UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Maria Isabel Costa Santiago

Sociedade Empreendedora, uma opção de unir o povo ao povo

Recife

Dezembro de 2021

Maria Isabel Costa Santiago

Sociedade Empreendedora, uma opção de unir o povo ao povo

Artigo apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE
Departamento de Estatística e Informática
Curso de Bacharelado em Sistemas de Informação

Orientador: Silvana Bocanegra

Recife
Dezembro de 2021

Sociedade Empreendedora, uma opção de unir o povo ao povo

[Maria Isabel Costa Santiago] [1], **[Silvana Bocanegra]** [1]

[1]Departamento de Estatística e Informática – Universidade Federal Rural de Pernambuco
Rua Dom Manuel de Medeiros, s/n, - CEP: 52171-900 – Recife – PE – Brasil

[mariaisabelcs@live.com, silvana.bocanegra@gmail.com]

Resumo. *O Sociedade Empreendedora foi um projeto desenvolvido para atender pequenas comunidades de Recife, Pernambuco, a fim de criar um e-commerce para aplicativo móvel, visando atender aos cidadãos destas áreas que possuem pequenos comércios ou trabalham como pintores, pedreiros, encanadores, etc. O sistema é composto por um aplicativo e um painel administrativo responsável pelo gerenciamento de usuários, que será detalhado neste artigo. As soluções tecnológicas utilizadas no desenvolvimento do front-end do painel foram Vue.Js e Nuxt.Js, frameworks Javascript, sendo o Nuxt.Js uma extensão do Vue.*
Palavras-chaves*: Vue.js, Nuxt.Js, Front-end, Javascript, Frameworks.*

Abstract. *Sociedade Empreendedora was a project developed to serve small communities in Recife, Pernambuco, with the objective of creating a textit e-commerce for mobile application, aiming to serve citizens in these areas who have small businesses or work as painters, bricklayers, plumbers, etc. The system consists of an application and an administrative panel responsible for managing users, which will be detailed in this article. The technological solutions used in the development of the front-end of the panel were Vue.Js and Nuxt.Js, frameworks Javascript, with Nuxt.Js being an extension of Vue.*
Palavras-chaves*: Vue.js, Nuxt.Js, Front-end, Javascript, Frameworks.*

1. Introdução

Os brasileiros estão usando cada vez mais o celular, e apesar da maior parte deste uso ser em aplicativos de comunicação e entretenimento, 85% dos brasileiros que possuem *smarthphone* fazem compras por aplicativos de celular.[jornal do Brasil 2021]

A Shopify, quinta melhor plataforma de *e-commerce* do mundo em outubro[Haan and Bottorff 2021], define *e-commerce* como o comércio eletrônico ou comércio pela *internet*, referente a compra e venda de bens ou serviços pela *internet* e a transferência de dinheiro e dados para a execução dessas transações.[Shopify 2021]

Com o aumento no comércio por aplicativos cresce também a necessidade das lojas se adaptarem a venda de produtos *online*, seja utilizando alguma rede social para divulgação ou venda, uma plataforma de *e-commerce* ou um aplicativo próprio.

Movidos pela necessidade do mercado dois empresários, João(nome fictício) e Mário(nome fictício), viram a oportunidade de criar um aplicativo para atender pequenos comerciantes e prestadores de serviços residentes em comunidades da Região Metropolitana do Recife. Para o desenvolvimento do Sociedade Empreendedora(nome fictício) foi contratado o serviço da empresa MICS(Nome fictício), onde a autora deste artigo trabalha.

O sistema, conforme mostrado na Figura1, é formado por um aplicativo de *e-commerce* e um painel administrativo, responsável pelo gerenciamento de dados, conteúdo e transações do aplicativo.

