RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

# DESENVOLVIMENTO DE SISTEMA DE AUXÍLIO PARA MICRO VENDEDORES ONLINE

Tiago de Freitas Pinheiro
Graduando em Engenharia de software – Uni-FACEF
tiagofreitasp00@gmail.com

Yuri Tiófilo Silva
Graduando em Engenharia de software – Uni-FACEF
dev.yuritiofilo@gmail.com

Ely Fernando do Prado
Mestre em Computação – Uni-FACEF
elyfprado@gmail.com

**Resumo**
O projeto tem como objetivo oferecer um sistema para auxiliar e gerenciar os processos de vendas on-line para micro vendedores, com controles integrados que poderão otimizar as rotinas e os processos. Sendo assim, a proposta deste trabalho é apresentar o desenvolvimento de uma plataforma on-line para controle das informações de forma simples, facilitando e agilizando as operações com custo acessível para o público-alvo, que são os micros vendedores. Plataformas on-line podem ser de grande contribuição no processo e controle de vendas de produtos e serviços. Para este projeto, procurou-se fazer um estudo das necessidades do segmento, criando modelos, bem como um estudo das principais ferramentas para elicitação de requisitos e de atuais recursos voltados para o desenvolvimento de software para os ambientes web. Utilizou-se a linguagem de programação typescript, framework React junto ao NextJS para criação da interface, que foi interligada ao Firebase. Os resultados alcançados possibilitam aos micros vendedores usufruírem de um sistema web que contempla o controle e armazenamento das informações de suas vendas, organizados e apresentados por meio de rotinas integradas e relatórios que permitem interação. A interface do programa demonstra a preocupação dos autores em oferecer um layout amigável e intuitivo. O software ainda pode auxiliar na gestão da logística e do financeiro, com o propósito de oferecer atendimentos com qualidade, agilidade e atender a legislação vigente.

**Palavras-chave**: Desenvolvimento. Micro vendedores. Organização.

**Abstract**
The project aims to offer an ERP system to assist and manage online sales processes for micro sellers, with integrated controls optimizing routines and processes. Therefore, the purpose of this work is to present the development of an online platform to control information in a simple way, facilitating and streamlining operations at an affordable cost for the target audience. Online platforms can be of great contribution in the process and control of sales of products and services. For this project, we sought to study the needs of the segment, creating models, as well as a study of the main tools for elicitation of requirements and current resources for the development of software for web environments. Typescript programming languages, React framework and NextJS

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

were used to create the interface, which was linked to Firebase. The results achieved allow micro sellers to take advantage of a web system that includes the control and storage of their sales information, organized and presented through integrated routines and reports that allow interaction. The program's interface demonstrates the authors' concern to offer a friendly and intuitive layout. The software can also assist in the fulfillment of logistics and finance, with the purpose of offering services with quality, agility and complying with current legislation.

**Keywords**: Development. Micro sellers. Organization.

## 1 Introdução

Muitos vendedores online hoje em dia, possuem diversas dificuldades para gerenciar seu estoque, vendas e financeiro, o que pode ocasionar em perda de informações, vendas de produtos os quais não possuem estoque, dificuldade com fluxo de caixa e o planejamento da empresa. Para melhorar isso, procuram um sistema que supra as suas necessidades em um mesmo lugar.

Existem alguns sistemas que conseguem fazer o que precisam, mas em alguns casos são sistemas complexos ou com funções separadas, em que os micro vendedores não conseguem manter um sistema muito grande e que não vão usar o sistema por inteiro e acontece de usar um ou mais tipo de sistema para conseguir gerenciar de maneira correta a sua empresa. Muitas vezes, precisam ficar processando informações manualmente, de sistema para sistema.

Neste contexto, o trabalho apresentado neste artigo tem o objetivo de desenvolver uma aplicação que atenda às necessidades dos micros vendedores, conseguindo gerenciar toda a sua loja, em uma única aplicação simples, gerenciar seus produtos, estoques, financeiro, clientes, com as principais funcionalidades para isso.

Os principais conceitos utilizados neste artigo foram Engenharia de Software, Análise de requisitos, Metodologia Ágil, Prototipação, React, Typescript e MVP.

No Capítulo 2 são apresentados os temas teóricos relacionados ao tema proposto. No Capítulo 3 é apresentada a metodologia utilizada para o desenvolvimento do projeto. No Capítulo 4 são apresentados os resultados obtidos neste projeto.

## 2 Referencial Teórico

Nesta seção serão apresentadas as tecnologias e ferramentas usadas para o desenvolvimento do projeto de um sistema de auxílio para micro vendedores, tais como: Engenharia de Software, Análise de requisitos, Métodos Ágeis, Prototipação, MVP, React e Typescript.

### 2.1 Engenharia de Software

De acordo com André (2021), com a evolução da tecnologia, veio junto a necessidade de aprimoramento das técnicas, processos e desenvolvimentos de ferramentas voltadas para o tratamento de informações. Resumindo alguns pontos do

trabalho de Lampeão (2014), a Engenharia de software busca conhecimentos através de métodos científicos e adequações de estruturas, processos e dispositivos convertendo em recursos e melhorias, assim atendendo a necessidade constante de inovações.

Segundo Paula Filho (2000), as partes da engenharia de software podem ser definidas como “arte” ou ideia, necessidades, conhecimentos, habilitações, recursos, formas de aplicações, dispositivos, estruturas e processos com base em experimentos e observações convertendo informações em recursos para projetar e desenvolver aplicativos e programas com tecnologias atuais. A "Arte" consiste em colocar em prática uma ideia que necessita da apuração de várias informações e fatores que poderão influenciar na materialização desta ideia. Assim a engenharia de software, através de seus conhecimentos, fará com que esta ideia seja concretizada, projetando programas para serem executados em computadores.

Com conhecimentos científicos e empírico, a partir de estudos, utiliza-se métodos pré-definidos seguindo regras básicas para desenvolvimento de experiências e pesquisas a fim de produzir novos conceitos, corrigir e integrar conhecimentos evidenciando experimentos práticos, obtendo resultados que auxiliam os métodos da ciência para explicar, modelar e generalizar o que a prática descobriu. Parte dos métodos da Engenharia de Software provém da Ciência da Computação. “Na Engenharia de Software, muitas práticas são adotadas porque funcionam, mesmo quando ainda carecem de fundamentação teórica satisfatória” (PAULA FILHO, 2000).

A Engenharia de software, como uma atividade, necessita de habilitações específicas com a finalidade de comprovar que o engenheiro está apto a integrar equipes que irão projetar e desenvolver sistemas para computadores, assim como lidar com as etapas de gerenciamento dos projetos.

Toda engenharia parte de recursos naturais como de pessoas, máquinas, softwares que inter-relacionados tratam informações produzindo o resultado esperado. Na engenharia de software os resultados dos recursos naturais são convertidos em formas adequadas.

O resultado do trabalho da engenharia de software se concretiza em formas adequadas como programas de computadores ou softwares. Na engenharia de software o engenheiro tem infinitas possibilidades de criação de formas, porém nem todas atendem a necessidade e critérios de utilização.

A complexidade de programas de computadores desafia a engenharia de software a escolher e montar estruturas através de conjuntos de dispositivos, extrair funções úteis capazes de satisfazer a necessidade do projeto.

Os processos de Engenharia de Software seguem determinadas normas e métodos baseados na ação sistemática, seguindo passos ordenados e relacionados no projeto, tendo objetivo o desenvolvimento, validação e evolução do software projetado. Dentre estes passos, um dos mais importantes é o de Análise de requisitos [2.2].

## 2.2 Análise de requisitos

A Análise de requisitos consiste em gerar especificações adequadas compreendendo e valorizando as técnicas e estratégia na engenharia de software atendendo às definições e necessidades do projeto (DE SOUZA; SANTANDER, 2011).

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

Para evitar problemas graves e evidentes e tornar a análise de requisitos mais efetiva possível, deve-se levar em conta o entendimento e domínio do contexto envolvido para o desenvolvimento do sistema e a análise e viabilidade da solicitação do cliente auxiliando no projeto do software.

O entendimento e ponto inicial passa pelos relatos das necessidades do solicitante pelo projeto, detentor do conhecimento apontando as especificações, forma mais efetiva possível e descrições de suas necessidades.

A importância de entender a necessidade do solicitante, avalia-se o seu conhecimento sobre o contexto que na maioria das vezes tem uma visão simplificada em relação a sistemas de computadores até comprometendo o resultado esperado, mas o analista de requisitos deve se orientar sobre esta perspectiva.

