1 – Requisitos Funcionais.	23
Tabela 3.2 – Requisitos Não Funcionais.	24

Lista de Abreviaturas e Siglas

AJAX	<i>Asynchronous JavaScript and XML</i> /JavaScript Assíncrono e XML
ANATEL	Agência Nacional de Telecomunicações
API	<i>Application Programming Interface</i> /Interface de Programação de Aplicação
CLT	Consolidação das Leis do Trabalho
CSS	<i>Cascading Style Sheets</i> /Folha de Estilo em Cascatas
DECOM	Departamento de Computação
FGV	Fundação Getulio Vargas
HTML	<i>HiperText Markup Language</i> /Linguagem de Marcação de Hipertexto
IBGE	Instituto Brasileiro de Geografia e Estatística
JS	JavaScript
JSON	<i>JavaScript Object Notation</i>
PNAD	Pesquisa Nacional por Amostra de Domicílio
SMS	<i>Short Message Service</i> /Serviço de Mensagens
UI	<i>User Interface</i> /Interface do Usuário
UFOP	Universidade Federal de Ouro Preto
UX	<i>User Experience</i> /Experiência do Usuário

Sumário

1	Introdução	1
1.1	Justificativa	2
1.2	Objetivos	4
1.2.1	Objetivo Geral	4
1.2.2	Objetivos Específicos	5
1.3	Organização do Trabalho	5
2	Revisão Bibliográfica	6
2.1	Fundamentação Teórica	6
2.1.1	Front-end e Back-end	6
2.1.2	HTML	7
2.1.3	CSS	7
2.1.4	JavaScript	7
2.1.5	MongoDB	7
2.1.6	WebSocket	8
2.1.7	Twilio	8
2.1.8	OpenID Connect	8
2.1.9	Node.js	9
2.1.10	React Native	9
2.1.11	React	9
2.1.12	Figma	9
2.1.13	Canva	9
2.1.14	Git e GitHub	10
2.2	Trabalhos Relacionados	11
2.2.1	GetNinjas	11
2.2.2	Workana	18
2.2.3	Fiverr	19
2.2.4	99Freelas	20
2.2.5	Profes	21
2.2.6	Uber	21
2.2.7	iFood	21
3	Desenvolvimento	22
3.1	Especificação de Requisitos	22
3.1.1	Missão do Software	22
3.1.2	Levantamento de Requisitos	22
3.1.2.1	Requisitos Funcionais	22
3.1.2.2	Requisitos Não Funcionais	24

3.2	Protótipos e Fluxos de Telas	24
3.2.1	Logomarca do Autônomos	25
3.2.2	Visão Geral do Protótipo	25
3.2.2.1	Tela de Carregamento	25
3.2.2.2	Tela Inicial	26
3.2.2.3	Tela de Cadastro	27
3.2.2.4	Tela Inicial do Cliente	28
3.2.2.5	Solicitando Serviços	29
3.2.2.6	Verificando Serviços Solicitados	29
3.2.2.7	Conversas com autônomos	30
3.2.2.8	Tela Inicial do Autônomo	30
3.2.2.9	Escolhendo Áreas de Atuação	30
3.2.2.10	Verificando Serviços Disponíveis	31
3.2.2.11	Conversas com Clientes	33
3.2.2.12	Tela de Notificações	33
3.3	Implementação da Aplicação	34
3.3.1	Front-end do Autônomos	34
3.3.2	Back-end do Autônomos	38
4	Resultados	43
5	Considerações Finais	45
5.1	Conclusão	45
5.2	Trabalhos Futuros	45
Referências	46	

1 Introdução

Autônomo é uma pessoa que trabalha de forma independente e sem vínculo empregatício, exercendo sua atividade profissional de acordo com suas próprias habilidades e conhecimentos. Além disso, os autônomos prestam serviços para diversos clientes, pessoas ou empresas e, dependendo do serviço prestado e do cliente, eles podem escolher seus horários de trabalho e decidir se vão trabalhar ou não, ou então trabalhar menos em um determinado período do ano, possibilitando a realização de outras atividades nos momentos menos produtivos ([DINIZ; VARELA](#),).

O trabalhador autônomo necessita de muita organização para executar suas atividades já que a remuneração varia bastante conforme a quantidade de projetos e clientes, o local e os acontecimentos da época: pandemia do coronavírus, por exemplo. Os trabalhadores autônomos também não possuem os mesmos direitos que os trabalhadores de carteira assinada, como férias e 13º salário, conforme previsto na legislação trabalhista brasileira, conhecida como CLT.

Nesse contexto, os trabalhadores autônomos também podem ser chamados de *freelancers*. O termo *freelancer*, vem do inglês, significa o mesmo que autônomo. No entanto, existem outras categorias de trabalhadores que atuam de maneira semelhante aos autônomos, como:

- **Profissional Liberal:** trabalhador que exerce uma profissão de nível técnico ou superior, ou seja, sua formação lhe concede liberdade. Podem ser empregados em uma empresa ou exercer suas atividades como liberais, obtendo a remuneração através dos serviços prestados. Em alguns casos a forma como deve ser praticada a profissão está regulamentada pelo Conselho responsável pela área de atuação. São exemplos deste tipo de trabalho os médicos, os dentistas, os engenheiros, os arquitetos, os professores, diversos programadores, entre outros.
- **Trabalhador Eventual:** trabalhador que presta serviços sem vínculo empregatício e sem garantir a permanência no trabalho, em outras palavras, não estar em atuação constante para o mesmo contratante. Exemplos: técnicos em manutenção e montadores de móveis.
- **Trabalhado Voluntário:** nesse tipo de trabalho não há remuneração envolvida e geralmente é organizado em prol de causas sociais e humanitárias, porém qualquer serviço prestado na intenção de ajudar o próximo sem a intenção de obter de algo em troca também se encaixa nessa categoria de trabalho.
- **Trabalhador Informal:** aquele trabalhador que exerce suas atividades sem fazer as suas contribuições para a Previdência e nem declara sua renda a Receita Federal. O trabalhador informal não tem contrato com quem faz os pagamentos pela atividade, consequentemente

não tem nenhum tipo de amparo legal. O trabalho informal é muito comum no Brasil, e é resultado da falta de oportunidade, altos níveis de desemprego e desamparo ao desempregado e a sua família ([SUISSO, 2006](#)). A dificuldade maior deste tipo de trabalho é não contar tempo de aposentadoria e nem permitir que o profissional tenha direitos como: licença saúde ou aposentadoria por invalidez. São exemplos de trabalhadores informais: diaristas, jardineiros, carregadores, pintores, pedreiros, motoristas.

A figura 1.1 apresenta diversas profissões que, desde que não haja vínculo empregatício, podem ser consideradas autônomas.



Figura 1.1 – Variedade de profissões autônomas.

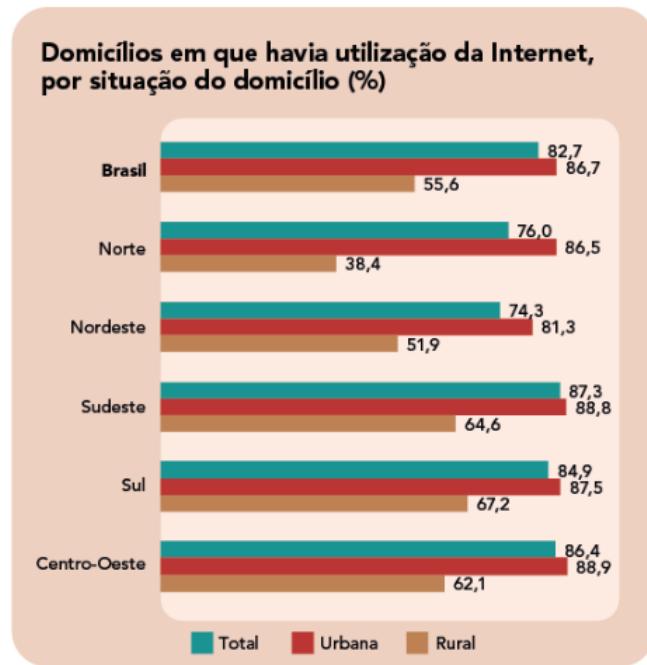
Fonte: ([TEIXEIRA, 2020](#)).

Segundo o IBGE, trabalho autônomo é a categoria empírica de pessoas que exploram seu próprio empreendimento, sozinhas ou com um sócio, sem empregar auxiliar assalariado, mas podendo se valer de auxílio de pessoa não remunerada. De acordo com IBGE, em 2021, o número de trabalhadores autônomos no Brasil chegou a mais de 25 milhões, ainda, o rendimento médio de um trabalhador autônomo é de R\$ 2.021 por mês, sendo 3% menor que em 2020 e o menor rendimento dos últimos cinco anos ([NACIONAL, 2022](#)).

A precariedade da condição desses trabalhadores está associada a incertezas quanto aos rendimentos, dependência da demanda da clientela e a vulnerabilidade a eventos que os impeçam, permanente ou temporariamente, de continuarem trabalhando e obterem alguma remuneração ([HOLZMANN, 2013](#)), por exemplo, a pandemia do coronavírus que gerou muitos prejuízos a esses trabalhadores ([BITENCOURT, 2021](#)).

1.1 Justificativa

No Brasil 82,7% dos domicílios possuem acesso à Internet (Figura 1.2). Além disso, existem 424 milhões de dispositivos digitais em uso, os quais 234 milhões são smartphones ([FGV, 2020](#)).



Fonte: IBGE, Diretoria de Pesquisas, Coordenação de Trabalho e Rendimento, Pesquisa Nacional por Amostra de Domicílios Contínua 2019.

Figura 1.2 – Porcentagem de domicílios no Brasil que utilizam à Internet.

Fonte: (IBGE, 2021)

Segundo a Pesquisa Nacional por Amostra de Domicílios Contínua 2018 (PNAD), o smartphone é o equipamento mais usado para acessar à Internet, conforme observa-se na figura 1.3 (IBGE, 2020). Ainda, de acordo com a ANATEL, em julho de 2020 existiam 225,89 milhões de smartphones em uso no Brasil, desses, 12,25% utilizavam a tecnologia 2G para acessar a internet, 16,30% utilizavam 3G e 71,45% utilizavam 4G para essa finalidade (FILHO; CASTIONI, 2021).



Figura 1.3 – Equipamentos utilizados para acessar à Internet por pessoas de 10 anos ou mais de idade.

Fonte: (IBGE, 2021)

Durante a pandemia de covid-19 o Brasil teve uma alta taxa de desemprego, muitos trabalhadores brasileiros sofreram prejuízos e recorreram a aplicativos para garantir o sustento da família. A pesquisa do Instituto Locomotiva mostrou um crescimento de 7% no número de trabalhadores que fazem o uso de aplicações tecnológicas para obter serviço entre fevereiro de 2020 e março de 2021, antes 13% e agora são 20%, um em cada cinco trabalhadores usufruem dessas ferramentas como uma fonte de renda (CAVALCANTE, 2021).

Visto que boa parte da população brasileira tem acesso à Internet e utilizam em sua maioria smartphones, e sabendo das oscilações e riscos que os autônomos têm em relação as suas atividades econômicas, seria viável encontrar uma maneira de aumentar a renda desses trabalhadores por meio de um aplicativo.

1.2 Objetivos

A seguir são apresentados o objetivo geral e os objetivos específicos deste trabalho.

1.2.1 Objetivo Geral

O objetivo geral desse trabalho é o desenvolvimento de um aplicativo móvel chamado AUTÔNOMOS, para as plataformas Android e iOS, no intuito de servir como uma alternativa para aumentar a renda dos trabalhadores autônomos. A ideia é introduzir toda forma de trabalho

supracitada dentro do aplicativo: Profissional Liberal, Eventual, Voluntário e Informal. Dentro do software, os *freelancers* podem se cadastrar nas áreas de trabalho de interesse e receber notificações quando clientes estiverem necessitando de um serviço afim. Os autônomos poderão visualizar os tipos de serviços solicitados e caso se interessem podem entrar em contato com os clientes dentro da aplicação por meio de um chat.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral, os seguintes objetivos específicos terão de ser alcançados:

- Levantar o diferencial da aplicação através da análise dos trabalhos correlatos;
- Selecionar tecnologias a serem utilizadas;
- Levantar os requisitos do software;
- Produzir o protótipo e fluxo das telas;
- Desenvolver o código da aplicação (*Back-end* e *Front-end*).

1.3 Organização do Trabalho

Esse trabalho está estruturado em 5 capítulos. Capítulo 1: Introdução. É apresentado o problema e como será resolvido, juntamente dos objetivos gerais e específicos deste trabalho. Capítulo 2: Revisão Bibliográfica: referencial teórico e trabalhos relacionados. Nesse capítulo é dissertado sobre as tecnologias que serão utilizadas no desenvolvimento, posteriormente é realizado uma análise crítica dos trabalhos relacionados. Capítulo 3: Metodologia ou Desenvolvimento. No capítulo 3 é abordado os requisitos, funcionais e não funcionais, que a aplicação terá, a construção do protótipo com o fluxo das telas utilizando o Figma, e a implementação da aplicação usando as tecnologias escolhidas. Capítulo 4: Resultados e Discussões. O capítulo 4 tem ênfase na análise do projeto desenvolvido. Capítulo 5: Conclusão e trabalhos futuros.