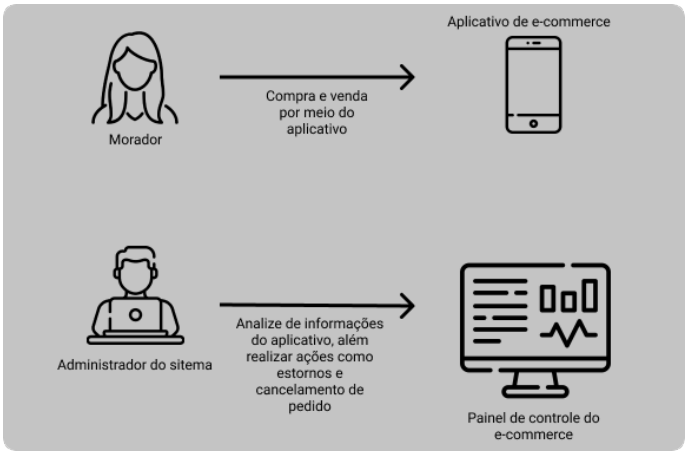

Figura 1. Visão geral do sistema Sociedade Empreendedora

As etapas de idealização e criação do *design* foram feitas por Emanuele (nome fictício) em parceria com líderes e moradores das comunidades, e não serão abordados neste artigo, bem como o aplicativo, desenvolvido por outros programadores da empresa MICS(nome fictício). O objeto de estudo deste artigo é apenas o desenvolvimento *front-end* do painel administrativo desenvolvido pela autora.

2. Objetivos

Desenvolver o *front-end* de uma aplicação *web* responsável pelo gerenciamento do conteúdo publicado, dados do aplicativo e transações financeiras no sistema Sociedade Empreendedora(nome fictício).

3. Referencial teórico

3.1. *Front-end web*

Segundo a w3C, principal organização reguladora do *World wide web*, conhecido popularmente como www, define-se como *front-end* a parte do produto que será responsável por interagir com o usuário, mais especificamente a *interface* do usuário[w3C 2021].

A estrutura mais básica de uma página *web* é composta pela linguagem de marcação HTML, *HyperText Markup Language*, responsável por definir a estrutura do conteúdo da página. Também fazem parte desta estrutura o CSS, que adiciona o estilo e o javascript que cuida das funcionalidades e interações.[Mozilla 2021b].

A DOM é uma convenção padrão organizada em forma de árvore(Figura2), responsável pela representação e interação de elementos HTML.[Mozilla 2021b] A tag é uma representação de um elemento em HTML e para cada uma destas a DOM cria um elemento que será mostrado na página.

Já o javascript é uma linguagem de programação interpretada e segundo a MDN é a mais conhecida no desenvolvimento de páginas web[Mozilla 2021c]. A partir dele é

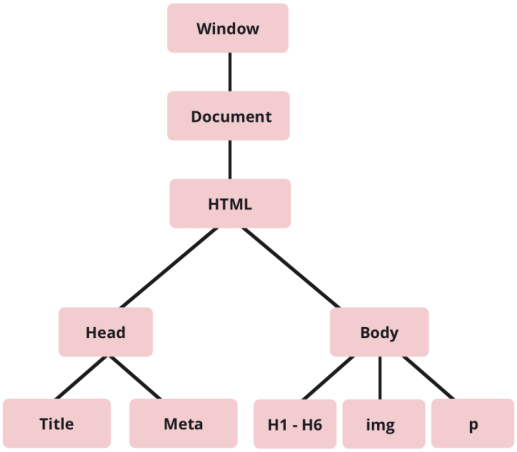

Figura 2. Árvore criada pela DOM

possível manipular elementos da DOM utilizando seletores. Além disso, ele ainda é base para *frameworks*, *web back-end*, ambiente sem o *browser* e responsável pelo gerenciamento dos dados, quanto para *frameworks front-end*, como VueJs, AngularJs e ReactJs.

4. Ferramentas

Tratando-se de um projeto *front-end web* foram escolhidas algumas ferramentas de modo a garantir qualidade na entrega e reutilização de código. Neste projeto o framework principal é o NuxtJs e será apresentado a seguir.

4.1. Nuxt.js

O NuxtJs é um *framework* baseado em outro *framework*, o VueJs, que trata da camada de visualização e possui fácil integração com outras bibliotecas. Assim como o Vue, o Nuxt possui o sistema de componentes que oferece a opção de reutilizar os elementos já criados, evitando repetição de código e ganhando tempo de codificação. Além disso, ele possui alguns facilitadores na criação de um projeto *front-end* quando comparado ao VueJs puro, como o conceito de rotas, o *middleware* e os modos.

O sistema de rotas no NuxtJs é abstraído, enquanto no VueJs existe um arquivo com toda a árvore de rotas, nome, *urls*, entre outros detalhes. No NuxtJs se transforma em pages, logo o nome da sua página seria o nome da rota, se a página estiver numa pasta o nome da pasta estaria na url e assim por diante, assim como mostrado na Figura 3.