De acordo com Paula Filho (2000), descrevendo os requisitos funcionais e não-funcionais o analista assume um envolvimento e compromisso maior com o processo e as avaliações do projeto. Pode-se criar maior credibilidade envolvendo o solicitante ao processo, chegando a informações corretas e resultados esperados. Requisitos funcionais, representam os comportamentos que um programa ou sistema deve apresentar diante de certas ações de seus usuários e os não-funcionais quantificam determinados aspectos do comportamento (PAULA FILHO, 2000).

Inicia-se os trabalhos com a apresentação de modelos do segmento para levantamento dos requisitos. O modelo deve contemplar o tipo de software que será desenvolvido, observando o contexto, público-alvo, conteúdo, avaliação e equipe multidisciplinar. O resultado desta análise dos dados e elaboração da modelagem proporciona a ideia de vários modelos para o desenvolvimento de um software. O trabalho de engenharia de Software define o planejamento, pesquisas, capacitação das equipes, delimitação da envergadura do software e modelagem dele, considerando questões extremamente relevantes deste modelo para o desenvolvimento do software (PAULA FILHO, 2000).

### 2.2.1 Técnicas para levantamento de requisitos

Ao identificar a parte interessada e seus diferentes pontos de vista sobre o problema, suas necessidades e influências, cria-se uma visão geral do sistema a ser desenvolvido, permitindo descrever os requisitos em linguagem natural, gerando modelos conceituais e eliminando ambiguidades, inconsistências, omissões e erros, definindo os diagramas a serem usados, para atender às partes interessadas e a outros que farão uso do mesmo sistema (PAULA FILHO, 2000).

(Os requisitos são especificados em termos técnicos, requisitos de sistemas atendendo a necessidade do projeto. Mudando da perspectiva de requisitos de usuário para requisitos de sistema.) Os requisitos do sistema são declarações articuladas de forma clara sobre o que um sistema deve ser capaz de fazer atendendo as necessidades dos requisitos do usuário.

Deverá certificar que ao longo do processo de desenvolvimento do sistema seus requisitos sejam atendidos. Garantir o rastreio de mudanças, analisando o impacto sobre a proposta inicial para evidenciar sua viabilidade técnico-financeira.

Em todo desenvolvimento de software, um aspecto fundamental é a captura dos requisitos dos usuários. Para apoiar este trabalho, diversas técnicas podem ser utilizadas, como amostragem, investigação, entrevistas, planejamento, questionários, observação, prototipação, em seu artigo explica estas técnicas. (JANAINA, 2009)

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

## 2.3 Métodos Ágeis

A metodologia ágil foi definida em meados de 2001. Antes disso, os projetos de software eram baseados no modelo *'Waterfall'*, em que a mudança de requisitos após o planejamento não era aceita, sendo que com isso acarretava a necessidade de se fazer grandes alterações nos documentos e códigos (SILVA, 2010).

Segundo Dos Santos Soares (2004) as metodologias ágeis não possuem nada de novo, mas o que as diferencia das tradicionais, são os enfoques e valores. As metodologias ágeis são definidas como um conjunto de técnicas e práticas, sendo que juntas, pode-se fazer uma entrega mais rápida a partir de processos otimizados e mudanças que possam ser alteradas conforme o projeto. Desta forma, as entregas são realizadas com um ciclo menor e mais bem definido no que vai ser feito.

Os métodos ágeis vêm ganhando espaço e importância na indústria de software, tendo como seu objetivo criar um projeto de altíssima qualidade fazendo com que atendam a necessidade de cada usuário. Ele apresenta uma abordagem rápida para o desenvolvimento de software. Os planos para inicialização do projeto precisam ser bem detalhados, mas os planos futuros podem servir apenas como um rascunho, fazendo com que eles possam mudar e se adaptar ao longo do processo de desenvolvimento.

### 2.3.1 Scrum

O scrum é uma estrutura para ajudar as equipes da empresa a trabalharem juntos. Segundo Silva (2010), o nome vem do jogo de Rugby, que é composto por uma equipe que precisa trabalhar junto para ganhar o jogo seguinte. O scrum estimula a aprender com a experiência, a se organizar quando está resolvendo um problema ou tarefa, e a melhorar nos fracassos que ocorrem.

Pode-se dizer que o scrum é um conjunto de reuniões, ferramentas e cargos que ajuda a equipe a gerenciar o projeto. Como diz a TOTVS (2021) o scrum divide cada projeto em ciclos segmentados podendo ser semanais ou mensais sendo chamado de *sprints*. Embora o scrum seja usado mais pela equipe de desenvolvimento de software, o princípio do scrum pode ser aplicado para diversos tipos de empresas, equipes ou trabalhos, fazendo com que seja tão popular, considerando uma estrutura de gestão de projeto de agilidade.

## 2.4 Prototipação

A prototipação é a representação de um design, que pode ser representada de diversas maneiras, como em uma folha sulfite, um conjunto de telas vinculada por hyperlinks, em fotos, etc. Segundo Sommerville e Sawyer (2003), um protótipo pode ser usado como meio de comunicação entre os diversos membros da equipe de desenvolvimento ou mesmo como meio de testar idéias. Ela é uma fase muito importante para os desenvolvedores do projeto, e podem ser geradas em categorias, como o protótipo de baixa fidelidade, que foca na interação do sistema, em componentes de interface e na estrutura do sistema, mostrando quais são as

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

atividades do sistema proporcionando uma visão mais ampla do que será desenvolvido no protótipo de alta fidelidade, o qual tem o design fielmente parecido com o sistema real, uma imagem real do sistema, fazendo com que crie um estilo padrão do design do sistema mostrando padrões e guia de estilo e o protótipo executável que trabalha na parte de navegação de páginas, mas não gerando a regra de negócio do sistema (ROSEMBERG, 2008).

### 2.4.1 Rabiscoframe

Segundo Sanches da Silva (2010) rabiscoframe ou prototipação em um papel é a criação de forma não tão bonita, mas que seja levado ao cliente e que seja fácil de fazer alterações para ser iniciado o Wireframe [2.4.2].

Os passos para que possa ser implementado no processo de desenvolvimento de software, primeiramente é perder a vergonha caso o desenho não fique bonito, fazendo com que seja apenas alguns rabiscos para que surjam ideias de outras pessoas. Ele quase nunca é feito sozinho e sim em grupos, além disso, o Rabiscoframe não é a entrega final do design ao cliente, servindo para os próprios designers e desenvolvedores entenderem como vai funcionar a interface do projeto.

A expressão vem do termo *sketch* (esboço, rabisco), em alguns times de projeto o Rabiscoframe já se tornou essencial no início, fazendo com que as ideias de design se concretizam. Já algumas pessoas não o implementaram no processo de desenvolvimento de software, seja por não saber desenhar ou por achar que ele não traz uma “versão” final do projeto como o Protótipo de alta fidelidade.

### 2.4.2 Wireframe

Wireframe é um desenho básico demonstrando uma forma simplificada de como o produto deveria funcionar e de acordo com Teixeira (2014), trata-se de uma “mistura bem humorada” do termo sketch.

Na maioria das vezes o wireframe é desenvolvido sem muitas cores vivas, apenas com cores neutras como cinza, preto, branco (DUARTE, 2015). Seu maior objetivo não é mostrar as cores do projeto e sim organizar os elementos que vão entrar no desenvolvimento final do design.

Um processo que demonstra as páginas estruturadas, bem como sua hierarquia e os componentes principais que compõem o seu sistema. Serve também para discutir a ideia inicial do projeto e para informar qual o trabalho e tempo que os designers e desenvolvedores terão que usar para desenvolver todo o projeto (TEIXEIRA, 2014).

Pensando em um processo de construção, para uma casa ser construída ela precisa ter uma planta, com todas as paredes, janelas, porta, onde cada fio de tomada irá passar. Após todo esse planejamento e aprovação do cliente, da construtora, a construção será iniciada, da mesma maneira que é desenvolvido um software. Sendo o wireframe a parte desse planejamento (DUARTE, 2015).

### 2.4.3 Protótipo de alta fidelidade

A prototipação de alta fidelidade utiliza materiais que provavelmente irão estar no produto já desenvolvido. No desenvolvimento do protótipo pode ser usado

algumas ferramentas como o Figma[1] ou Adobe XD[2] . A versão mais elaborada serve também para a criação de alguns testes de usabilidade do produto que está sendo desenvolvido, e se está se encaixando no objetivo do projeto, sendo assim, pode ocorrer algumas alterações, caso quem estiver testando e perceber que o material usado não tem necessidade ou não encaixa na regra de negócio, podendo eliminar esses materiais e por fim economizar tempo e dinheiro investido no desenvolvimento do produto final (STRYCHARZ, 2014) .

Podem ter algumas desvantagens no desenvolvimento do protótipo de alta fidelidade, levam muito tempo para desenvolver o protótipo, ele pode gerar uma alta expectativa no cliente, mas ao mesmo tempo também tem os pontos positivos, como, é útil para a venda de ideias e para os testes de algo técnico, eles oferecem ao usuário uma interação direta com a solução proposta pelo cliente (ROSEMBERG, 2008).