2 Revisão Bibliográfica

Este capítulo apresenta os conceitos e tecnologias utilizadas para o desenvolvimento da aplicação, tal como uma análise dos trabalhos correlatos.

2.1 Fundamentação Teórica

Nesta seção é proporcionado uma fundamentação teórica e explicações acerca dos conceitos discutidos ao longo do texto. Tais conceitos também dizem a respeito das tecnologias que serão utilizadas para o desenvolvimento do aplicativo.

2.1.1 Front-end e Back-end

O *front-end* está relacionado com a codificação da interface gráfica do projeto e com a arquitetura *client-side* (processamento diretamente no navegador), ou seja, é onde se desenvolve a aplicação com a qual o usuário irá interagir diretamente seja em softwares, sites, aplicativos, dentre outros, e o foco é o design, interface de navegação e ferramentas de interação com o usuário. Portanto, para desenvolver o *front-end* é necessário saber algumas linguagens de programação como HTML (linguagem de marcação), CSS (linguagem de estilo) e JavaScript (linguagem de script/programação), além de noções de design e UX (*User Experience/ Experiência do Usuário*) ([DIGITALHOUSE, 2019b](#)).

O *back-end* está relacionado com a arquitetura *server-side* (processamento diretamente no servidor), sendo assim, está diretamente ligado a tarefas que envolvem servidores, banco de dados, segurança e atualizações. Geralmente as linguagens de programação e tecnologias utilizadas para realizar essas funções são: TypeScript, Ruby, Java, Python, PHP e Node.js ([DIGITALHOUSE, 2019a](#)). A figura 2.1 retrata a comunicação entre *front-end* e *back-end*.



Figura 2.1 – Exemplo da definição de *front-end* e *back-end*.

Fonte: ([FERNANDES, 2020](#)).

2.1.2 HTML

O HTML (*HyperText Markup Language* - Linguagem de Marcação de Hipertexto), criada pelo britânico Tim Berners-Lee, é a principal linguagem utilizada na web. O HTML permite inserir o conteúdo e estabelecer a estrutura básica de um website, criando documentos estruturados em títulos, parágrafos, listas, links, tabelas, formulários, podendo incorporar imagens, objetos, animações ou vídeos. Portanto, é essencial o uso do HTML para o navegador entender como deverá exibir o conteúdo da internet (FLATSCHART, 2011).

2.1.3 CSS

CSS é uma linguagem de folhas de estilos, cuja sigla significa *Cascading Style Sheets*, ou seja, Folhas de Estilo em Cascatas em tradução livre. É uma maneira de dar estilo ao código criado por linguagens de marcação como o HTML, personalizando o conteúdo visível por meio de formatações e modificações no layout, como definição de cores e fontes. O CSS foi criado justamente pela dificuldade na criação de padrões na formatação das páginas web utilizando o HTML, enquanto o HTML é usado para estruturar os conteúdos, o CSS é utilizado para formatá-los e dar mais estética aos sites por meio de personalizações, evitando repetições de código (OKUBO, 2021).

2.1.4 JavaScript

JavaScript, ou JS, é uma linguagem de programação de alto nível, interpretada, com tipagem dinâmica fraca, multiparadigma e *client-side* (executada ao lado do usuário). É a linguagem mais utilizada em navegadores web e no *front-end*, em companhia do HTML e CSS. Através do JS é possível incluir em uma página estática elementos dinâmicos como: mapas, formulários, operações numéricas, animações, infográficos interativos, dentre outras funcionalidades. Contudo, essa linguagem não está limitada ao *front-end* e as páginas web, através dela, em conjunto dos seus *frameworks* e APIs, é possível desenvolver também aplicativos *mobile* por meio da biblioteca React Native, e trabalhar com o *back-end* utilizando o Node.js (ambiente de execução JavaScript *server-side*) (ROVEDA, 2020a).

2.1.5 MongoDB

O MongoDB é um software/servidor de banco de dados NoSQL (not only SQL) orientado a documentos que possui código aberto (*open source*) e foi projetado para permitir que se trabalhe de forma eficiente com grandes volumes de dados. A recuperação de dados no MongoDB não é feita no formato de tabelas, mas sim através de documentos semelhantes a documentos de formato JSON. O banco de dados também fornece suporte para diversas linguagens de programação como: C, C++, C#, Go, Java, Node.js, PHP, Python, Ruby, dentre outras (TECNOBLOG, 2021).

2.1.6 WebSocket

O WebSocket é um protocolo de comunicação bidirecional que permite que os dados sejam transmitidos entre o cliente e o servidor de forma síncrona. Ele oferece uma alternativa mais eficiente e escalável do que a comunicação baseada em AJAX (requisição assíncrona), permitindo que os desenvolvedores criem aplicativos web com menos latência e sobrecarga de rede (FETTE; MELNIKOV, 2011).

2.1.7 Twilio

A Twilio é uma plataforma de comunicação em nuvem que oferece APIs para envio e recebimento de mensagens de texto, chamadas de voz, chamadas de vídeo e autenticação de usuários. A API de envio de SMS da Twilio é fácil de usar, altamente escalável e personalizável, permitindo que desenvolvedores adicionem recursos de comunicação em seus aplicativos JavaScript com facilidade. Além disso, a plataforma oferece relatórios e análises detalhados sobre o envio de SMS para ajudar as empresas a medir o sucesso de suas campanhas de comunicação (TWILIO, 2023).

2.1.8 OpenID Connect

O OpenID Connect é um protocolo aberto *Single Sign-On* (autenticação única) que é construído usando o protocolo OAuth 2.0, o qual é utilizado pelo OpenID para autenticação e autorização, podendo envolver sites de terceiros como Google e Facebook, em seguida, o protocolo visa construir identidades que identificam os usuários exclusivamente (IBM, 2021). A figura 2.2 apresenta a ideia desse protocolo.

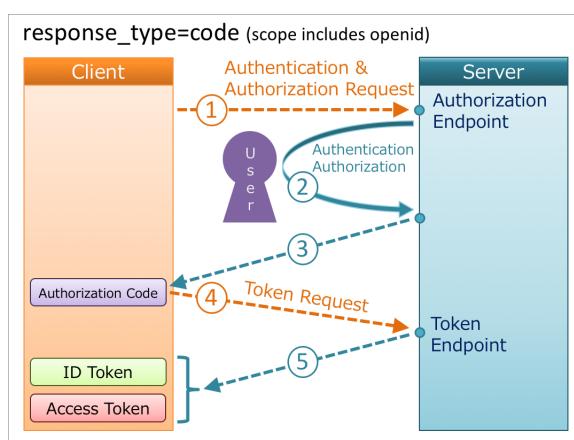


Figura 2.2 – Representação do OpenID Connect.

Fonte: (KAWASAKI, 2017)

2.1.9 Node.js

O Node.js pode ser definido como um ambiente de execução JavaScript *server-side* que utiliza *single-thread* para obter requisições. Portanto, utilizando o Node.js é possível criar aplicações JavaScript para trabalhar com o *back-end* mantendo na mesma linguagem que o *front-end*, garantindo um bom reuso de código, além disso, as bases de dados NoSQL são baseadas em JSON (JavaScript Object Notation), tornando a comunicação com Node.js bastante intuitiva ([LENON, 2018](#)).

2.1.10 React Native

Criado pelo Facebook em 2015 sobre a licença MIT, o React Native é uma biblioteca para desenvolvimento de aplicativos móveis multiplataforma (Android e iOS) utilizando apenas JavaScript. Com o React Native, a partir de apenas um código é possível gerar aplicações tanto para Android quanto para iOS, pois o código produzido é convertido para a linguagem nativa do sistema operacional em questão, o que também torna a aplicação mais fluida ([NATIVE, 2022](#)).

2.1.11 React

Criado em 2011 pelo Facebook, React é uma biblioteca *front-end* do JavaScript para a criação de interfaces de usuário (UI). Com o React é possível manter uma conexão simplificada entre HTML, CSS e JavaScript, permitindo a criação de aplicações web que são executadas em navegadores de diversos dispositivos, inclusive desktop e *mobile* ([ROVEDA, 2020b](#)).

2.1.12 Figma

Figma é um software online para edição e design gráfico focado na criação de interfaces gráficas e experiência de usuário. Através desse software é possível prototipar telas de aplicativos e o fluxo delas de acordo com a interação do usuário. O Figma também suporta o acesso simultâneo em tempo real a um projeto, o que facilita o trabalho remoto e em equipe ([GARRETT, 2021](#)). Apesar de possuir planos pagos, o plano gratuito possui diversas funcionalidades e consegue atender bem inúmeros usuários ([NAMA, 2020](#)).

2.1.13 Canva

O Canva (Figura 2.3) é uma ferramenta gratuita de design gráfico online que permite aos usuários criar diversos conteúdos gráficos como: posts para redes sociais, apresentações, cartazes, vídeos, pôsteres, ilustrações, logotipos, logomarcas, dentre outros conteúdos visuais ([CANVA, 2022](#)).



Figura 2.3 – Tela inicial do Canva após login.

Fonte: ([CANVA, 2022](#)) (Modificado pelo Autor).

2.1.14 Git e GitHub

Git, criado por Linus Torvalds em 2005, é um sistema de controle de versão de código-fonte distribuído que permite que várias pessoas trabalhem em um mesmo projeto de maneira simultânea e organizada ([STRAUB, 2009](#)).

Algumas funcionalidades do Git incluem:

- Criação de *branches* (ramificações) para trabalhar em novas funcionalidades sem afetar o código principal;
- *Merging* (mesclagem) de branches para juntar as alterações de volta ao código principal;
- *Commit* (registro) de alterações para salvar o histórico de evolução do projeto.

Já o GitHub, fundado em 2008 por Tom Preston-Werner, Chris Wanstrath e PJ Hyett, é uma plataforma online que fornece armazenamento e controle de versão para projetos (repositórios) utilizando o Git. Essa plataforma oferece recursos adicionais, como *issues*, *pull requests* e *wikis* ([INC, 2020](#)).

Algumas funcionalidades do GitHub incluem:

- Compartilhamento de projetos com outras pessoas;
- Colaboração em projetos com outros desenvolvedores através de *pull requests*;
- *Issues* para rastreamento de problemas e tarefas.

2.2 Trabalhos Relacionados

Existem diversos aplicativos com o propósito de conectar clientes aos provedores de serviços e produtos. As grandes aplicações nesse nicho do mercado possuem recursos úteis para os usuários, e, portanto, seria interessante analisá-los. Nesta seção é apresentado os mais relevantes trabalhos correlatos à aplicação AUTÔNOMOS e seguidamente é feita uma análise crítica deles com a finalidade de construir uma aplicação móvel baseada nas funcionalidades interessantes desses softwares e nas inovações que podem ser realizadas sobre os pontos negativos desses programas.

Os aplicativos GetNinjas, Workana, Fiverr, 99Freelas estão entre as principais aplicações utilizadas por *freelancers* para obter serviços (MAGALHÃES, 2021). Já o Profes está entre os melhores softwares para encontrar professores particulares (ANGÉLICA, 2017). Ainda, apps como Uber e iFood são um dos maiores empregadores do Brasil (CONTEÚDO, 2019). Sendo essas as aplicações supracitadas, as escolhidas para a análise crítica de suas funcionalidades.

2.2.1 GetNinjas

A plataforma online, web e *mobile*, GetNinjas (Figura 2.4), tem como objetivo conectar clientes a trabalhadores autônomos (*freelancers*) e prestadores de serviço, no Brasil. Essa aplicação disponibiliza inúmeros serviços em diversas áreas, como: Assistência Técnica, Aulas, Consultoria, Design e Tecnologia, Evento, Moda e Beleza, Reformas, Saúde, Serviços Domésticos, dentre outros.



Assistência técnica	Aulas	Autos	Consultoria	Design e Tecnologia	Eventos	Moda e beleza	Reformas e reparos	Saúde	Serviços Domésticos
Aluguel de Maquinário	Construção	Instalação	Reformas e Reparos	Serviços Gerais	Para Casa				
Aluguel de maquinário	Arquitetos	Antenista	Encanador	Chaveiro	Banheira				
	Design de Interiores	Automação residencial	Eletricista	Dedetizador	Coifas e Exaustores				
	Empreiteiro	Instalação de eletrônicos	Gás	Desentupidor	Decorador				
	Engenheiro	Instalador tv digital	Gesso e drywall	Desinfecção	Instalador de Papel de Parede				
	Limpeza pós obra	Segurança eletrônica	Pavimentação	Impermeabilizador	Jardinagem				
	Marmoraria e Granitos	Toldos e Coberturas	Pintor	Marceneiro	Montador de Móveis				
	Pedreiro		Serralheria e solda	Marido de Aluguel	Paisagista				
	Poço Artesiano		Vidraceiro	Mudanças e Carretos	Piscina				

Principais serviços pedidos
Os serviços mais realizados de cada categoria



Montagem de móveis



Mudanças e Carretos



Serviço de Pedreiro

Figura 2.4 – Alguns serviços oferecidos pela plataforma GetNinjas.

Fonte: ([GETNINJAS, 2022](#)) (Modificado pelo Autor).