Ao iniciar um projeto com este framework é feita a escolha do modo, que pode ser SPA ou Universal. O primeiro modo SPA e quer dizer *single page application*. Nele a renderização é feita no *browser*, enquanto no modo universal ela é ssr, *server-side-rendering*, ou seja, é feita no nuxt server, um pequeno servidor Node rodando em background. O projeto *front-end* do Sociedade Empreendedora(nome fictício) fez uso do modo universal para seu desenvolvimento.

A última grande diferença é o *Middleware*, que permite que uma função seja chamada antes da renderização da página. Permitindo, por exemplo, a validação de *token* para usuário logado.

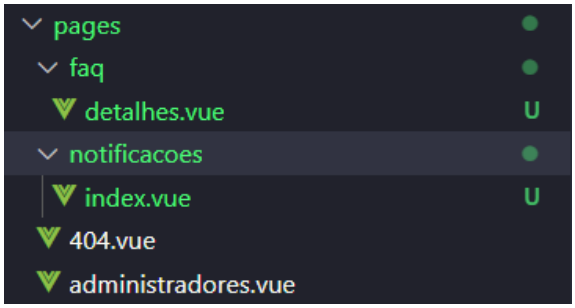

Figura 3. Representação das rotas no NuxtJs

5. Arquitetura inicial

Antes de iniciar um projeto com Nuxt deve-se ter previamente alguns pacotes instalados. No caso do Sociedade Empreendedora(nome fictício) foi escolhido o yarn como gerenciados de pacotes, além do Node atualizado para a versão LTS(*long-term support*), última versão estável e segura, na época de início do desenvolvimento.

No início desse projeto, alguns comandos foram necessários para a criação da estrutura inicial, além disso, é neste momento que o desenvolvedor define o nome e algumas outras configurações. Após a seleção de todas as opções o desenvolvedor terá uma visualização semelhante à (Figura4).[NuxtJs 2021]

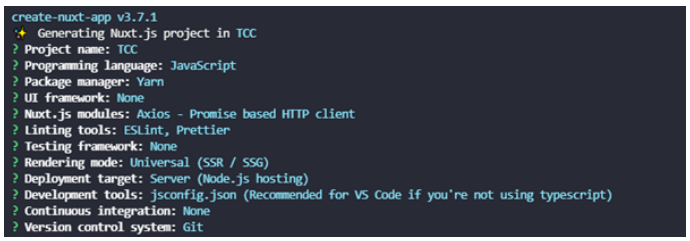

Figura 4. Terminal após seleção das configurações iniciais de um projeto NuxtJs

5.1. Módulos

Para este projeto, algumas bibliotecas e módulos foram selecionados para auxiliar o desenvolvimento. Como Axios, utilizado para integração, que funciona como um *client* HTTP baseado em *Promise*. No lado do servidor, ele usa o módulo http node.js nativo, enquanto no navegador ele usa *XMLHttpRequests*.[Mozilla 2021a] É possível estender algumas configurações para aprimorar a integração com alguma API, como tratamento de respostas e de erros. Toda chamada HTTP é feita por este módulo [Axios 2021].

Outro módulo adicionado ao projeto foi o DotEnv, utilizado para armazenar as variáveis de ambiente. Ao invés de colocar diretamente no código dados sensíveis como: chaves de acesso, *urls* ou diretórios, são utilizadas apenas referências onde os valores reais estão armazenados no ambiente responsável pelo *deploy*.

5.2. *Plugins*

No projeto existe uma pasta *plugin*, que armazena os pacotes e módulos personalizados caso seja necessário. Para ter um *plugin* adicionado em um projeto Nuxt é necessário

adicioná-lo utilizando o gerenciados de pacotes do projeto, no caso do Sociedade Empreendedora(nome fictício) é o yarn, seguido do comando de adição e a referência passada na documentação, após isto é feita a importação para uso em um arquivo na pasta, chamada no ”nuxt.config.js”para uso em toda a aplicação.

O primeiro adicionado foi o *Vuex-persist* que armazena dados no *localStorage*, responsável por salvar, adicionar, recuperar ou excluir dados localmente em um navegador *web*. Nele é feita a persistência da sessão e dos dados, para aumentar a segurança é possível encriptar o que é armazenado. Outro *plugin* utilizado foi o *vue-notification*, que gera notificações no painel administrativo, e que aceita personalização no que diz respeito ao texto, cor, posição na tela e duração da notificação, por exemplo. Os tipos de notificações pelo qual ele é responsável no Sociedade Empreendedora(nome fictício) são mensagens de erro ou sucesso após uma requisição da API.