## 2.5 JAVASCRIPT

Lançado em meados de 1995, com seu nome de lançamento Livescript, foi desenvolvido por Brendan Eich, primeiramente para Netscape segundo Duarte (2015) .

Ainda no ano de lançamento, eles queriam se juntar com a Sun Microsystems, a empresa criadora e gerenciadora da linguagem de programação JAVA, querendo aproveitar toda a publicidade da empresa, fazendo com que ela alterasse o nome de Livescript para JAVASCRIPT, gerando algumas confusões por se basear no nome da linguagem JAVA, tendo impressões de que era baseado nela, mas isso não é verdade (DUARTE, 2015) .

De acordo com Brandão (2015) o Javascript teve bastante sucesso na sua versão 1.0 e o Netscape lançou uma nova versão, a 1.1 para o navegador “Netscape navigator 3”. Sendo assim, a Microsoft decidiu fazer uma implementação superior na internet Explorer e implementando o Javascript fazendo com que mudasse o nome para Jscript (DUARTE, 2015).

A linguagem de programação Javascript permite a criação de aplicações com interação total com o usuário. Com isso existem algumas características, tais como: Linguagem interpretada, não precisa ser compilada pelo navegador, ele executa o código linha a linha quando chega para o navegador, Baseada em protótipo, ela é orientada a objetos, sendo também um processo de cópia de objetos que já existem e servindo como um protótipo; Funções em primeiro lugar, permitindo que o desenvolvedor passe argumentos de função à outra função, conseguindo reservar as variáveis globais; Tipagem fraca, as variáveis do Javascript não precisam ser tipadas como booleano, strings, number, etc., com isso elas podem mudar o tipo delas durante o processo de desenvolvimento. Para a melhoria disso, surgiu o Typescript [2.6], que é uma extensão do Javascript para suprir essa deficiência.

## 2.6 Typescript

[1] https://www.figma.com/

[2] https://www.adobe.com/br/products/xd.html

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

Como citado no final do tópico acima [2.5], o typescript veio para suprir a deficiência do Javascript [2.5] segundo de Oliveira Lopes (2017). Ele é uma ferramenta a qual adiciona as tipagens estáticas ao Javascript, que por padrão tem a sua tipagem dinâmica, e com isso, variáveis e funções podem assumir outros tipos ao decorrer do processo (DE OLIVEIRA LOPES 2017).

O Typescript é usado somente no ambiente de desenvolvimento, o navegador e o node não conseguem ler o typescript, então existe um tipo de *build* para compilar o Typescript em Javascript.

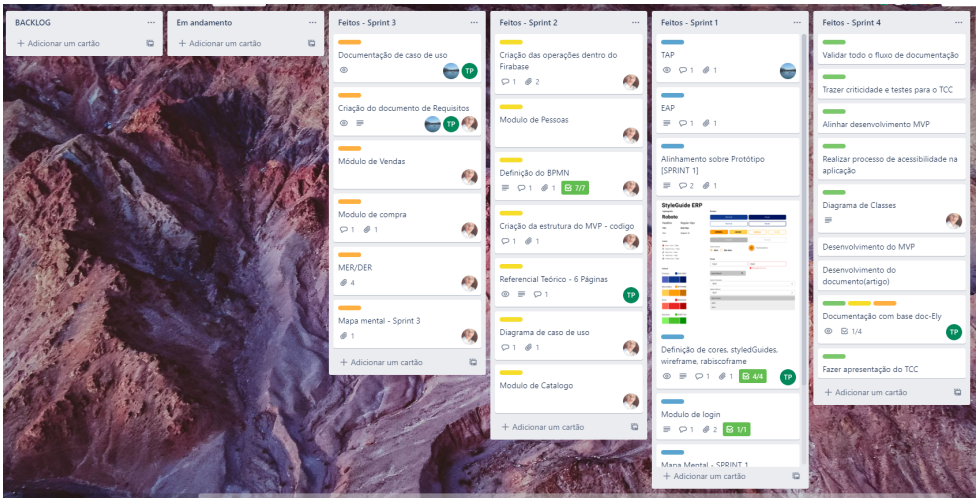
## 3 Desenvolvimento do Projeto

Neste capítulo, é apresentado como o sistema de forma geral foi desenvolvido, desde a necessidade real de um micro vendedor online partindo desta necessidade elaboramos este projeto desde toda a engenharia de artefatos e requisitos, mas também como foi implementado o back-end e front-end explicando as principais bibliotecas utilizadas em cada etapa, mostrando seu funcionamento, apresentando as linhas de códigos e em seguida mostrando os resultados. Também é mostrado como ficou toda a parte de versionamento do GitHub.

### 3.1 Metodologia ágil

Como todo projeto precisa de um cronograma para ser desenvolvido de maneira correta, usamos o scrum para definir tarefas a serem desenvolvidas, utilizamos a ferramenta Trello para separar nossas tarefas em Sprints, backlogs, tarefas em andamentos e tarefas já finalizadas.

Figura 1 - Definição de sprints.




Fonte: Elaborado pelos autores

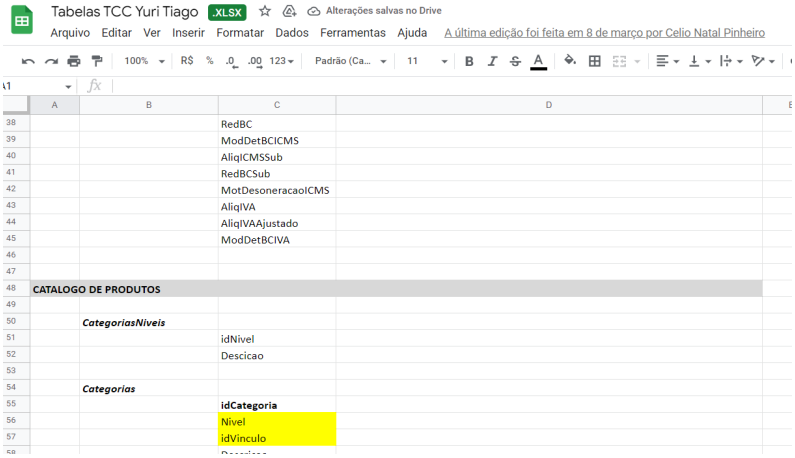
### 3.2 Engenharia de software

Antes de iniciarmos qualquer desenvolvimento de código foi realizado um levantamento de requisitos junto aos stakeholders para entender a real necessidade do sistema e assim juntos idealizarmos as suas funcionalidades. Neste

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

processo realizamos reuniões online sempre tendo como base uma anotação por meio de Excel para assim termos dados necessários, entradas e saídas esperadas como podemos ver na Figura 2.

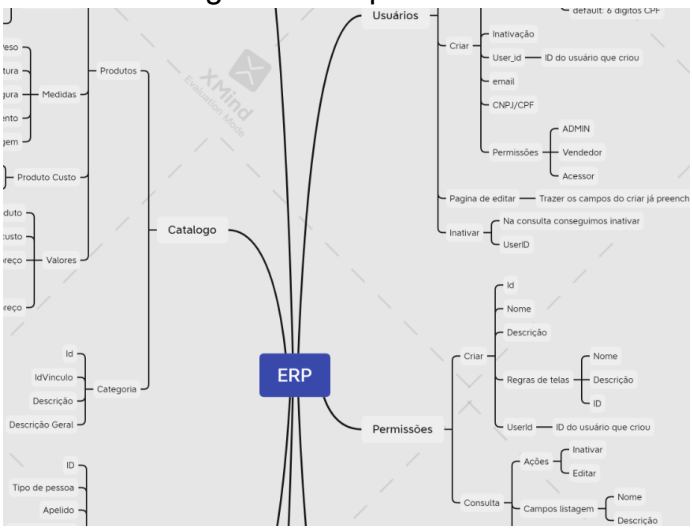
Figura 2 - Levantamento de campos e funções.




Fonte: Elaborado pelos autores

Para visivelmente analisarmos cada funcionalidade optamos por desenvolver um mapa mental que ajuda, junto do cliente, avaliarmos se as funcionalidades estão de acordo e assim conseguirmos enxergar melhor a solução, visando sempre o valor ao nosso cliente. Com isso obtivemos os resultados de mapa mental conforme mostrado parcialmente na Figura 3.