Por ser uma plataforma que oferece serviços diversos, vários dos mesmos se fará presente na aplicação AUTÔNOMOS. No entanto, diferente do aplicativo AUTÔNOMOS, o GetNinjas não disponibiliza funcionalidades como: transporte de pessoas e trabalho voluntário, além disso, muitos serviços, principalmente na área da educação, poderiam ter mais especializações, as quais seriam reorganizadas dentro dos serviços gerando uma melhor organização do aplicativo.

É possível observar na figura 2.5 que, ao tentar buscar aulas de Matemática para o Ensino Superior, o cliente deve deixar nas informações adicionais qual é a disciplina que ele precisa de ajuda, não existindo um tópico para ser escolhido a respeito das disciplinas possíveis de serem ofertadas, o que leva uma dificuldade ao profissional em filtrar as áreas de interesses.



Figura 2.5 – Telas de solicitação de um pedido de aulas para o Ensino Superior no GetNinjas.

Fonte: (GETNINJAS, 2022) (Modificado pelo Autor).

Ao continuar a avaliar o aplicativo, é notável que a tela de login é bastante intuitiva, e oferece a opção para que o usuário possa escolher logar como cliente ou prestador de serviços, conforme ilustrado na figura 2.6. Independentemente da escolha, o processo de autenticação é realizado através de mensagem de texto (SMS).



Figura 2.6 – Tela de login do GetNinjas.

Fonte: ([GETNINJAS, 2022](#)) (Modificado pelo Autor).

Caso o usuário escolha logar como cliente, ele poderá solicitar um serviço dentre os disponíveis na plataforma e aguardar para que um profissional entre em contato, o qual não é feito através da plataforma, mas sim através do WhatsApp cadastrado ao realizar a autenticação, e pelo fato do WhatsApp ser um aplicativo amplamente utilizado pelos brasileiros ([PURZ, 2022](#)), torna esse recurso muito interessante para comunicação entre cliente e prestador de serviço. Por outro lado, essa comunicação gera uma quebra de privacidade, já que o cliente e o autônomo terão compartilhado os números de WhatsApp entre si. Portanto, seria vantajoso manter a confidencialidade dos usuários, através de uma ferramenta de comunicação interna no aplicativo, como um chat.

Além disso, o cliente pode ser contatado de maneira gratuita para cada pedido que ele solicita por apenas 4 profissionais (Figura 2.7), gerando uma dificuldade para encontrar um profissional adequado. Para resolver esse problema o AUTÔNOMOS permitirá que o cliente defina quantos profissionais podem contatá-lo.



Figura 2.7 – Tela de contatar clientes do GetNinjas.

Fonte: (GETNINJAS, 2022) (Modificado pelo Autor).

Outro problema que acontece na área do cliente é que, ao solicitar um pedido, é perguntado quando é pretendido receber o serviço, mas as opções possíveis de escolha não ajudam a definir uma data certeira. Por essa razão, seria mais intuitivo se houvesse um calendário, no qual o cliente poderia marcar o dia desejado para o atendimento (Figura 2.8).

The screenshot shows a service request form on the GetNinjas platform. The user has selected 'Desenvolvimento Web' from a grid of categories. Arrows point from this selection to several questions:

- Pra qual tipo de Aulas você precisa de um orçamento?** (Artes, Aulas de Artesanato, Beleza, Bem-Estar, Circo, Concursos, Aulas de Dança, Educação Especial)
- Qual aula você procura?** (Criação de Websites, Desenvolvimento de Aplicativos, Dreamweaver, Wix, Wordpress)
- A aula será:** (Online, Presencial)
- Qual será a frequência das aulas?** (1x por semana, 2x por semana, 3x por semana, Aula única, Mensal, Não tenho certeza, Quinzenal)
- Quando você pretende realizar a aula?** (Nos próximos 15 dias, Nos próximos 30 dias, Nos próximos 6 meses, Nos próximos 7 dias, Não tenho data definida, Urgente (o quanto antes possível))
- Qual é o melhor período para as aulas?** (Manhã, Noite, Não tenho certeza, Tarde)

At the bottom left is a 'CONTINUAR' button.

Figura 2.8 – Tela de solicitação de serviço de aulas para Desenvolvimento Web no GetNinjas.

Fonte: (GETNINJAS, 2022) (Modificado pelo Autor).

Ainda, o aplicativo móvel GetNinjas envia mensagens na barra de notificação para cada possível cliente próximo do autônomo, sendo que cada notificação pode ser de uma área distinta que o trabalhador atua, não tendo muita utilidade, pois, ao receber muitas notificações o trabalhador não consegue verificar com detalhes quais são os tipos de serviços que estão sendo solicitados (Figura 2.9). Seria mais intuitivo se o aplicativo enviasse somente uma única notificação avisando que o profissional possui novos clientes e permitir filtrar serviços dentro da aplicação.

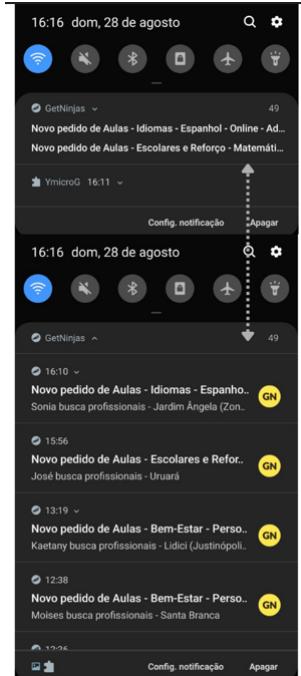


Figura 2.9 – Tela de notificações de serviços da aplicação GetNinjas.

Fonte: ([GETNINJAS, 2022](#)) (Modificado pelo Autor).

Para desbloquear o contato de um cliente, o profissional deve gastar suas moedas que são compradas dentro do aplicativo por meio de pacotes, sendo que o pacote mais barato atualmente provê 1000 moedas ao comprador pelo valor de R\$ 149,99. O preço de cada contato de cliente varia de acordo com o serviço. Entretanto, o profissional pode desbloquear um cliente por 141 moedas, o que equivale aproximadamente a R\$ 21,00, e pode não conseguir o serviço, tendo pago apenas por ter obtido o contato do cliente (Figura 2.10). O aplicativo AUTÔNOMOS minimizará essa perda do autônomo, sendo gratuito a liberação do contato do cliente independente do serviço.

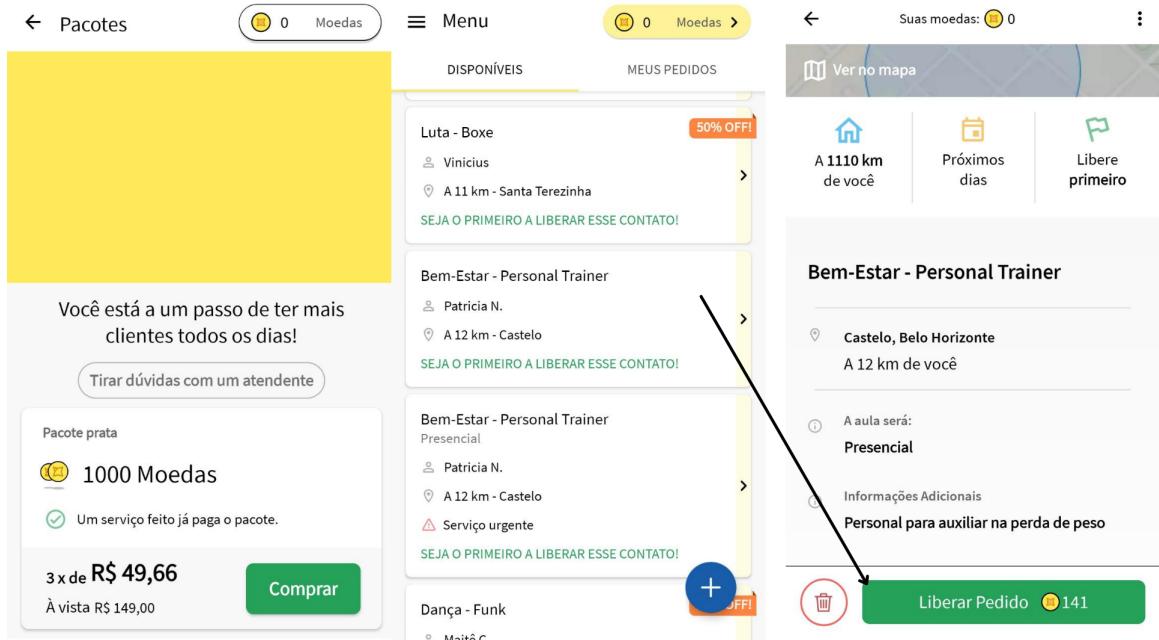


Figura 2.10 – Tela de compra de moedas e tela de liberação do contato de um cliente no GetNinjas.

Fonte: (GETNINJAS, 2022) (Modificado pelo Autor).

2.2.2 Workana

Workana (Figura 2.11) é uma plataforma, web e *mobile*, que atua como mediadora no contato entre profissionais e clientes através de publicações de anúncios. O cliente indica o que está procurando um serviço e os *freelancers* cadastrados podem enviar propostas. Em seguida, o cliente pode analisar as mensagens e os perfis de cada profissional, além de poder entrar em contato individualmente dentro da plataforma com cada profissional interessado. Após escolher o profissional para realizar o projeto, a plataforma estabelece uma fase de garantia onde o cliente realiza o pagamento acordado dentro da plataforma e o dinheiro fica retido no sistema até que o profissional entregue o trabalho.

Além disso, na Workana existe uma funcionalidade que permite avaliar tanto o contratante quanto o contratado ao final do serviço e tais recomendações ficam visíveis para serem lidas por outros usuários.

O AUTÔNOMOS tem o mesmo objetivo que a Workana, porém, diferente dessa aplicação, o AUTÔNOMOS permite que o provedor de serviço desbloqueie quantos clientes e serviços forem necessários em qualquer período e, como não terá ranking de profissionais no aplicativo, profissionais iniciantes tem as mesmas chances de conseguir um serviço que os profissionais que já utilizam a aplicação a mais tempo. Ainda, o aplicativo desenvolvido permite que todo o lucro fique para o trabalhador, diferente da Workana que adquire de 5% a 20% de cada serviço realizado.

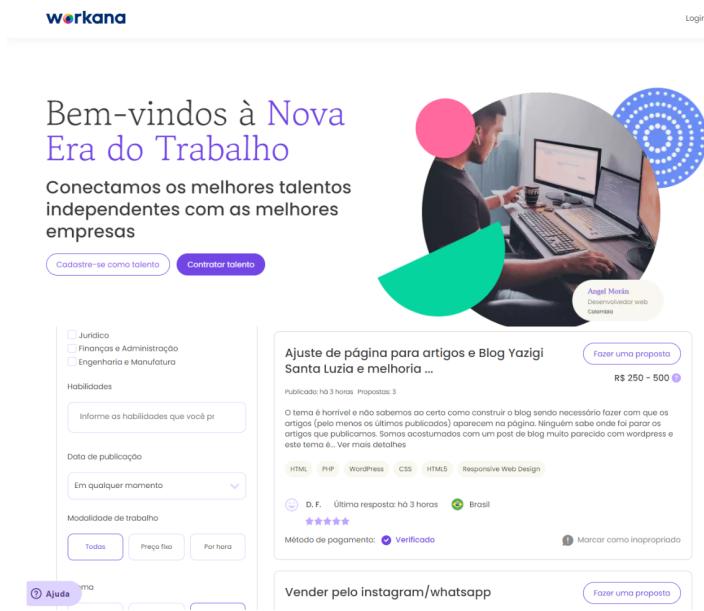


Figura 2.11 – Página de login e página de propostas para um trabalhador autônomo da Workana.

Fonte: (WORKANA, 2022) (Modificado pelo Autor).

2.2.3 Fiverr

O Fiverr (Figura 2.12) é uma aplicação, web e *mobile*, que permite que clientes encontrem trabalhadores que realizam serviços *freelancers*. Diferente de muitas plataformas e aplicativos, o Fiverr engloba *freelancers* e clientes de toda parte do mundo.

Nessa plataforma, o profissional se cadastra nas suas áreas de interesses e, diferentemente do AUTÔNOMOS, é o cliente que buscará o profissional com base em suas necessidades, ou seja, a aplicação funciona como uma vitrine de profissionais podendo um profissional ser contatado pouquíssimas vezes. Depois que o trabalho for entregue e aceito pelo cliente, o pagamento é liberado para o contratado, porém a plataforma ficará com 20% do valor do pagamento.

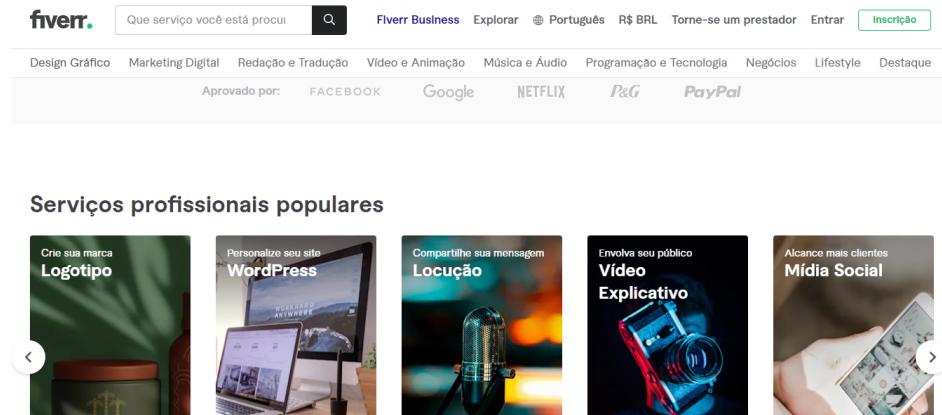


Figura 2.12 – Página inicial do site da aplicação Fiverr.