Figura 3 - Mapa mental

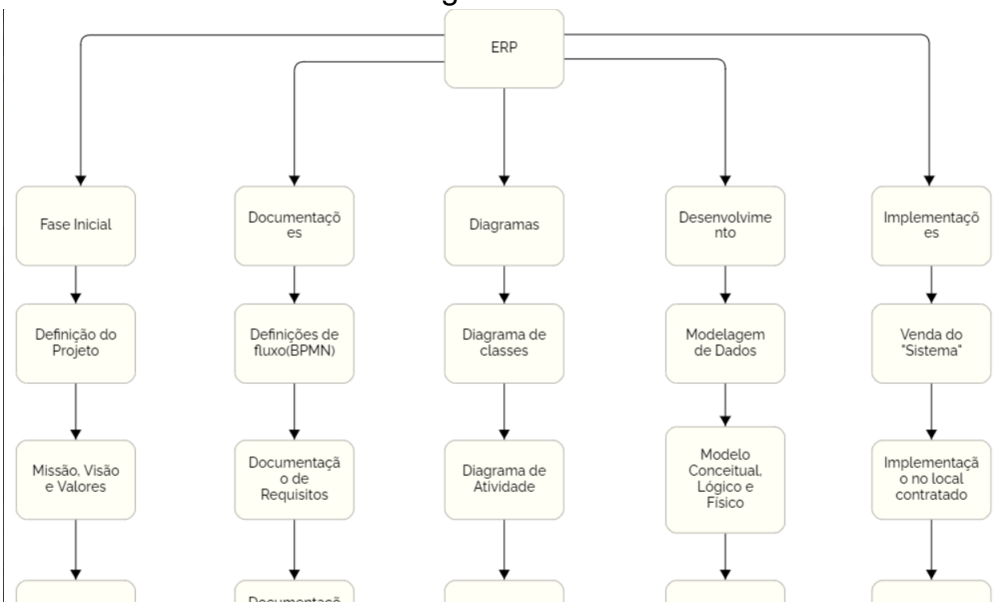



Fonte: Elaborado pelos autores

Para realizarmos toda a engenharia de software deste projeto, primeiro desenvolvemos um TAP (Termo de Abertura do Projeto), onde idealizamos, prazos de entregas de sprints (entregas semanais), definimos o que seria abordado, desde a construção até a entrega de todos os documentos e software. Dentro disso o cliente recebeu este termo, assinou e assim demos início às demais demandas da engenharia de software. Após isso elaboramos um EAP (Estrutura Analítica do

Projeto) definindo o processo que iríamos construir em cada etapa, o qual as suas partes principais podem ser visualizadas na Figura 4.

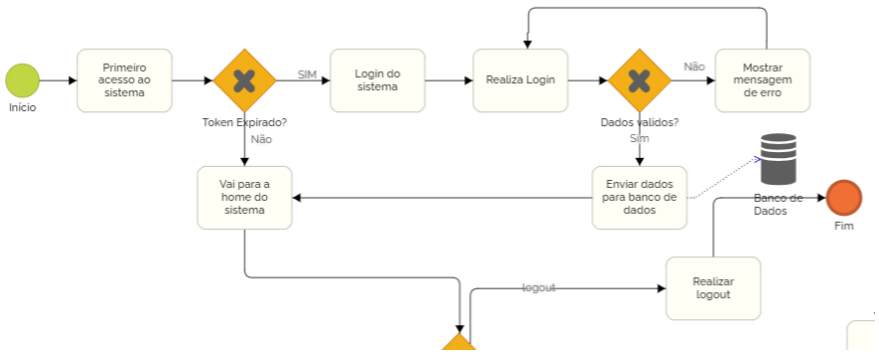
Figura 4 - EAP




Fonte: Elaborado pelos autores

Após isso, desenvolvemos o artefato que direcionou todo o fluxo de processos que haveria dentro do sistema. O BPMN *(Business Process Model and Notation)*, que foi feito com duas raias, que representam os usuários/serviços que iremos utilizar, neste caso temos a raia de usuário (sistema) e a raia de serviço externo de E-commerce. Conforme o fluxo se discorre, temos as tomadas de decisões, que é onde temos toda a regra de negócio. Um exemplo que fica bem claro a definição de regra de negócio é: “Quando o usuário entra no sistema, se o token do usuário está expirado temos uma tomada de decisão para isso, neste caso ele pode ser direcionado para a tela de login para assim refazer todo o fluxo de autenticação na aplicação, como também o token pode não estar expirado e ele ser direcionado para a tela de dashboard”, podemos mostrar isso conforme a Figura 5.

Figura 5 - BPMN - Fluxo de token

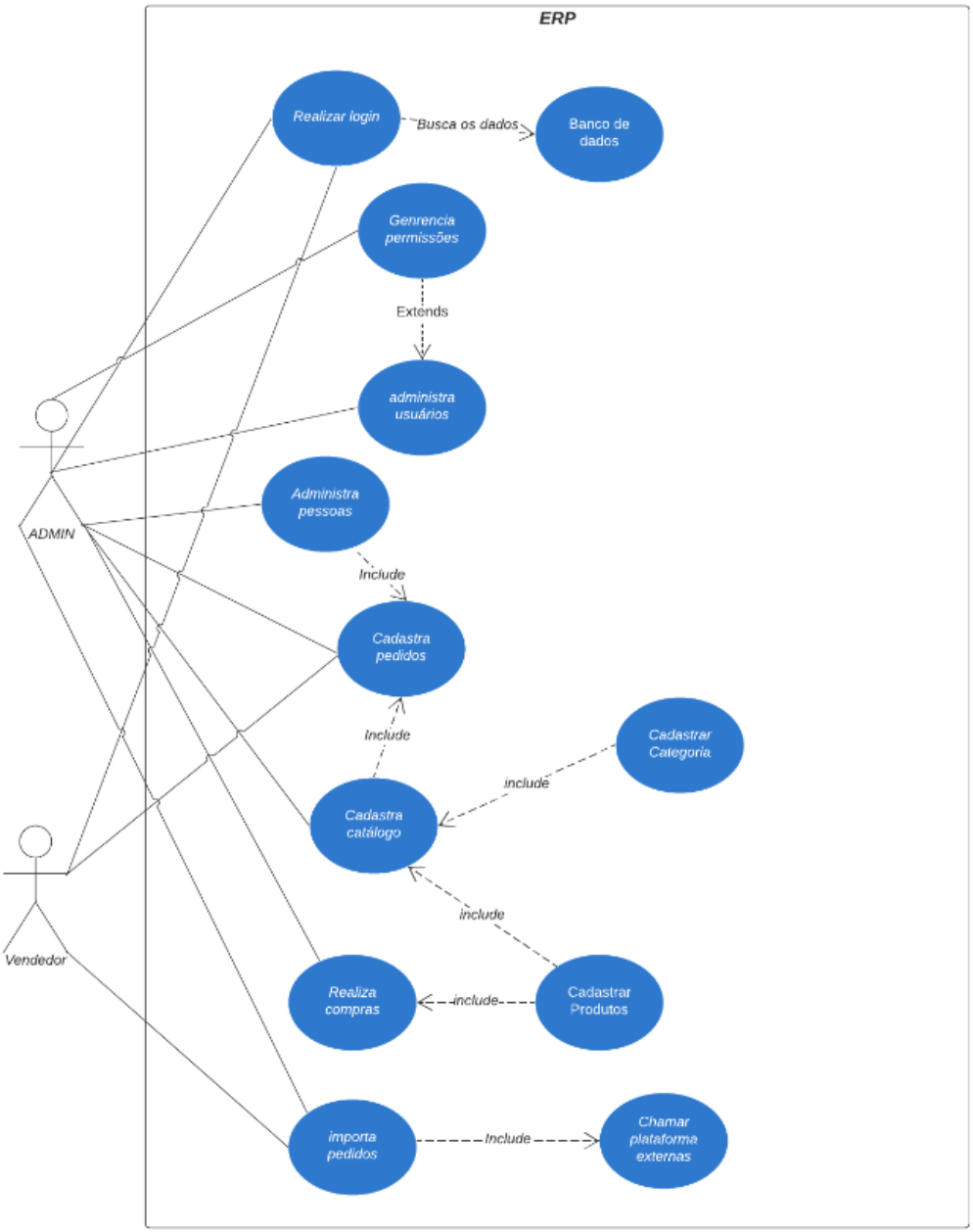



Fonte: Elaborado pelos autores

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

Após realizarmos todos os processos é necessário desenhar isso em um diagrama de caso de uso, para assim entendermos onde o usuário irá ter que realizar ações e onde será apenas intervenções do sistema. Neste caso cada balão do diagrama representa uma funcionalidade existente dentro do BPMN, e as flechas demonstram o que cada usuário pode executar e a ligação de *extends* entre elas. Com isso podemos visualizar o diagrama conforme a Figura 6.