Fonte: (FIVERR, 2022).

2.2.4 99Freelas

O 99Freelas (Figura 2.13) é uma plataforma criada no intuito de ajudar na contratação de profissionais *freelancers*, sem vínculo empregatício por uma determinada empresa. Para utilizar a plataforma, o autônomo deve criar seu perfil com seus dados pessoais, habilidades e áreas de atuação.

Após o cadastro, o trabalhador pode se candidatar gratuitamente nas vagas de interesse sendo necessário informar o valor do serviço. Quando contratado, o trabalhador receberá o dinheiro no sistema e, após a realização da tarefa, o dinheiro é liberado para transferências em contas bancárias, sendo que 15% do valor ganho irá para a plataforma.



Figura 2.13 – Página inicial do site da aplicação 99Freelas.

Fonte: (99FREELAS, 2022).

2.2.5 Profes

O Profes (Figura ??) é uma das maiores plataformas de educação tecnológica, web e *mobile* e a maior e mais relevante plataforma de aulas particulares do Brasil. O Profes possui milhares de professores cadastrados e que atuam em diversas áreas do conhecimento, como: Ciência da Computação, Matemática, Física, Química, Português, Idiomas, Música e Instrumentos, dentre outros. Nessa aplicação, o aluno pode procurar um tema sobre o qual deseja ter aula e diversos professores capacitados irão aparecer como resultado da busca. O problema dessa aplicação é que existe um ranking de professores, dificultando o encontro entre professores novos na plataforma e clientes ([PROFES, 2022](#)).

2.2.6 Uber

A Uber (Figura ??) é a maior plataforma de motoristas por aplicativo. Seu objetivo é conectar motoristas a clientes que precisam se locomover, isso é realizado através de um aplicativo *mobile*. Existe ainda o Uber Flash, que é o serviço de entregas de objetos e mercadorias oferecido pela Uber. O AUTÔNOMOS também garante essas funcionalidades, no entanto, ao contrário da Uber, o motorista deverá decidir o valor da corrida e não será cobrado taxas sobre esse valor ([UBER, 2022](#)).

2.2.7 iFood

O iFood (Figura ??) é uma aplicação móvel e web construída no intuito de oferecer serviços de delivery e aproximar clientes, restaurantes e entregadores de forma simples e eficaz, proporcionando uma boa experiência de entrega para os seus usuários. A plataforma permite o cadastro das empresas ou estabelecimentos responsáveis por oferecerem os produtos, dos clientes que solicitam algum pedido, e dos entregadores ([IFOOD, 2022](#)).

3 Desenvolvimento

Esse capítulo aborda os detalhes do desenvolvimento do aplicativo AUTÔNOMOS e é dividido em 3 partes: Especificação de Requisitos, Protótipos e Fluxos de Telas e Implementação da Aplicação.

3.1 Especificação de Requisitos

3.1.1 Missão do Software

Sabe-se que a maioria da população brasileira possui acesso à Internet e utiliza smartphones para acessá-la. Além disso, existem mais de 25 milhões de trabalhadores autônomos no país, os quais enfrentam incertezas e riscos em suas atividades econômicas. Diante disso, a missão do software é ser uma alternativa para o aumento da renda dos trabalhadores autônomos brasileiros.

3.1.2 Levantamento de Requisitos

Nessa seção é realizado o levantamento de requisitos, ou seja, são definidos os serviços, funcionalidades e restrições que a aplicação terá. Neste trabalho os requisitos são divididos em requisitos funcionais e não funcionais.

3.1.2.1 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades e serviços providos pelo aplicativo. É mostrado na tabela 3.1 os requisitos funcionais que o aplicativo AUTÔNOMOS terá:

Código	Requisito Funcional	Descrição
RF01	Autenticação	O usuário deve ser capaz de fazer login no aplicativo, tanto como cliente quanto como um trabalhador autônomo, via SMS.
RF02	Cadastro	O usuário deve ser capaz de realizar o cadastro no aplicativo como cliente e como autônomo.
RF03	Cadastro de Atuações	O autônomo deve ser capaz de escolher as áreas de interesses nas quais deseja atuar.
RF04	Solicitação de Serviço	O usuário logado como cliente deve ser capaz de publicar uma proposta de serviço dentre as opções disponíveis na plataforma, a qual ficará ativa até o cliente fechar o serviço ou então após ter passado uma data especificada pelo cliente. Também deve ser possível que o cliente: escolha quantos autônomos podem entrar em contato com ele a partir de uma proposta, descreva sobre o serviço, e determine se o serviço será online ou presencial.
RF05	Visualização de Proposta	O usuário logado como autônomo deve ser capaz de visualizar as propostas disponíveis, assim como as seguintes características de cada uma: <ul style="list-style-type: none"> - Área da proposta. - Nome do cliente solicitante. - Quantidade de autônomos que podem desbloquear a proposta. - Quantidade de autônomos que desbloquearam a proposta. - Tipo de serviço: online ou presencial. - A descrição sobre o serviço. - Distância do cliente.
RF06	Filtrar Propostas	O usuário logado como autônomo deve ser capaz de filtrar as ofertas de acordo com uma área de interesse, tipo de serviço (online e presencial), proximidade do cliente, ou então serviços solicitados recentemente.
RF07	Desbloquear o Contato	O usuário logado como autônomo, pode escolher uma proposta dentre as disponíveis e desbloquear o chat com o cliente em questão, de maneira gratuita, dentro da plataforma.
RF08	Serviços Disponíveis	Os tipos de serviços que estão disponíveis na plataforma, para os profissionais atuarem, são: Professor, Desenvolvedor, Pintor, Diarista, Pedreiro, Montador de Móveis, Técnico em Manutenção, Engenheiro, Arquiteto, Médico, Motorista e Trabalhador Voluntário.
RF09	Notificação	O aplicativo deve notificar, pela barra de notificações, o profissional logado, indicando que há novos clientes que podem ser atendidos.

Tabela 3.1 – Requisitos Funcionais.

3.1.2.2 Requisitos Não Funcionais

Os requisitos não funcionais descrevem as restrições da aplicação, os quais são apresentados pela tabela 3.2 abaixo:

Código	Requisito Não Funcionais	Descrição
RNF01	Plataforma	O aplicativo deve ser utilizado em dispositivos móveis com o sistema operacional Android e iOS.
RNF02	Integridade	De acordo com A Lei Geral de Proteção de Dados Pessoais (LGPD), Lei nº 13.709, de 14 de agosto de 2018, os dados fornecidos pelo usuário devem estar protegidos.
RNF03	Banco de Dados	O aplicativo deverá utilizar o MongoDB para armazenar os dados dos usuários com segurança.
RNF04	Arquitetura	O <i>front-end</i> do aplicativo (telas, UI e UX) deve ser feito utilizando JavaScript juntamente da biblioteca React Native, já o <i>back-end</i> , conexão com o bancos de dados e criação das rotas e dos <i>endpoints</i> , deve ser feito utilizando o Node.js.
RNF05	Autenticação	A autenticação via SMS deve ser implementada utilizando API da Twilio no Node.js.
RNF06	Desempenho e Escalabilidade	Deve ser utilizado o protocolo de comunicação WebSocket para que a comunicação entre cliente e servidor seja feita de forma síncrona, garantindo a eficiência e escalabilidade, principalmente no chat dentro da aplicação.
RNF07	Usabilidade	A interface deve ser simples e intuitiva para que os usuários tenham facilidade em usar o aplicativo.

Tabela 3.2 – Requisitos Não Funcionais.

3.2 Protótipos e Fluxos de Telas

A técnica de prototipagem permite que a concepção do produto seja avaliada sem que a aplicação final seja produzida. Portanto, a prototipagem garante uma visão mais concreta do que está sendo implementado, deixando mais perceptível as características e funcionalidades que o produto final terá (COUTINHO, 2006). Para o desenvolvimento do protótipo foi utilizado o Figma, software online para design de interfaces.

Nessa seção será apresentado o protótipo de alta fidelidade construído juntamente do fluxo das telas, visando destacar como devem ser implementados os requisitos funcionais e as telas da aplicação na etapa de codificação.

3.2.1 Logomarca do Autônomos

Como visto na seção de Trabalhos Relacionados, os softwares correlatos possuem uma logomarca, além disso uma logomarca faz parte da construção visual de uma empresa, sendo importante para criar um bom relacionamento entre a instituição e o cliente (MOTA, 2019). Portanto, se dá como necessário a criação de uma logomarca para o AUTÔNOMOS, a qual foi construída utilizando o Canva e é apresentada na figura 3.1.



Figura 3.1 – Logomarca do aplicativo AUTÔNOMOS.

Fonte: Criado pelo Autor.

3.2.2 Visão Geral do Protótipo

Dado os requisitos funcionais descritos e o auxílio da ferramenta Figma, foi possível construir o protótipo de alta fidelidade, o qual apresenta as telas e os fluxos das mesmas baseado nas interações do usuário com os componentes.

3.2.2.1 Tela de Carregamento

A figura 3.2 apresenta a tela de carregamento da aplicação, a qual possui a logomarca do aplicativo e um ícone indicando que o software está iniciando.



Figura 3.2 – Tela de carregamento da aplicação.

Fonte: Criado pelo Autor.

3.2.2.2 Tela Inicial

Após a aplicação ser completamente inicializada, o usuário é redirecionado para a tela de login que é encarregada de autenticar os usuários e fornecer o direcionamento para tela de cadastro, de acordo com a figura 3.3.

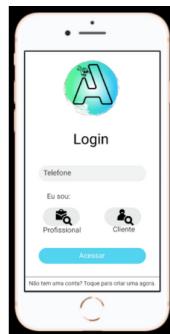


Figura 3.3 – Tela inicial da aplicação.

Fonte: Criado pelo Autor.

Para realizar o login o usuário deve preencher o campo de telefone e escolher, apertando um botão, se irá logar como um cliente que necessita de um profissional ou como um profissional autônomo que atenderá clientes, e, se o telefone estiver cadastrado na plataforma, seja qual for o usuário, ele receberá um SMS e será direcionado para uma tela onde, para efetuar o login, o utilizador deverá inserir o código recebido, tal como ilustrado na figura 3.4.

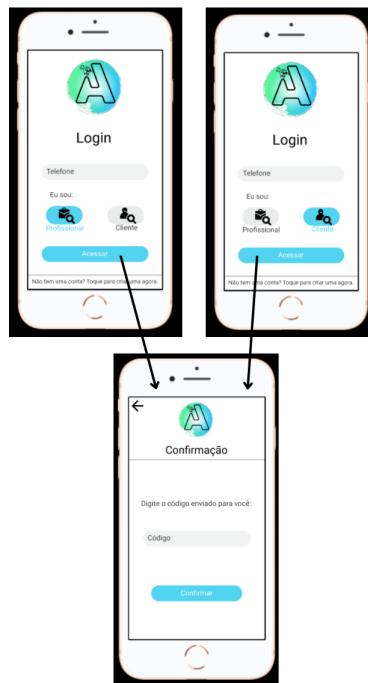


Figura 3.4 – Logando no aplicativo AUTÔNOMOS.

Fonte: Criado pelo Autor.

3.2.2.3 Tela de Cadastro

A partir da tela inicial o usuário poderá clicar no botão do canto inferior para ser redirecionado para a tela de cadastro, onde será capaz de se cadastrar na plataforma após preencher os dados corretamente, como pode-se ver na imagem da figura 3.5.

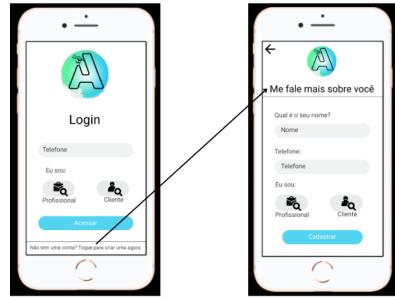


Figura 3.5 – Acessando a tela de cadastro.

Fonte: Criado pelo Autor.

Após o usuário preencher os campos de nome e telefone, além de escolher se deseja cadastrar como profissional ou como cliente, ele deverá inserir o código enviado ao telefone

cadastrado, via SMS, para confirmar o cadastro e fazer login, como pode ser verificado na figura 3.6.

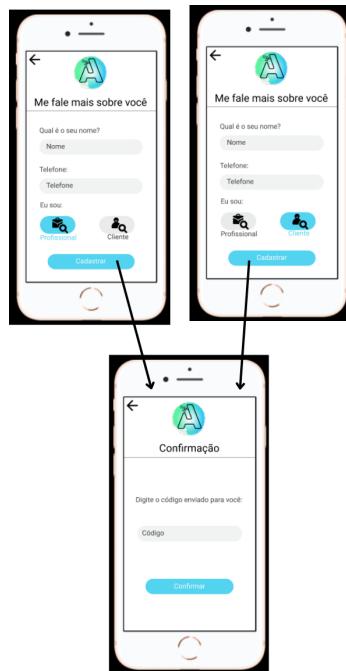


Figura 3.6 – Cadastrando usuário na plataforma.

Fonte: Criado pelo Autor.

3.2.2.4 Tela Inicial do Cliente

Acessando o aplicativo como cliente, o usuário poderá: solicitar um serviço de um profissional da plataforma, verificar serviços solicitados e acessar as conversas com profissionais que entraram em contato, de acordo com a representação da figura 3.7.



Figura 3.7 – Tela inicial do cliente.

Fonte: Criado pelo Autor.

3.2.2.5 Solicitando Serviços

O usuário logado como cliente poderá solicitar um serviço de um profissional da plataforma. Para isso deve-se: escolher o tipo do serviço requerido, como, por exemplo, o serviço de pintura, digitar a quantidade de profissionais que poderão entrar em contato, definir se o serviço será presencial ou online, selecionar a data limite de disponibilidade do serviço na plataforma através de um calendário e descrever os detalhes do serviço solicitado, conforme a imagem da figura 3.8.

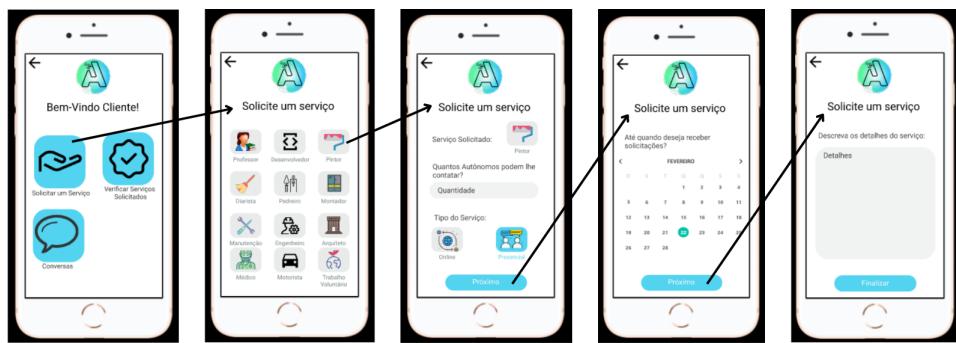


Figura 3.8 – Solicitando um serviço de pintura.

Fonte: Criado pelo Autor.

3.2.2.6 Verificando Serviços Solicitados

O usuário logado como cliente poderá verificar as características dos serviços solicitados e remover algum, caso seja necessário, pressionando o botão de excluir serviço, assim como é apresentado na figura 3.9.

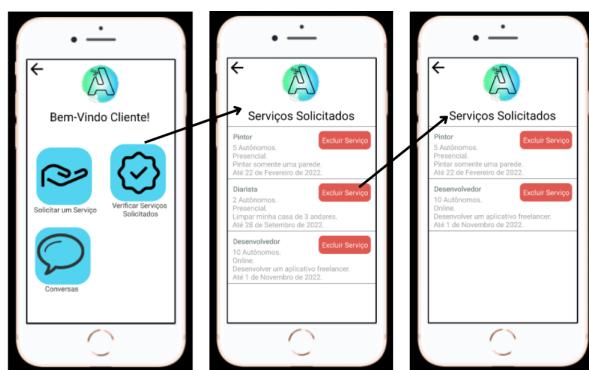


Figura 3.9 – Removendo um serviço e verificando serviços solicitados.

Fonte: Criado pelo Autor.

3.2.2.7 Conversas com autônomos

Após um serviço solicitado por um cliente ser observado por um autônomo, e o profissional ter desbloqueado o chat com o cliente, ambos podem se comunicar através da plataforma e deletar o contato um do outro, tal como demonstrado na figura 3.10.

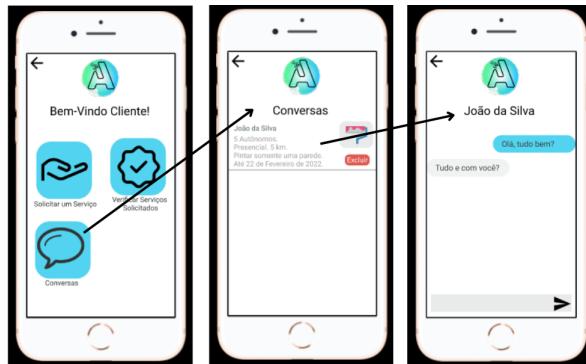


Figura 3.10 – Conversando com um pintor autônomo dentro da plataforma.

Fonte: Criado pelo Autor.

3.2.2.8 Tela Inicial do Autônomo

Acessando o aplicativo como autônomo, o usuário poderá: indicar as áreas em que atua, verificar serviços disponíveis em relação a área de atuação e entrar em contato com clientes através de um chat dentro da plataforma, em consonância com a figura 3.11.



Figura 3.11 – Tela inicial do autônomo.

Fonte: Criado pelo Autor.

3.2.2.9 Escolhendo Áreas de Atuação

O usuário logado como autônomo poderá escolher as áreas em que atua para que, futuramente, receba notificações de clientes das áreas escolhidas. Para isso, ele deve clicar nos botões relativos à área de atuação e clicar novamente caso deseje desmarcar, de acordo com o que é mostrado na figura 3.12).

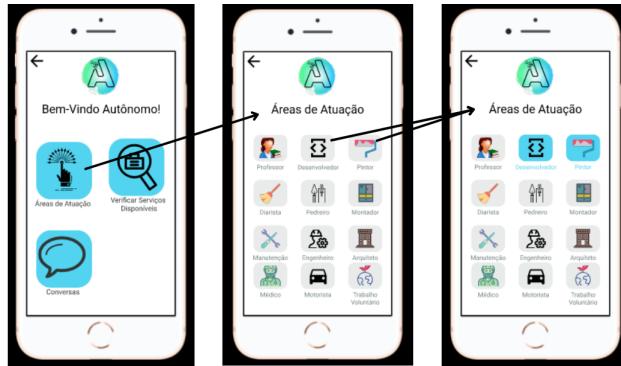


Figura 3.12 – Usuário autônomo indicando que atuará como Desenvolvedor e Pintor dentro da plataforma.

Fonte: Criado pelo Autor.

3.2.2.10 Verificando Serviços Disponíveis

Assim como mostrado na figura 3.13, o usuário logado como autônomo poderá verificar os serviços disponíveis.

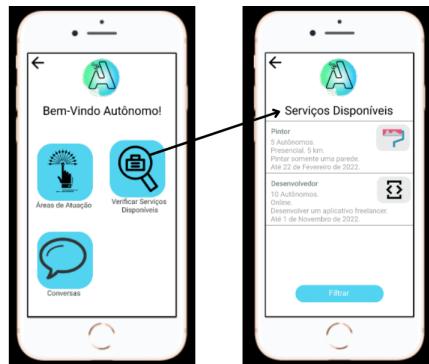


Figura 3.13 – Tela de Verificação de Serviços Disponíveis.

Fonte: Criado pelo Autor.

Além disso, o profissional poderá filtrar os serviços disponíveis pelo tipo, como: online, presencial, mais recentes, mais próximos, e áreas de atuação dentro das escolhidas, conforme ilustra-se na figura 3.14.

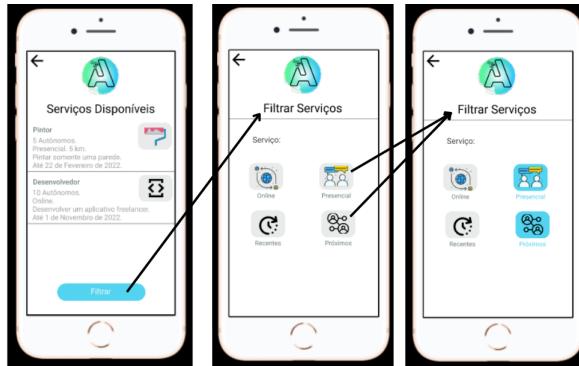


Figura 3.14 – Filtrando serviços de pintura presenciais e mais próximos.

Fonte: Criado pelo Autor.

O autônomo também pode visualizar os detalhes de uma proposta e desbloquear o chat com o cliente em questão, de acordo com a ilustração da figura 3.15.

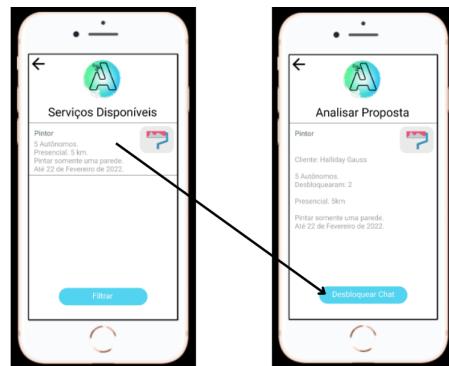


Figura 3.15 – Desbloqueando o contato de um cliente para um serviço de pintura.

Fonte: Criado pelo Autor.

Ainda na tela de verificação de serviços disponíveis para um profissional, é possível deslizar para baixo, atualizando a tela, e isso resultará no aparecimento de novos serviços caso estejam disponíveis, seguindo a figura 3.16.

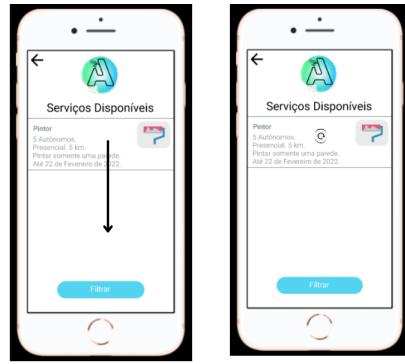


Figura 3.16 – Atualizando a tela de serviços disponíveis para um autônomo.

Fonte: Criado pelo Autor.

3.2.2.11 Conversas com Clientes

Após um profissional liberar o bate-papo com o cliente, ambos podem se comunicar através de um chat dentro da plataforma, conforme a representação gráfica da figura 3.17.

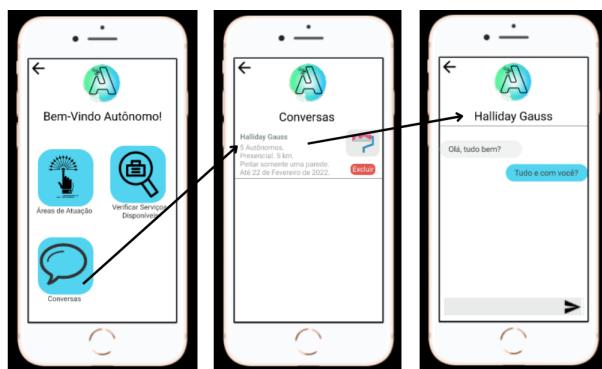


Figura 3.17 – Conversando com um cliente dentro da plataforma.

Fonte: Criado pelo Autor.

3.2.2.12 Tela de Notificações

A figura 3.18 apresenta a tela de notificações para o usuário autônomo, a qual indica que há novos clientes que podem ser atendidos.

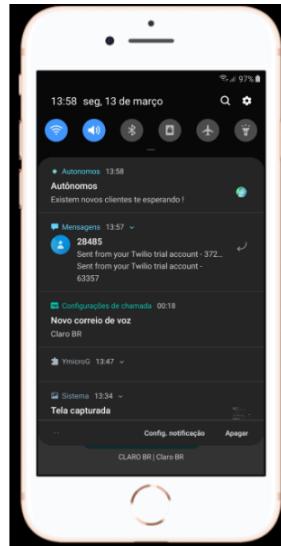


Figura 3.18 – Tela de notificações.

Fonte: Criado pelo Autor.

3.3 Implementação da Aplicação

Após a realização da prototipação e fluxo das telas, a fase da implementação do aplicativo foi iniciada, seguindo rigorosamente o protótipo, visando concluir a missão do software por meio da implementação dos requisitos funcionais e não funcionais. O desenvolvimento da aplicação foi dividido em duas partes: *front-end* e *back-end*, de acordo com a figura 3.19.

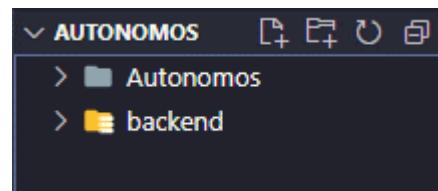


Figura 3.19 – Diretório geral da aplicação contendo dois outros diretórios: "Autonomos"(*front-end*) e "backend".

Fonte: Criado pelo Autor.

3.3.1 Front-end do Autônomos

Utilizando JavaScript juntamente da biblioteca React Native foi construído o *front-end* da aplicação, ou seja, todas as telas e o fluxo das mesmas foram codificadas baseando-se no protótipo produzido no Figma. A escolha dessa biblioteca se dá ao fato de que a partir de um único código construído em React Native é possível gerar um aplicativo para Android e iOS, garantindo a realização dos requisitos não funcionais de "Plataforma" e de "Arquitetura". O

requisito não funcional de "Usabilidade" foi atestado por meio do *feedback* de 5 pessoas que utilizam aplicativos de serviços diariamente e testaram o aplicativo AUTÔNOMOS. A figura 3.20 exibe a organização geral do projeto.