Figura 6: Diagrama de caso de uso




Fonte: Elaborado pelos autores

Além de representar os requisitos de forma gráfica, precisamos também de representar isso por meio de uma documentação, na qual possui ligação direta ao

diagrama, entretanto especificamos de forma aprofundada cada funcionalidade, em que temos entradas e saídas esperadas, um contador único, uma localização, uma criticidade da funcionalidade, um cenário de teste e uma pré-condição. Podemos visualizar uma funcionalidade de login conforme a figura 7.

Figura 7: Documentação do caso de uso

| | |
|---|---|
| Contador | 001 |
| Localização | Tela de login |
| Criticidade | Alta |
| Objeto de teste | Verificar os funcionamentos da tela de login |
| Caso de teste | Testar a funcionalidade login. |
| Pré-condição | 1- O usuário deverá estar cadastrado na base de dados. |
| Procedimento | 0 - Entrar no aplicativo.<br>1- Clicar input de "E-mail".<br>2- Digitar um e-mail válido.<br>2.1- Digitar um email invalido.<br>3- Clicar input de "Senha".<br>4- Digitar uma senha válida.<br>4.1- Não digitar nenhum caractere.<br>5- Clicar no botão "Entrar".<br>5.1 - Clicar no botão "Entrar" com algum campo com erro.<br>6 - Clicar na âncora de "Esqueci minha senha". |
| Resultado esperado | 0 - O sistema deverá validar se existe um login realizado. Redirecionando para a home ou direcionando para login.<br>1- O sistema deverá dar foco no input de "E-mail".<br>2- O sistema deverá verificar se o e-mail possui um @ e aceitar a digitação do e-mail.<br>2.1- O sistema deverá retornar uma mensagem de erro abaixo do input.<br>3- O sistema deverá dar foco no input de "Senha".<br>4- O sistema deverá aceitar a digitação da senha.<br>4.1- O sistema deverá retornar uma mensagem de erro abaixo do input.<br>5- O sistema deverá realizar o login e direcionar para a tela home.<br>5.1 - O sistema deverá abrir um "Toast Error" para notificar o erro.<br>6 - O sistema deverá redirecionar o usuário para a página "Resetar senha". |

Fonte: Elaborado pelos autores

Após todo esse fluxo refinamos todos os requisitos dentro de uma documentação de requisitos que é onde utilizaremos tabela para especificar um ID para cada requisito, uma descrição, uma categoria (evidente ou não evidente), uma descrição do que ele é exatamente e por fim toda a regra envolvida nele, podemos ver um exemplo na Figura 8:

Figura 8: Documentação de requisitos

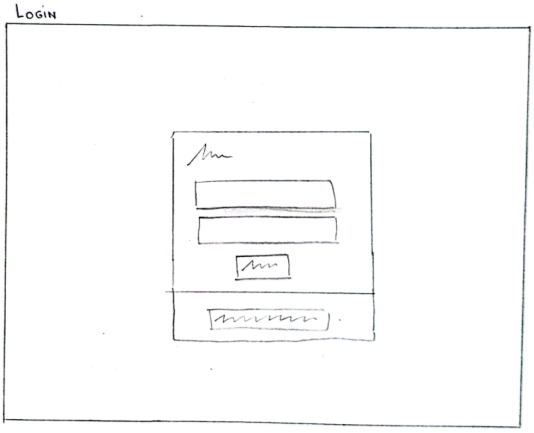
| | |
|---|---|
| ID: RF02 | Nome do requisito: Realizar login do usuário com email e senha. |
| Descrição → | Realizar login no sistema. |
| Categoria: Evidente | Prioridades: Essencial |
| Informações → | O usuário irá realizar login com o email e senha. |
| Regra de negócio → | Verificar se o e-mail está cadastrado no banco de dados, e a senha cadastrada corresponde a mesma que foi inserida. |

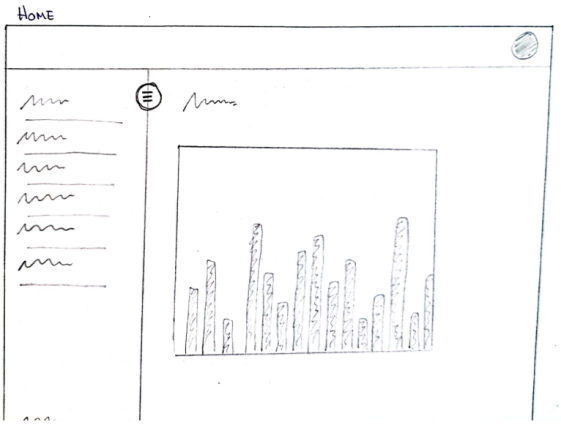
Fonte: Elaborado pelos autores

## 3.3 Protótipos

Para iniciar o desenvolvimento do aplicativo, precisamos desenvolver o protótipo de alta resolução para iniciarmos o desenvolvimento, mas antes disso, optamos por iniciar fazendo um esboço em uma folha sulfite apenas para começarmos a ter uma ideia de como ficaria o aplicativo. Sendo assim, obtivemos os resultados com o rabiscoframe mostrado na Figura 9.

Figura 9 - a)Rabiscoframe da tela de login b)Rabiscoframe da tela de Home

a) 

b) 


Fonte: Elaborado pelos autores

Em seguida, optamos por desenvolver o wireframe, sem termos cores, fontes padronizadas, ícones, para termos uma ideia melhor que o rabiscoframe do aplicativo, com isso, tivemos as seguintes telas criadas conforme pode ser visto na Figura 10:

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

Figura 10 - a)WireFrame da tela de Login b)WireFrame da tela Home

a) 

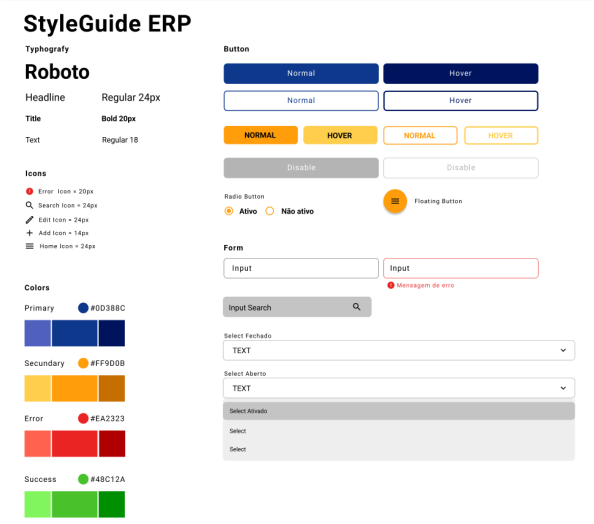
b) 


Fonte: Elaborado pelos autores

Todas essas telas e as demais, foram criadas apenas com a cor cinza, sendo diferenciado o componente apenas com o grau de preto nela.

Após o desenvolvimento do wireframe, partimos para decidir quais as cores que o aplicativo iria ter, colocando-as todas em uma paleta de cor, assim conseguindo ter um padrão no aplicativo por inteiro, componentes criados para a padronização e também selecionando a fonte e os ícones que seria usado durante todo o desenvolvimento da aplicação. Assim, tivemos o seguinte resultado com o styleguide mostrado na Figura 11:

Figura 11 - StyleGuide




Fonte: Elaborado pelos autores

Assim que todas as cores, fontes e componentes foram definidos, começamos a dedicar no protótipo de alta resolução, fazendo com que o mesmo ficaria da seguinte forma apresentada na Figura 12:

Figura 12 - a)Tela de Login b)Tela de Home

a) 

b) 

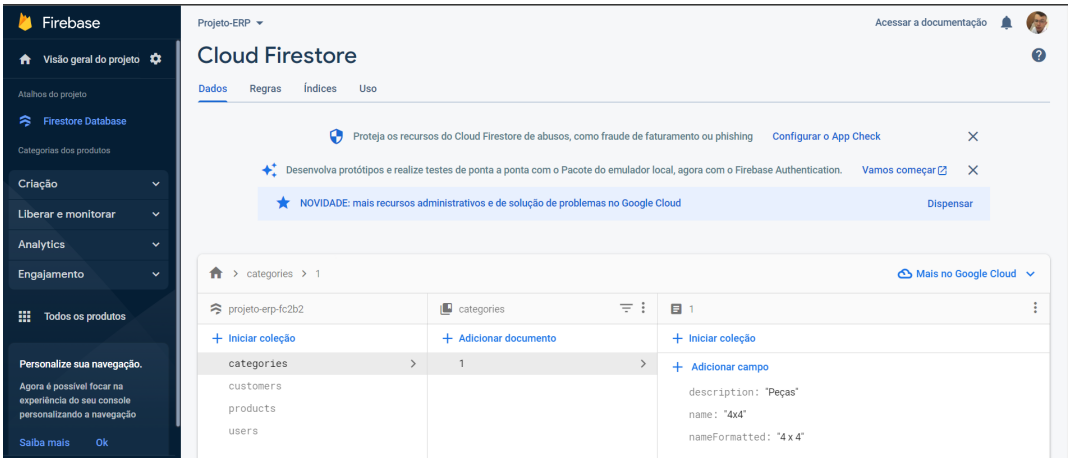

Fonte: Elaborado pelos autores

## 3.4 Back-end

Para criarmos nosso primeiro MVP do projeto, optamos por utilizar um banco de dados não relacional, onde pudéssemos ter uma facilidade para realizar mineração de dados futuramente para extrair possíveis indicadores. Deste modo utilizamos o Firebase, pois além de ser fácil de ser utilizado, possui muitas ferramentas de análise de dados, deste modo criamos a partir do artefato de Diagrama de Classes todas as credenciais e os documentos, já que neste caso este banco de dados é orientado a documentos assim como outros bancos de dados não relacionais.