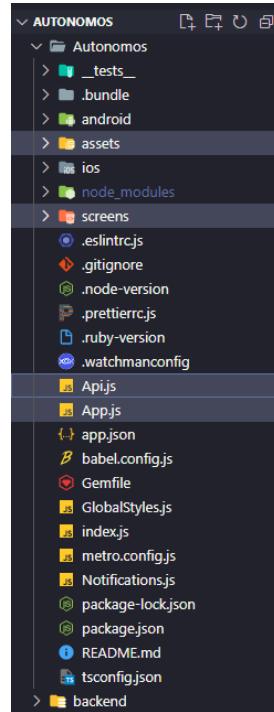


Figura 3.20 – Organização geral do *front-end* do projeto.

Fonte: Criado pelo Autor.

Dentro do subdiretório "*assets*" estão as imagens utilizadas na aplicação e um subdiretório chamado "*fonts*", o qual possui as fontes que foram empregadas, conforme figura 3.21.

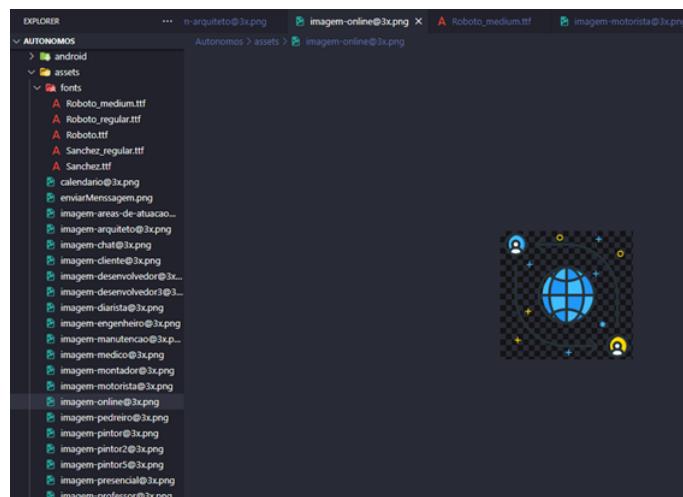


Figura 3.21 – Subdiretório "*assets*" e ícone que representa um serviço online.

Fonte: Criado pelo Autor.

Ainda no *front-end*, localiza-se o diretório "screens" que contém os códigos em JavaScript para cada tela desenvolvida utilizando a biblioteca React Native, em acordo com a imagem da figura 3.22.

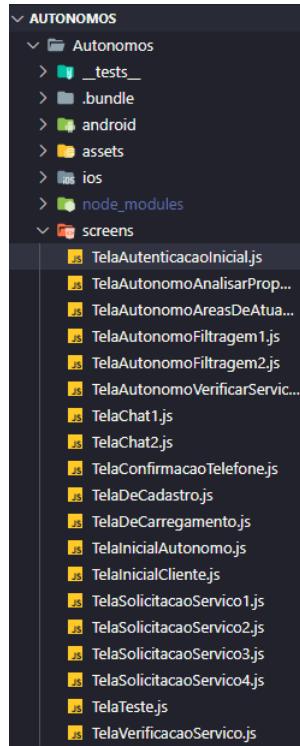


Figura 3.22 – Diretório "screens" dentro do *front-end*.

Fonte: Criado pelo Autor.

Cada arquivo JavaScript referente a codificação de uma tela possui as importações dos componentes gráficos necessários para construí-la e contém um componente que possui um nome e diversas características como: o recebimento de parâmetros de outras telas, o controle da interação com o usuário (incluindo requisições ao servidor), a criação dos elementos gráficos e suas estilizações.

A figura 3.23 apresenta a interface gráfica da tela inicial do cliente e a transição para tela de solicitação de serviços caso seja clicado sobre o botão de solicitar serviços. Também é apresentado: as importações no código para a construção da interface, o recebimento do telefone do cliente ao chegar na tela inicial, a lógica de transição de tela caso seja clicado no botão supracitado, a montagem de alguns componentes da interface gráfica e os estilos aplicados a eles.

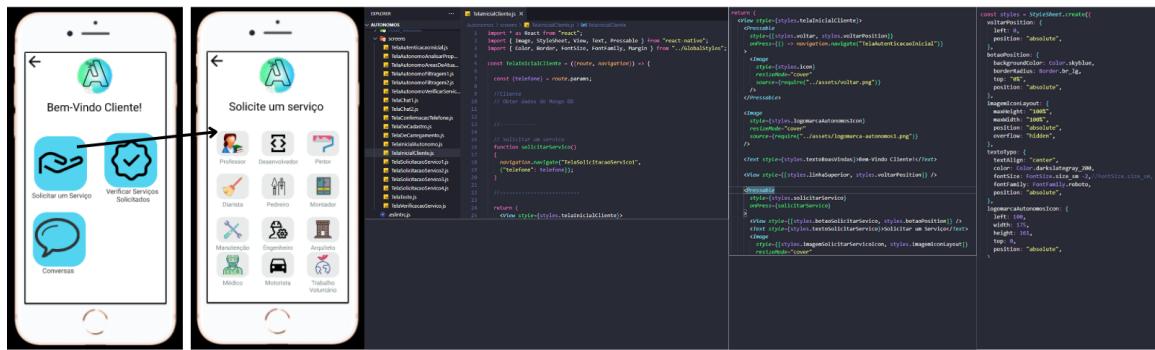


Figura 3.23 – Tela inicial do cliente: interface gráfica e código.

Fonte: Criado pelo Autor.

No arquivo "App.js" todas as telas são importadas e é criado uma ligação entre elas e a aplicação principal, ao mesmo tempo que é definido a tela inicial do aplicativo, como mostrado na figura 3.24.

```
import OneSignal from "onesignal-react-native-sdk";
import TelaChat2 from "./screens/TelaChat2";
import TelaAutonomoVerificarServico from "./screens/TelaAutonomoVerificarServico";
import TelaAutonomoAnalisarProposta from "./screens/TelaAutonomoAnalisarProposta";
import TelaAutonomoMensagem from "./screens/TelaAutonomoMensagem";
import TelaAutonomoMensagem2 from "./screens/TelaAutonomoMensagem2";
import TelaTeste from "./screens/TelaTeste";

// OneSignal -> 93a988c6-7990-46e1-a420-42caf9cb547e
// Código App Inicial
const Stack = createStackNavigator();

const App = () => {
  const [hideSplashScreen, setHideSplashScreen] = React.useState(true);
  //Initial Routename = TelaInicialAutonomo
  return (
    <NavigationContainer>
      {hideSplashScreen ? (
        <Stack.Navigator screenOptions={{ headerShown: false }}>
          <Stack.Screen
            name="TelaDeCarregamento"
            component={TelaDeCarregamento}
            options={{ headerShown: false }}
          />
          <Stack.Screen
            name="TelaAutenticacaoInicial"
            component={TelaAutenticacaoInicial}
            options={{ headerShown: false }}
          />
          <Stack.Screen
            name="TelaCadastro"
            component={TelaCadastro}
            options={{ headerShown: false }}
          />
        </Stack.Navigator>
      ) : (
        <Stack.Navigator screenOptions={{ headerShown: false }}>
          <Stack.Screen
            name="TelaInicialAutonomo"
            component={TelaInicialAutonomo}
            options={{ headerShown: false }}
          />
          <Stack.Screen
            name="TelaTeste"
            component={TelaTeste}
            options={{ headerShown: false }}
          />
        </Stack.Navigator>
      )}
    </NavigationContainer>
  );
};

export default App;
```

Figura 3.24 – Código principal da aplicação.

Fonte: Criado pelo Autor.

Assim como é mostrado na figura 3.25, o arquivo "Api.js" exporta a API pronta para fazer requisições ao servidor.

```

import axios from "axios";
export const ApiUrl = "http://192.168.2.6:3000"
export const Api = axios.create({
  baseURL: "http://192.168.2.6:3000"
})

```

Figura 3.25 – API utilizada no *front-end* para fazer requisições ao servidor.

Fonte: Criado pelo Autor.

3.3.2 Back-end do Autônomos

A partir da utilização do Node.js, ambiente de execução JavaScript *server-side*, foi construído o *back-end* da aplicação, garantido os requisitos não funcionais de Integridade e de Banco de Dados ao utilizar o MongoDB para o armazenamento e segurança dos dados dos usuários. Foi escolhido o Node.js pois além de possuir suporte ao banco de dados MongoDB, que é baseado em objetos JSON (JavaScript Object Notation), também se manterá na mesma linguagem que o *front-end*, permitindo um maior reuso de código. A figura 3.26 apresenta a organização geral do *back-end* do projeto.

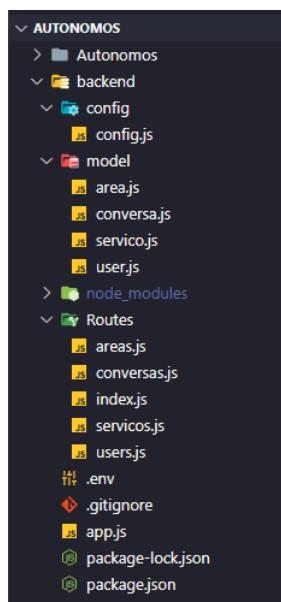
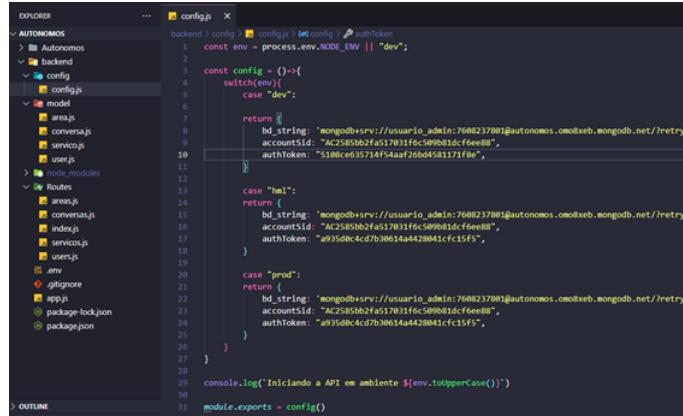


Figura 3.26 – Organização geral do *back-end* do projeto.

Fonte: Criado pelo Autor.

O subdiretório "config" abriga um arquivo JavaScript de configurações do *back-end*, o qual contém a *string* de conexão com o MongoDB e os dados necessários para se autenticar na plataforma de envio de SMS, a Twilio, de acordo com a figura 3.27.



```

const env = process.env.NODE_ENV || "dev";

const config = () => {
  switch(env){
    case "dev":
      return {
        bd_string: "mongodb://usuario_admin:7608237801@autonomos.omoixeb.mongodb.net/?retryWrites=true&ssl=true",
        accountSid: "AC2585bb2fa557011fc509981dcfcfeeb8",
        auth-token: "5108ce635714f54aaef26bd4581171f8e",
      }
    case "uat":
      return {
        bd_string: "mongodb://usuario_admin:7608237801@autonomos.omoixeb.mongodb.net/?retryWrites=true&ssl=true",
        accountSid: "AC2585bb2fa557011fc509981dcfcfeeb8",
        auth-token: "a935db4cd7b306514a4428041cf15f5",
      }
    case "prod":
      return {
        bd_string: "mongodb://usuario_admin:7608237801@autonomos.omoixeb.mongodb.net/?retryWrites=true&ssl=true",
        accountSid: "AC2585bb2fa557011fc509981dcfcfeeb8",
        auth-token: "a935db4cd7b306514a4428041cf15f5",
      }
  }
}

console.log(`Iniciando a API em ambiente ${env.toUpperCase()}`)

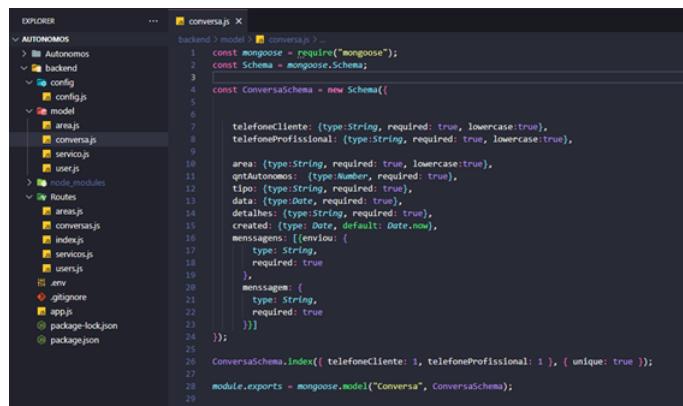
module.exports = config()

```

Figura 3.27 – Arquivo de configurações do *back-end*.

Fonte: Criado pelo Autor.

Já o subdiretório "model" contém os arquivos JS para criar *schemas* no banco de dados MongoDB. A figura 3.28 mostra a criação do esquema de conversas, o qual possui diversos atributos como: telefone do cliente, telefone do profissional, o vetor de mensagens trocadas entre eles, entre outros. Apresenta também a criação de uma chave primária, para o *schema*, composta pelos atributos relativos ao telefone do cliente e o telefone do profissional.



```

const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const ConversaSchema = new Schema({
  telefoneCliente: {type:String, required: true, lowercase:true},
  telefoneProfissional: {type:String, required: true, lowercase:true},
  area: {type:String, required: true, lowercase:true},
  qntAutonomos: {type:Number, required: true},
  data: {type:Date, required: true},
  detalhes: {type:String, required: true},
  created: {type: Date, default: Date.now},
  messages: [{enviou: {
    type: String,
    required: true
  },
    message: {
      type: String,
      required: true
    }
  }]
});

ConversaSchema.index({ telefoneCliente: 1, telefoneProfissional: 1 }, { unique: true });

module.exports = mongoose.model("Conversa", ConversaSchema);

```

Figura 3.28 – Criação do *schema* de conversas do *back-end*.