Para já deixarmos uma estrutura pré configurada, tivemos que criar todos os documentos inserindo campo a campo dentro da plataforma do firebase, assim teremos uma melhor assertividade quando formos desenvolver o front-end da aplicação. Para isso utilizamos o firestone que é um banco de dados dentro do firebase, onde configuramos toda a estrutura do banco de dados orientado a documentos(classes). Conforme a Figura 13, podemos ver uma pré-condição de como os documentos deveriam ficar, o qual realizamos para mostrar um diagrama de classe.

Figura 13 - Estrutura dos documentos no firebase



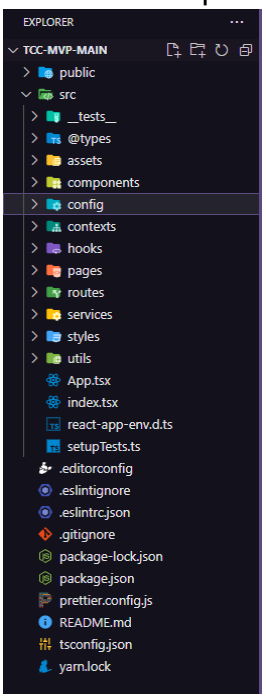

Fonte: Elaborado pelos autores

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

## 3.5 Front-end

Assim que finalizamos o protótipo de alta resolução e algumas regras de negócio, começamos a estruturar o front-end conforme mostrado na figura 14, utilizamos a CLI do NextJS para criar essa estrutura visando que NextJS utiliza também React sendo um facilitador, neste caso apenas uma biblioteca que utiliza React. Ele gera também um servidor que poderíamos utilizar, porém, optamos por não utilizar este servidor no projeto. Os pontos mais importantes a serem comentados são os componentes conforme mostrado na figura 15, onde faz com que o código fique mais fácil de ser usado e dar manutenções, outra parte a ser comentada é a pasta config, onde fica a configuração do firebase seguindo na figura 15. Podemos citar também a utilização do ChakraUI, que é uma biblioteca que já possui diversos componentes prontos, porém poucos estilizados suportando uma reestilização. Ele substitui a utilização de CSS, trazendo maior velocidade no desenvolvimento. Ele possui ainda muitos padrões de acessibilidade para a interação do usuário com seus componentes (CHAKRA, *online*).

Figura 14 - Estrutura completa do projeto




Fonte: Elaborado pelos autores

Figura 15 - Configuração do firebase


Fonte: Elaborado pelos autores

Na configuração do firebase, conforme a figura 15, são importados os módulos do firebase e criado uma constante com as configurações de chave, domínio do projeto, apiKey, nome do projeto, entre outros. E é instanciada uma classe constante app com o initializeApp para fazer o firebase funcionar, o contexto do firebase, a constante analytics que serve para o google analytics funcionar, passando o contexto app do firebase. A constante db recebe o banco de dados diretamente do firestone com o contexto app.

Figura 16 - Configuração do firebase


Fonte: Elaborado pelos autores

Na funcionalidade da imagem 17 é mostrada como é feita a chamada para conseguir realizar chamadas ao firebase e utilizar o mesmo no front-end. No início do código, ele cria uma constante users que recebe o collection, que tem a funcionalidade de fazer uma chamada ao banco de dados que está setado na configuração do firebase, e passamos a referência do collection ao qual quer buscar. Após isso, é feita a busca por todos os documentos que o banco de dados retorna, fazendo com que os dados sejam mapeados, retornando o .data e setando a lista dos usuários. Podemos visualizar as importações do módulo de autenticação do firebase na Figura 18.

Figura 17 - Configuração do firebase

```
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";
```

Fonte: Elaborado pelos autores

Assim que importados começamos a buscar todos os auths que possui dentro do firebase. Usando o signInWithEmailAndPassword, ele faz a verificação do usuário para ver se está cadastrado ou não, caso esteja cadastrado ele entra no sistema. Fazendo com que ele guarde a informações no localStore e no setData, caso ele não esteja cadastrado no sistema, ele retorna no .catch, mostrando o erro para o usuário e não deixando o mesmo fazer login.

Figura 18 - Configuração do firebase

```
const signIn = useCallback(async ({ email, password }) => {
  signInWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
      const { email, uid } = userCredential.user;

      localStorage.setItem('@MvpTCC:user', JSON.stringify({
        email, uid
      }));
      setData({
        user: {
          email, uid
        }
      })
    })
    .catch((error) => {
      const errorCode = error.code;
      const errorMessage = error.message;
      console.log('error', errorCode, errorMessage);
  });
}, [auth]);
```

Fonte: Elaborado pelos autores

## 3.6 Testes

Assim que o front-end foi todo desenvolvido, começamos a fazer testes para não ocorrer nenhum erro durante o uso do sistema. De início fizemos testes de caixa preta na tela de login, seguindo com entradas que podem ser válidas ou inválidas e um resultado que o sistema retornará, conforme mostrado na Figura 19.

Figura 19 - Tabela do teste de caixa preta (login)

| Entrada | | | Resultado esperado |
|---|---|---|---|
| Email | Senha | | |
| | | | Erro: Campos vazio |
| Teste | | | Erro: Email ou senha incorreta |
| teste@gmail.com | teste | | Error: Email ou senha incorreta |
| teste@gmail.com | teste123 | | Login realizado |

Fonte: Elaborado pelos autores

Para finalizar toda a parte de testes realizamos alguns testes de integração aos quais tivemos o teste de login. Testamos quando o usuário visita a página “/” e assim temos o formulário de login, o teste insere os campos de login e direciona ele para o dashboard após o usuário clicar na tecla “enter” no teclado, conforme mostrado na Figura 20.

Figura 20 - Demonstrativo de código utilizado nos testes

```
function loginUsingForm(email: string, password: string) {
  cy.visit("/");

  cy.get("input[name=email]").type(email);
  cy.get("input[name=password]").type(password).type("{enter}");

  cy.location("pathname").should("include", "/profile");
}

describe('signin.cy.ts', () => {
  it('Should be able sign in form', () => {
    loginUsingForm("yuri.dev@gmail.com.br", "123456")
  })
})
```

Fonte: Elaborado pelos autores

## 4 Resultados da solução proposta

Neste capítulo são apresentados todos os resultados conforme as funcionalidades necessárias e um layout padrão em toda a aplicação a fim de mostrar a navegabilidade do sistema e sua utilização.

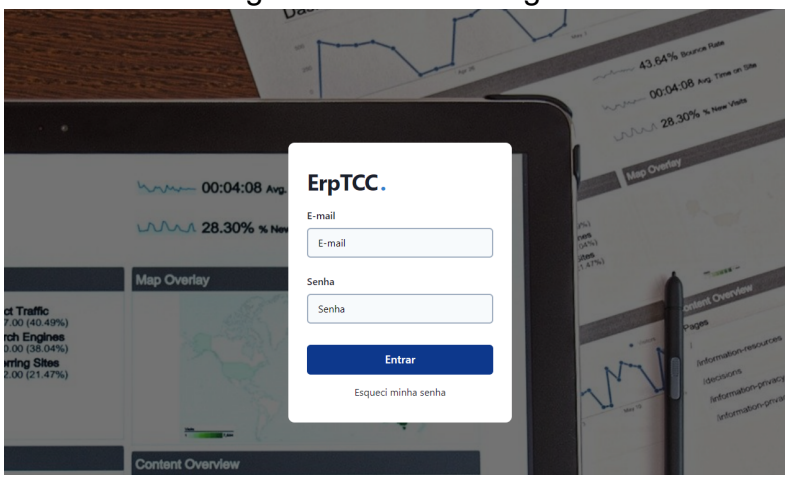
### 4.1 Desenvolvimento Web

Nesta seção são apresentados os resultados do sistema após o desenvolvimento deste. Conforme na Figura 21 temos a tela de login onde o funcionário ou o administrador do sistema.