Fonte: Criado pelo Autor.

No intuito de atender ao requisito não funcional de Arquitetura, dentro do diretório "Routes" foi feito a criação das rotas e dos *endpoints* para que o *front-end* consiga se comunicar com o *back-end*, sendo capaz de passar dados que possam ser inseridos, removidos ou atualizados no banco de dados, como pode ser observado na figura 3.29.



The screenshot shows a code editor with the file `conversas.js` open. The file contains a Node.js application using Express to handle client requests. It includes imports for `express`, `Conversas`, and a `router.post` callback. The `router.post` function checks if a telephone number is provided in the request body. If it is, it sends an error response. Otherwise, it attempts to find a conversation by the provided telephone number. If found, it returns the conversation; if not found, it sends an error response.

```
const express = require("express");
const router = express.Router();
const Conversas = require("../model/conversa");

router.post("/cliente", async (req, res) => {
  const {telephone} = req.body;
  if(!telephone) {
    return res.status(400).send({error: "Erro no leitura das conversas! Não foi passado o número de telefone"});
  }
  try {
    const conversas = await Conversas.find({telephoneCliente: telephone});
    return res.send(conversas);
  } catch (err) {
    return res.status(400).send({error: "Erro na consulta de conversas!"});
  }
})
```

Figura 3.29 – Estabelecendo rotas e *endpoints* das conversas, e utilizando o *endpoint* "/cliente" para obter conversas de um cliente consultando o banco de dados.

Fonte: Criado pelo Autor.

Assim com é apresentado na figura 3.30, no arquivo "App.js" diversas tarefas são realizadas, como: importações das bibliotecas, criação do WebSocket, estabelecimento de conexão com a Twilio e com o MongoDB, indexação das rotas e inicialização do servidor.

```
backend/index.js

1 const express = require('express'); // conectar no express
2 const app = express(); // configurar rotas
3 const config = require('./config/config'); // importar arquivos de configurações
4 const mongoose = require('mongoose'); // importar mongoose para trabalhar com mongo db
5 const bodyParser = require('body-parser'); // importar body-parser para trabalhar com envio de objetos via requisição
6 const jsonParser = bodyParser.json();

7 const socket = require('socket.io');
8 const http = require('http');
9 const server = http.createServer(app);
10 const io = new socket.Server(server);

11 const twilio = require('twilio');

12 const client = new Twilio(config.accountId, config.authToken);

13 const url = config.bd_string; // url de conexão

14 //Configurações do Mongo
15 mongoose.set('strictQuery', true);
16
17 mongoose.connect(url);

18 mongoose.connection.on('error', (err) => {
19   console.log(`Erro na conexão com o banco de dados: ${err}`);
20 });
21
22 mongoose.connection.on('disconnected', () => {
23   console.log(`Aplicação desconectada do banco de dados!`);
24 });
25
26 mongoose.connection.on('connected', () => {
27   console.log(`Aplicação conectada ao banco de dados!`);
28 });

app.js

13 mongoose.connection.on("connected", () => {
14   console.log("Aplicação conectada ao banco de dados!");
15 });
16
17 // Configurando porta da aplicação
18 app.use(bodyParser.urlencoded({extended:false}));
19 app.use(bodyParser.json());
20 app.use(primo);
21
22 const indexRoute = require("./routes/index");
23 const userRoute = require("./routes/user");
24 const serviceRoute = require("./routes/service");
25 const conversaRoute = require("./routes/conversation");
26 const areaRoute = require("./routes/area");

27 // Adicionando rotas à aplicação
28 app.use("/", indexRoute);
29 app.use("/users", userRoute);
30 app.use("/services", serviceRoute);
31 app.use("/conversations", conversaRoute);
32 app.use("/areas", areaRoute);

33 // Subindo um servidor com express
34 const port = 3000;
35 const server = app.listen(port, () => {
36   const message =
37     `process.env.PROD_EM == 'true' ?` + '\n'
38     `  "Server is running in the production environment."` + '\n'
39     `  : "Server is running on https://localhost:${port}";`;
40   console.log(message);
41 });


```

Figura 3.30 – Código principal do *back-end*.

Fonte: Criado pelo Autor.

O requisito não funcional de Autenticação foi garantido através do uso da API da Twilio. A figura 3.31 apresenta o código que permite acessar a plataforma da Twilio e a rota estabelecida para que seja enviado o código de verificação via SMS para o telefone recebido na requisição.

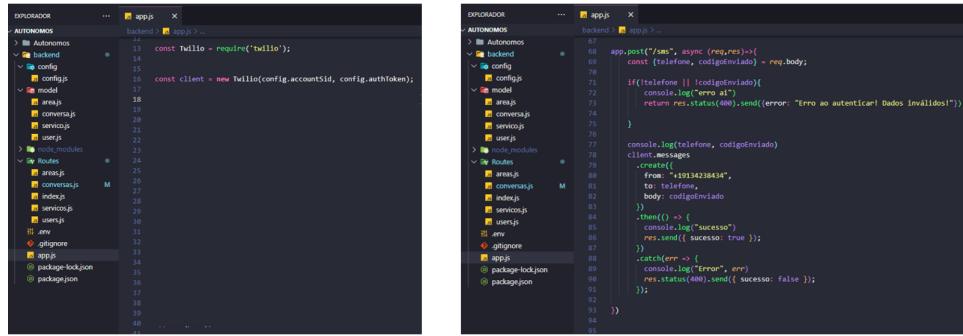


Figura 3.31 – Utilizando a API da Twilio para enviar código de verificação via SMS.

Fonte: Criado pelo Autor.

Finalmente, foi implantado, no *back-end* do projeto, o protocolo de comunicação WebSocket para que a comunicação entre cliente e servidor seja feita de forma síncrona dentro do chat da aplicação. Para isso, foi criado um servidor *http* na aplicação, e um WebSocket (*socket.io*) sobre esse servidor, como representado na figura 3.32.

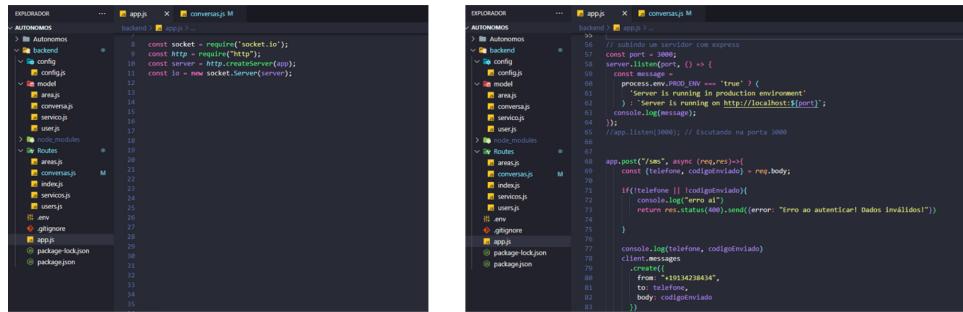


Figura 3.32 – Criando um servidor *http*, vinculando o mesmo a aplicação e a um WebSocket, e iniciando o servidor.

Fonte: Criado pelo Autor.

Seguidamente, o WebSocket emite um sinal com o identificador (id) da conversa quando chega uma mensagem nova, o que é mais eficiente do que verificar a todo instante se o objeto da conversa foi atualizado no banco de dados, e, como no *front-end* o usuário ficará esperando um sinal, o aplicativo pode atualizar a tela somente quando essa mensagem nova chegar, portanto, a implementação do WebSocket garante desempenho e escalabilidade, tal como demonstrado na figura 3.33.

```

MENSAGENS
...
  router.post("/add", async (req,res)=>{
    const {id, envio, mensagem} = req.body;
    if(id || envio || mensagem) {
      return res.status(400).send(error: "Erro ao adicionar mensagem ! Dados invalidos !");
    }
    try {
      const conversas = await Conversas.find({id});
      if(conversas.length > 0) {
        let id;
        let envio;
        let mensagem;
        const [id, envio, mensagem] = conversas[0].mensagens;
        envio = envio ? envio : '';
        mensagem = mensagem ? mensagem : '';
        io.emit(id, mensagem);
        return res.status(200).send(error: "Sucesso ao enviar mensagem !");
      }
    }
    module.exports = router;
  });

AUTONOMOS
...
  React.useEffect(()=>{
    iniciarConversas();
    const socket = io(`ws://${apiUrl}`, { transports: ["websocket"] });
    socket.on("connect", ()=>{iniciarConversas()});
    socket.on("disconnect", ()=>{socket.disconnect();});
    socket.removeAllListeners();
  }, []);
}

```

Figura 3.33 – Código para emitir um sinal ao chegar uma mensagem nova, e aguardar um sinal na aplicação.

Fonte: Criado pelo Autor.

4 Resultados

O software em desenvolvimento é um aplicativo *mobile* chamado AUTÔNOMOS que funciona como uma alternativa para que os trabalhadores autônomos brasileiros, que têm acesso à internet e utilizam dispositivos móveis, possam aumentar suas rendas por meio de propostas de serviços em suas áreas de atuação.

Para que o objetivo do aplicativo fosse alcançado foi feita uma análise crítica dos maiores trabalhos correlatos no mercado visando extrair as características que mais beneficiam o trabalhador autônomo e levantar novas funcionalidades não previstas nos mesmos, levando a especificação de requisitos, seguido da missão do software e levantamento de requisitos funcionais e não funcionais.

As tecnologias para o desenvolvimento foram escolhidas, posteriormente, foi realizado a construção da logomarca e do protótipo e fluxo das telas, resultando na codificação e consolidação do *front-end* e *back-end* da aplicação, a qual seguiu rigorosamente o protótipo, além de cumprir a missão do software e garantir a implantação de todos os requisitos funcionais e não funcionais. A figura 4.1 mostra o aplicativo construído em execução em dois celulares com sistemas operacionais Android e iOS, respectivamente.

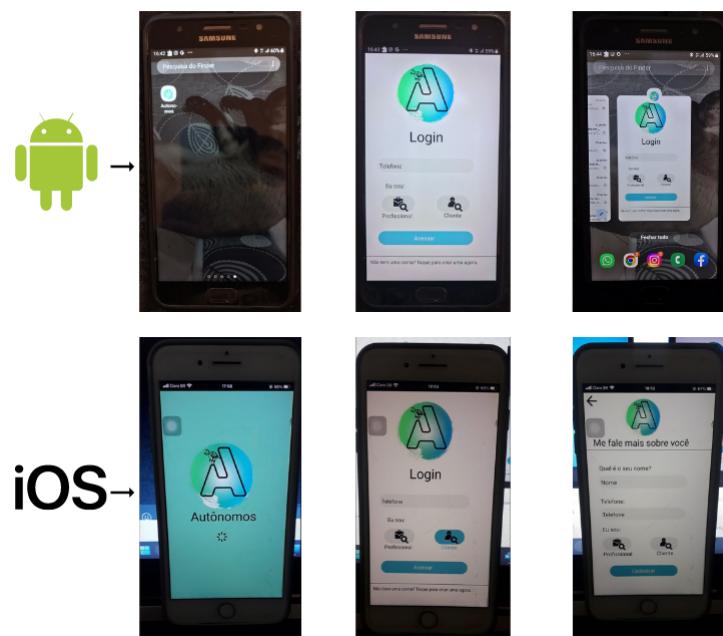


Figura 4.1 – Aplicativo AUTÔNOMOS em execução no Android e iOS.

Fonte: Criado pelo Autor.

A figura 4.2 apresenta o protótipo feito no Figma e o repositório do projeto no GitHub.

Os links para acessar o protótipo e o repositório com todo o código desenvolvido seguem em ordem:

- Link para o protótipo no Figma:
[<https://www.figma.com/community/file/1210321967693751191>.](https://www.figma.com/community/file/1210321967693751191)

- Link para o repositório no GitHub: [<https://github.com/gandalfgauss/autonomos>.](https://github.com/gandalfgauss/autonomos)

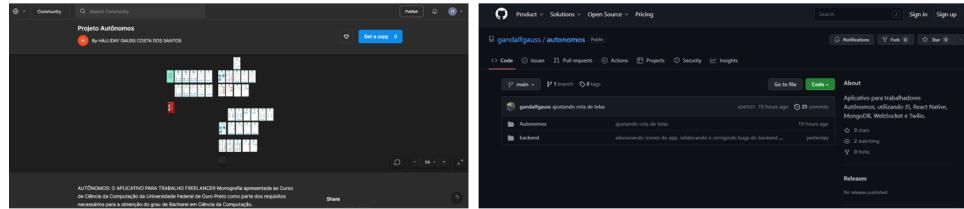


Figura 4.2 – Protótipo na comunidade do Figma e repositório da aplicação no GitHub.

Fonte: Criado pelo Autor.

5 Considerações Finais

5.1 Conclusão

Este trabalho propôs a criação de um aplicativo para dispositivos móveis, Android e iOS, intitulado AUTÔNOMOS, com a finalidade de melhorar a renda dos trabalhadores autônomos no Brasil, os quais enfrentam flutuações em suas atividades econômicas e incertezas quanto aos ganhos.