Temos a tela de Dashboard do sistema, na Figura 22, onde nela pode-se ver alguns dados essenciais sobre gráficos que futuramente serão manipulados

pelo administrador do sistema. Onde também podemos visualizar a *sidebar*, que neste caso é fixa e dá acesso para todas as funcionalidades da aplicação.

Figura 21: Tela de Login



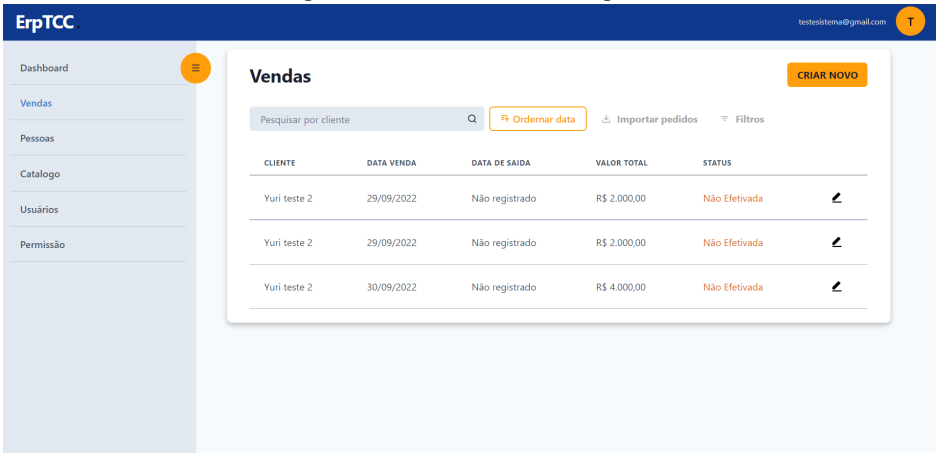

Fonte: Elaborado pelos autores

Figura 22: Tela de Dashboard


Fonte: Elaborado pelos autores

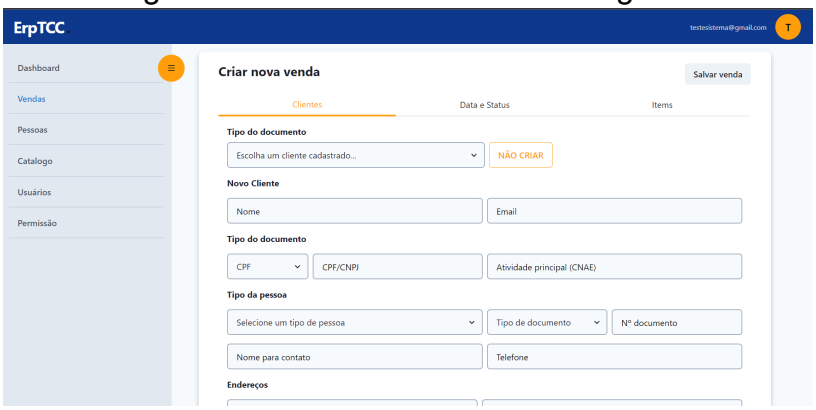
Após isso temos diversas funcionalidades como a de cadastrar usuário, permissões, categorias, produtos, compras e vendas. Neste caso, sempre temos uma listagem na tela inicial e a partir dela podemos criar um novo registro diante da funcionalidade, visualizar ou editar alguma informação dos registros atuais. Pode-se ver um exemplo de listagem na figura 23, ao qual visualizamos alguns filtros que foram mencionados na prototipação do sistema. Na figura 24, podemos ver um exemplo de uma tela do sistema quanto a funcionalidade para criar um novo registro. Conforme visto na Figura 25, pode-se ver quando acessamos a informação para editar temos um botão para sempre editar pois não é sempre que esta informação irá precisar ser atualizada.

Figura 23: Tela de listagem



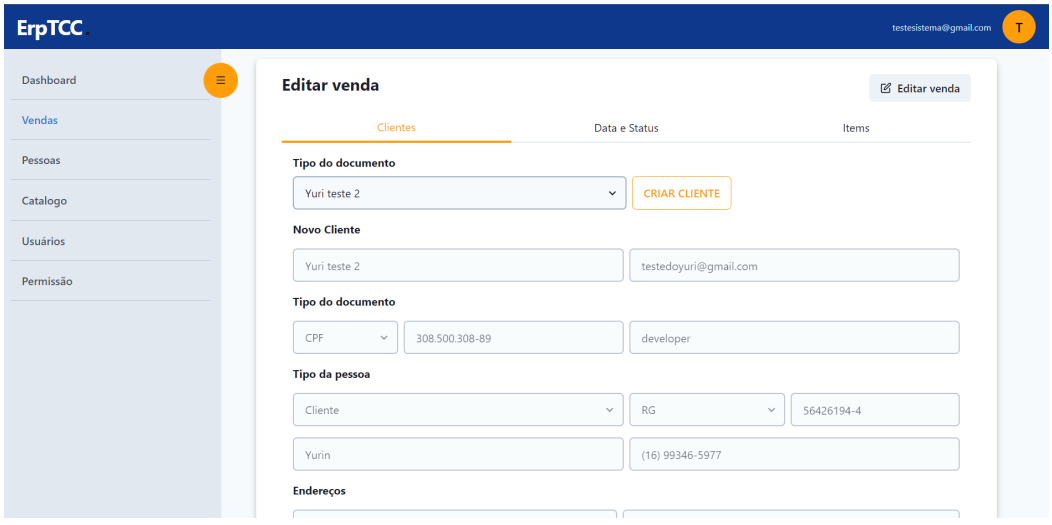

Fonte: Elaborado pelos autores

Figura 24: Tela de criar um novo registro




Fonte: Elaborado pelos autores

Figura 25: Tela de editar um registro




Fonte: Elaborado pelos autores

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

## 4.2 Validações de campos

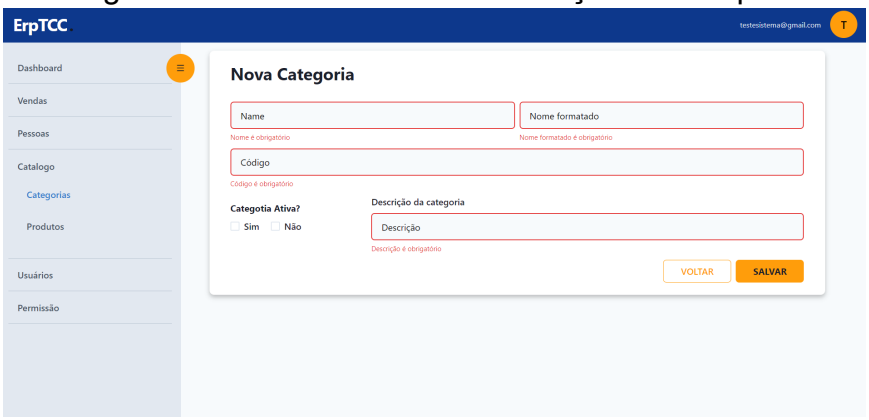
Nesta seção são apresentados alguns exemplos de validação para impedir que o usuário preencha de maneira equivocada. São apresentadas também algumas funções que foram utilizadas para realizarmos máscaras de campos para que a informação possa ser a mais próxima da realidade do usuário. Foi utilizado a biblioteca react-hook-form junto ao yup para realizar validações, podemos visualizar o código na Figura 26 e uma visualização dela no sistema na Figura 27. Na Figura 28 temos as máscaras de campo que foram utilizadas no sistema.

Figura 26: Código com as validações

```
import { yupResolver } from "@hookform/resolvers/yup";
import { SubmitHandler, useForm } from "react-hook-form";
import * as yup from "yup";

const createUserSchema = yup.object().shape({
  name: yup.string().required("Nome é obrigatório"),
  description: yup.string().required("Descrição é obrigatório"),
  nameFormatted: yup.string().required("Nome formatado é obrigatório"),
  cod: yup.string().required("Código é obrigatório"),
});

export default function CreateCategory() {

  const {
    register,
    handleSubmit,
    formState: { errors, isSubmitting },
  } = useForm<CategoriesDTO>({
    resolver: yupResolver(createUserSchema),
  });
```

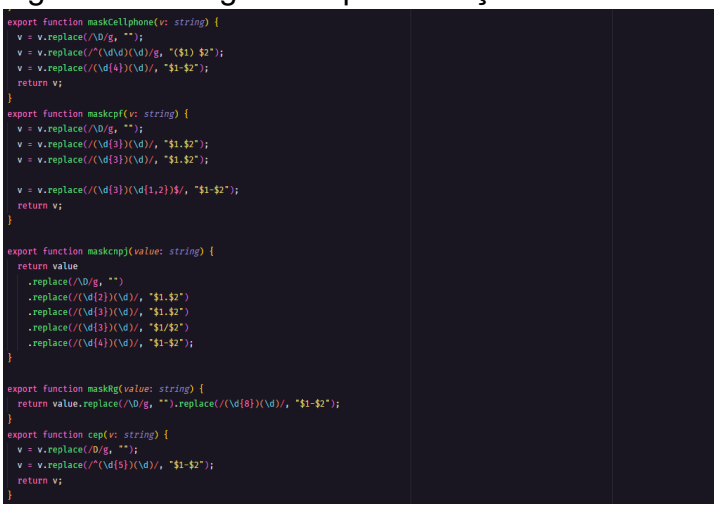
Fonte: Elaborado pelos autores

Figura 27: Formulário com a validação de campos




Fonte: Elaborado pelos autores

Figura 28: Código da implementação de mascará



```
export function maskCellphone(v: string) {
  v = v.replace(/\D/g, "");
  v = v.replace(/^(\d\d)(\d)/g, "($1) $2");
  v = v.replace(/(\d{4})(\d)/, "$1-$2");
  return v;
}
export function maskcpf(v: string) {
  v = v.replace(/\D/g, "");
  v = v.replace(/(\d{3})(\d)/, "$1.$2");
  v = v.replace(/(\d{3})(\d)/, "$1.$2");

  v = v.replace(/(\d{3})(\d{1,2})$/, "$1-$2");
  return v;
}

export function maskcnpj(value: string) {
  return value
    .replace(/\D/g, "")
    .replace(/(\d{2})(\d)/, "$1.$2")
    .replace(/(\d{3})(\d)/, "$1.$2")
    .replace(/(\d{3})(\d)/, "$1/$2")
    .replace(/(\d{4})(\d)/, "$1-$2");
}

export function maskRg(value: string) {
  return value.replace(/\D/g, "").replace(/(\d{8})(\d)/, "$1-$2");
}
export function cep(v: string) {
  v = v.replace(/D/g, "");
  v = v.replace(/^(\d{5})(\d)/, "$1-$2");
  return v;
}
```

Fonte: Elaborado pelos autores

## 5 Considerações finais

O sistema surgiu a partir de uma dor ao qual o dono de uma empresa, que neste caso é pai de um dos integrantes do nosso grupo, o qual percebeu que era uma necessidade não somente dele, porém também de mais alguns companheiros próximos que trabalham no mesmo segmento, sendo eles também MEI`s. Assim desta forma o objetivo central era desenvolver um sistema que atendesse algumas regras de ERP, para assim ele conseguir organizar e administrar melhor a sua empresa.

Existem diversos aplicativos no mercado atualmente, porém nem todos possuem a finalidade de atender a empresas menores que não necessariamente dependem de funcionalidades complexas. Assim surgiu a ideia de desenvolver um sistema mais simples de modo que atendesse a pelo menos a dor do pai do nosso companheiro de grupo.

O desenvolvimento em si contou com diversas reuniões para um melhor aprofundamento e sobre o assunto, sendo uma experiência gratificante em colocar todas as técnicas de Engenharia de Software na prática e assim desenhar tudo que era necessário para futuramente desenvolvermos. Com isso, a primeira versão foi pensada para ser, no mínimo, viável às suas necessidades, de modo que o usuário pudesse entrar, realizar cadastros de usuários, clientes, produtos e pudesse ali cadastrar as suas vendas e assim acompanhar de perto seus resultados. Contudo foram surgindo ideias, no início pensávamos em fazer apenas tudo Web, porém após algumas entrevistas decidimos fazer um Web responsivo ao ponto de futuramente transformarmos isso em um PWA.

Um dos maiores desafios que tivemos foi desenvolver a tela de vendas, pois nela existia a união de cliente, datas e itens (produtos). A funcionalidade contava com diversos campos e se deixássemos tudo em uma tela apenas isso iria atrapalhar a jornada do usuário. Após alguns minutos de brainstorming decidimos dividir a funcionalidade em abas o que trouxe um retorno excelente para a funcionalidade otimizando a jornada do usuário.

Desenvolver um software para organização e administrar, envolve detalhes e uma certa atenção para experiência que o usuário irá ter, tendo que sempre ser fácil e intuitivo ao ponto de qualquer usuário saber o que cada funcionalidade faz.

Deste modo podemos ainda cogitar melhorias futuras e implementação de funcionalidades para este projeto, já que este assunto é muito amplo e atípico.

O sistema final alcançou as expectativas deste artigo, visto que ele supera todos os objetivos esperados, embora exista um pesar por não termos implementados todos os módulos, faltando as compras e controle de permissão. Porém, são módulos que, posteriormente, serão desenvolvidos e implementados na aplicação. O fator de não termos implementado, é que a regra de compras envolve diretamente a regra de negócios voltada a um estoque, por ser muito complexa, e levaria mais tempo que o disponível para o seu desenvolvimento.

Este projeto ainda tem muito a ser evoluído, seja devido a interface podendo ser melhorada ou até mesmo para expansão de funcionalidades para assim melhorar o fluxo de trabalho em empresas menores. A melhoria contínua sempre leva o sistema a uma evolução e assim faz ele atender às necessidades gerais.

**Referências**

ANDRÉ, Thiago. *Inbound Marketing: entenda a importância das estratégias de atração!* 3. Set. 2021. Disponível em: < https://agencialoopers.com.br/blog/inbound-marketing/>. Acesso em: 25.ago.2022.

BATISTA, Guilherme Rodrigues Silva; DE MELO, Mariana Bertanha; ROLAND, Carlos Eduardo de França. *MercadimApp: estudo de protótipo de interface para app de compras online*. Revista Eletrônica de Computação Aplicada, v. 2, n. 2, 2021.

CHAKRA. *Create accessible React apps with speed. sd*. Disponível em: <https://chakra-ui.com/>. Acesso em: 17.ago.2022.

DE OLIVEIRA LOPES, Jean. *PHP ou TypeScript: uma comparação de duas linguagens para web pelas suas características. sd.* Disponível em: <Instruções aos Autores de Contribuições para o SIBGRAPI (ifrs.edu.br)>. Acesso em: 17.ago.2022.

DE SOUZA, Cristiano Ferreira; SANTANDER, Victor FA. *Uma Proposta de Elicitação e Análise de Requisitos no Contexto de Médias e Pequenas Empresas de Desenvolvimento de Software*. In: CIbSE. 2011. p. 285-296

DOS SANTOS SOARES, Michel. *Metodologias ágeis extreme programming e scrum para o desenvolvimento de software*. Revista Eletrônica de Sistemas de Informação, v. 3, n. 1, 2004.

DUARTE, Nuno Filipe Brandão. *Frameworks e Bibliotecas Javascript* . 2015. *Tese de Doutorado.*

ESTEVES, Jéssica Rodrigues; ARAÚJO, Jair Jonko. *Aprender e ensinar design digital no contexto da cibercultura: experiência de ensino com metodologias ativas*. RELACult-Revista Latino-Americana de Estudos em Cultura e Sociedade, v. 6, 2020.

RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

JANAINA. Artigo da Plataforma DevMedia. *Técnicas para levantamento de Requisitos*. 2009. Disponível em: <Artigos > Engenharia de Software > Técnicas para levantamento de Requisitos>. Acesso em: 24.ago.2022

JUNIOR, Alexandre Badoco; DO PRADO, Ely Fernando. GETEASY: *Prototipação de Aplicativo auxiliar em representação comercial*. Revista Eletrônica de Computação Aplicada, v. 2, n. 2, 2021.

LAMPEÃO, Odibar João. *Ciclo de Vida de Desenvolvimento de Sistemas* (2014).

PAULA FILHO, Wilson de Pádua. *Engenharia de Software: fundamentos, métodos e padrões*, Março de 2000.

ROSEMBERG, Carlos et al. *Prototipação de software e design participativo: uma experiência do atlântico*. IHC , v. 8, p. 312-315, 2008.

SANTOS, Wesley Anjos dos. *Aplicativo para o serviço de atenção domiciliar*. 2020.

SATO, Danilo Toshiaki. *Uso eficaz de métricas em métodos ágeis de desenvolvimento de software*. Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo , v. 139, 2007.

SILVA, Bruno Firmino da. *Scrum aplicado ao desenvolvimento de jogos digitais*. 2010.

TEIXEIRA, Fabrício. *Introdução e boas práticas em UX Design*. Editora Casa do Código, 2014.

TOMÁS, Mário Rui. *Métodos ágeis: características, pontos fortes e fracos e possibilidades de aplicação*. 2009.

TOTVS. *Metodologia ágil: o que é e como implementar.* 2021. Disponível em: <Metodologia ágil: o que é e como implementar - TOTVS>. Acesso em: 02.out.2022.