Analizando os trabalhos relacionados e aplicativos mais utilizados no mercado que possuem o mesmo propósito, possibilitou o levantamento das características desses softwares que mais beneficiam os trabalhadores autônomos.

Após a análise supracitada, avaliou-se a implantação de algumas funcionalidades ausentes nos trabalhos correlatos que são possivelmente vantajosas para os clientes e trabalhadores autônomos, seguindo do levantamento de requisitos, o desenvolvimento do protótipo de alta fidelidade e a construção do aplicativo.

A partir dos resultados apresentados, conclui-se que o aplicativo móvel AUTÔNOMOS vale-se de uma fonte de renda alternativa para trabalhadores autônomos brasileiros, pois o software implementado permite de maneira gratuita que o autônomo, na sua área de atuação, obtenha contatos de clientes que necessitam de um trabalhador para executar um determinado serviço.

5.2 Trabalhos Futuros

Visando aumentar a quantidade de usuários que utilizam o aplicativo desenvolvido, pretende-se acrescentar, dentro do software, novos serviços, como: dentista e personal trainer, e especialidades de determinadas áreas, por exemplo: professor de matemática, português, ciências, dentre outras no âmbito da educação.

No intuito de aumentar a conexão entre clientes e autônomos, propõe-se criar uma versão do AUTÔNOMOS em uma aplicação web utilizando JavaScript, HTML, CSS e React.

Para deixar o aplicativo rentável é planejado a utilização de anúncios dentro da plataforma no período gratuito que durará 90 dias para o profissional autônomo. Após esse período, será cobrada uma mensalidade do profissional para continuar usando os recursos da plataforma.

Como último trabalho futuro, é programado o desenvolvimento de testes para melhorar a consistência e segurança do aplicativo, implantação de novas forma de login, tais como: Google e Facebook, e do protocolo OpenID Connect, publicar a aplicação nas lojas de aplicativos: Play Store e App Store, e levantar os gastos para escalar a aplicação em crescimento.

Referências

- 99FRELAS. *Contrate os melhores freelancers do Brasil | 99Freelas*. 2022. Disponível em: <<https://www.99freelas.com.br/>>. (Acessado em: 27/08/2022).
- ANGÉLICA, M. *10 sites para você encontrar professores particulares*. 2017. Disponível em: <<https://canaldoensino.com.br/blog/10-sites-para-voce-encontrar-professores-particulares>>. (Acessado em: 03/09/2022).
- BITENCOURT, L. P. O impacto da pandemia nos contratos de trabalho: Efeitos sobre os empregados e empregadores. Pontifícia Universidade Católica de Goiás, 2021.
- CANVA. *Início - Canva*. 2022. Disponível em: <<https://www.canva.com/>>. (Acessado em: 24/09/2022).
- CAVALCANTE, L. *Do WhatsApp ao Uber: 1 em cada 5 trabalhadores usa apps para ter renda - 12/05/2021 - UOL Economia*. 2021. Disponível em: <<https://economia.uol.com.br/noticias/redacao/2021/05/12/do-whatsapp-ao-uber-1-em-cada-5-brasileiros-usa-apps-para-ter-renda.htm>>. (Acessado em: 20/08/2022).
- CONTEÚDO, E. *Apps como Uber e iFood se tornam "maior empregador" do Brasil | Exame*. 2019. Disponível em: <<https://exame.com/economia/apps-como-uber-e-ifood-sao-fonte-de-renda-de-quase-4-milhoes-de-pessoas>>. (Acessado em: 03/09/2022).
- COUTINHO, J. R. T. de S. *Prototipagem Rápida como Forma de Envolvimento de Usuário em Metodologia Ágil de Desenvolvimento de Software*. Tese (Doutorado) — Tese de Mestrado. Universidade Federal de Pernambuco. 18, 19, 2006.
- DIGITALHOUSE. *Back-end: o que é, para que serve e como aprender?* 2019. Disponível em: <https://www.digitalhouse.com.br/blog/back-end-o-que-e-para-que-serve-e-como-aprender/?utm_source=blog&utm_medium=social&utm_campaign=awareness&utm_term=clusterprogramacao&utm_content=b2c-clusterprogramacao-linkbotao-blog-front-end-o-que-e-para-que-serve-e-como-aprender>. (Acessado em: 14/08/2022).
- DIGITALHOUSE. *Front-end: o que é, para que serve e como aprender?* 2019. Disponível em: <<https://www.digitalhouse.com.br/blog/front-end-o-que-e-para-que-serve-e-como-aprender/>>. (Acessado em: 14/08/2022).
- DINIZ, A. P. S. M.; VARELA, M. d. G. A. Doutor, por que sou trabalhador autônomo? *Revista Eletrônica do Tribunal Regional do Trabalho da Bahia*.
- FERNANDES, H. M. *O que é um desenvolvedor frontend e o que ele faz?* 2020. Disponível em: <<https://marquesfernandes.com/tecnologia/o-que-e-um-desenvolvedor-frontend-e-o-que-ele-faz/>>. (Acessado em: 14/08/2022).
- FETTE, I.; MELNIKOV, A. *The websocket protocol*. [S.l.], 2011.

FGV. *Brasil tem 424 milhões de dispositivos digitais em uso, revela a 31ª Pesquisa Anual do FGVCia | Portal FGV*. 2020. Disponível em: <<https://portal.fgv.br/noticias/brasil-tem-424-milhoes-dispositivos-digitais-uso-revela-31a-pesquisa-anual-fgvcia>>. (Acessado em: 30/09/2022).

FILHO, P. d. S.; CASTIONI, R. Smartphones no processo educacional: Propondo possibilidades. Programa de Pós-Graduação em Informática na Educação, do Centro ..., 2021.

FIVERR. *Fiverr - Marketplace de serviços freelance*. 2022. Disponível em: <<https://br.fiverr.com/>>. (Acessado em: 27/08/2022).

FLATSCHART, F. *HTML 5-Embarque Imediato*. [S.l.]: Brasport, 2011.

GARRETT, F. *O que é Figma? Quatro perguntas sobre como usar o site* | Editores | TechTudo. 2021. Disponível em: <<https://www.techtudo.com.br/listas/2021/06/o-que-e-figma-quatro-perguntas-sobre-como-usar-o-site.ghtml>>. (Acessado em: 14/08/2022).

GETNINJAS. *GetNinjas | Orçamento de Profissionais Confiáveis no GetNinjas.com.br*. 2022. Disponível em: <<https://www.getninjas.com.br/>>. (Acessado em: 27/08/2022).

HOLZMANN, L. O trabalhador por conta própria no brasil. *Revista Paranaense de Desenvolvimento*, Instituto Paranaense de Desenvolvimento Econômico e Social, v. 34, n. 124, p. 119–137, 2013.

IBGE. *PNAD Contínua TIC 2018: Internet chega a 79,1% dos domicílios do país* | Agência de Notícias. 2020. Disponível em: <[https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/27515-pnad-continua-tic-2018-internet-chega-a-79-1-dos-domiciliros-do-pais#:~:text=O%20equipamento%20mais%20usado%20para,48%2C1%25%20desses%20lares.](https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/27515-pnad-continua-tic-2018-internet-chega-a-79-1-dos-domiciliros-do-pais#:~:text=O%20equipamento%20mais%20usado%20para,48%2C1%25%20desses%20lares.>)> (Acessado em: 30/09/2022).

IBGE, E. *Uso de Internet, televisão e celular no Brasil* | Educa | Jovens - IBGE. 2021. Disponível em: <<https://educa.ibge.gov.br/jovens/materias especiais/20787-uso-de-internet-televisione-e-celular-no-brasil.html>>. (Acessado em: 03/09/2022).

IBM. *Autenticação do OpenID Connect (OIDC) - Documentação da IBM*. 2021. Disponível em: <<https://www.ibm.com/docs/pt-br/sva/9.0.5?topic=methods-openid-connect-oidc-authentication>>. (Acessado em: 16/10/2022).

IFOOD. *Delivery de Comida e Mercado - iFood*. 2022. Disponível em: <<https://www.ifood.com.br/>>. (Acessado em: 27/08/2022).

INC, G. *GitHub Documentation*. 2020. Disponível em: <<https://docs.github.com/en>>. (Acessado em: 27/01/2023).

KAWASAKI, T. *Diagrams of All The OpenID Connect Flows* | by Takahiko Kawasaki | Medium. 2017. Disponível em: <<https://darutk.medium.com/diagrams-of-all-the-openid-connect-flows-6968e3990660>>. (Acessado em: 16/10/2022).

LENON. *Node.js - O que é, como funciona e quais as vantagens* | OPUS. 2018. Disponível em: <<https://www.opus-software.com.br/node-js/>>. (Acessado em: 14/08/2022).

MAGALHÃES, A. L. *Principais aplicativos para freelancers* - Canaltech. 2021. Disponível em: <<https://canaltech.com.br/apps/principais-aplicativos-freelancers/>>. (Acessado em: 03/09/2022).

- MOTA, H. *A importância de ter uma logomarca - iZap Softworks*. 2019. Disponível em: <<https://izap.com.br/blog/a-importancia-de-ter-uma-logomarca/#:~:text=Uma%20logomarca%20faz%20parte%20da,mercado%2C%20principalmente%20atrav%C3%A9s%20da%20internet>>. (Acessado em: 24/09/2022).
- NACIONAL, J. *Número de trabalhadores autônomos bate recorde no início de 2022, mas renda cai | Jornal Nacional | G1*. 2022. Disponível em: <<https://g1.globo.com/jornal-nacional/noticia/2022/06/04/numero-de-trabalhadores-autonomos-bate-recorde-no-inicio-de-2022-mas-renda-cai.ghtml>>. (Acessado em: 02/10/2022).
- NAMA, R. *Figma vs Adobe XD vs Sketch: qual a melhor opção?* 2020. Disponível em: <<https://simple.nama.ai/post/figma-vs-adobe-xd-vs-sketch-qual-a-melhor-opcao>>. (Acessado em: 14/08/2022).
- NATIVE, R. *React Native · Learn once, write anywhere*. 2022. Disponível em: <<https://reactnative.dev/>>. (Acessado em: 14/08/2022).
- NINJA, P. *PODEROSÍSSIMO NINJA - PROGRAMA EU FICO LOKO #40 - YouTube*. 2020. Disponível em: <<https://www.youtube.com/watch?v=oEfT4s6Ghk4&t=0s>>. (Acesso em: 06 de Agosto de 2022).
- OKUBO, B. *Você Sabe o que é CSS? Entenda Como Funciona e Para que Serve - Blog*. 2021. Disponível em: <<https://br.godaddy.com/blog/voce-sabe-o-que-e-css-entenda-como-funciona-e-para-que-serve/>>. (Acessado em: 14/08/2022).
- PROFES. *Aulas Particulares - Professor Particular | Profes*. 2022. Disponível em: <<https://profes.com.br/>>. (Acessado em: 27/08/2022).
- PURZ, M. *WhatsApp no Brasil: números atuais e suas oportunidades comerciais*. 2022. Disponível em: <<https://www.messengerpeople.com/pt-br/whatsapp-no-brasil/#:~:text=Hoje%20podemos%20dizer%20que%20o,a%20147%20milh%C3%B3es%20em%202022.>>. (Acessado em: 03/09/2022).
- ROVEDA, U. *JavaScript: o que é, para que serve e como funciona o JS?* 2020. Disponível em: <<https://kenzie.com.br/blog/javascript/>>. (Acessado em: 14/08/2022).
- ROVEDA, U. *React: o que é, como funciona e porque usar e como aprender – Blog Kenzie Academy Brasil – Programação e Tecnologia*. 2020. Disponível em: <<https://kenzie.com.br/blog/react/>>. (Acessado em: 17/09/2022).
- STRAUB, S. C. e B. *Git - Book*. 2009. Disponível em: <<https://git-scm.com/book/en/v2>>. (Acessado em: 27/01/2023).
- SUISSO, F. Trabalho informal no brasil contemporâneo. *Revista eletrônica da faculdade de direito de campos*, 2006.
- TECNOBLOG. *O que é e para que serve o MongoDB? – Aplicativos e Software – Tecnoblog*. 2021. Disponível em: <<https://tecnoblog.net/responde/o-que-e-e-para-que-serve-o-mongodb/>>. (Acessado em: 14/08/2022).

TEIXEIRA, G. *Godoy Teixeira Advogados Associados | Artigos*. 2020. Disponível em: <<https://www.godoyeteixeira.com.br/post/17/voce-sabe-a-diferenca-entre-trabalho-terceirizado-autonomo-e-pejotizacao>>. (Acessado em: 14/08/2022).

TWILIO. *Twilio – APIs de comunicação para SMS, voz, vídeo e autenticação | Twilio*. 2023. Disponível em: <<https://www.twilio.com/pt-br>>. (Acessado em: 22/02/2023).

UBER. *Solicite uma viagem ou cadastre-se como motorista | Uber Brasil*. 2022. Disponível em: <<https://www.uber.com/br/pt-br>>. (Acessado em: 27/08/2022).

WORKANA. *Workana: Contrate Freelancers Talentosos do Brasil*. 2022. Disponível em: <<https://www.workana.com/>>. (Acessado em: 27/08/2022).