Para a criação de gráficos foi utilizado o *High-charts* e para os calendários, usados como filtros em algumas listagens, o v-calendar. Como os dados vêm brutos do *backend* fez-se necessária a adição de máscaras para formatá-los. A *money-mask*, foi responsável pela formatação dos valores na moeda brasileira. Já o *mask* realizou a formatação dos campos de CPF e telefone.

Filters é um recurso do VueJs, que se mantém no NuxtJs, e serve para aplicar formatação diretamente no *template* HTML, ao invés de usar máscaras, ele permite criar expressões para formatar um dado, de modo a evitar repetição de código, neste projeto eles foram criados para formatação de data e hora.

5.3. *Atomic Design*

Em 2013, Brad Frost criou o *Atomic Design* e propôs uma solução de organizar os componentes e criar um *Design System*. [Frost 2013]

O *Atomic* possui 5 níveis, são eles: átomo, moléculas, organismos, *templates* e páginas. O átomo é o nível mais interno e o menor componente como, por exemplo, um botão, ícone ou uma *label*. Moléculas são as junções dos átomos para se tornar algo um pouco maior, como um campo de busca, que possui um campo de texto com um botão. Individualmente o campo de texto é apenas um átomo, mas ao juntá-lo com o botão, torna-se uma molécula. Com a união de moléculas conseguimos criar um componente mais complexo, que seriam os organismos, o *menu* lateral seria um exemplo.É no *template* que os organismos são agrupados para criação da página, no entanto, ainda não é neste componente que é feita a adição de conteúdo. Isso ocorre nas páginas, o nível mais alto neste *Design System*.

6. Funcionalidades do painel administrativo

6.1. *Login* e sessão

O primeiro usuário do painel recebeu o acesso por *email*, enviado por nós, para realizar o *login* (Figura5). Em seguida, após *logar* ele terá permissão de criar administradores, na seção de gerenciar administradores. Também é possível realizar a redefinição de senha apenas com o *email*, na opção esqueci minha senha, neste caso um *link* de redefinição é enviado para o *email* do administrador se ele estiver cadastrado no sistema. A persistência da sessão do usuário é feita utilizando o *middleware*, que faz uma verificação antes de cada alteração de rota, para evitar que um usuário não logado consiga navegar pelo painel.

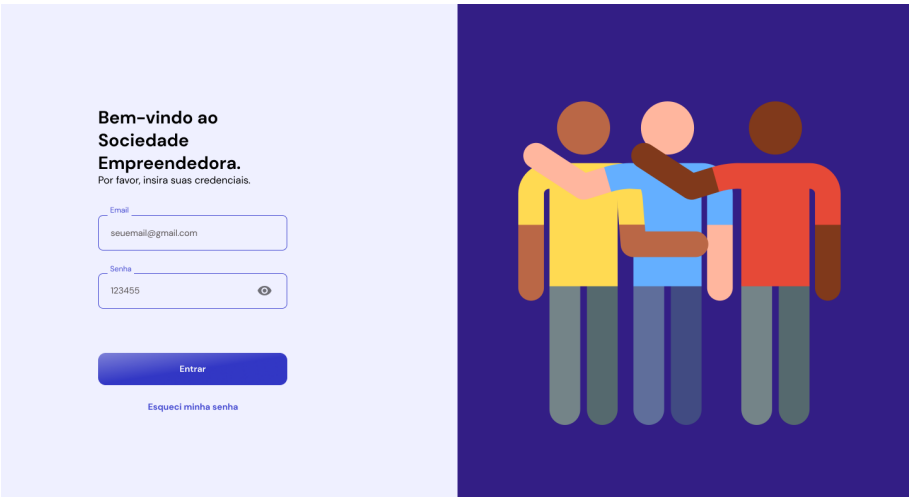

Figura 5. Tela de login

6.2. *Dashboard*

Esta seção é a primeira após o *login*, ela possui um breve resumo das últimas atividades do aplicativo, como quais últimos 10 usuários cadastrados na plataforma, as últimas 6 notificações, número de pedidos e além de ter filtros de novos usuários e de pedidos (Figura6).Ela é composta ainda por um gráfico de novos usuários e *cards* de pedidos, com filtro por período definido pelo cliente.

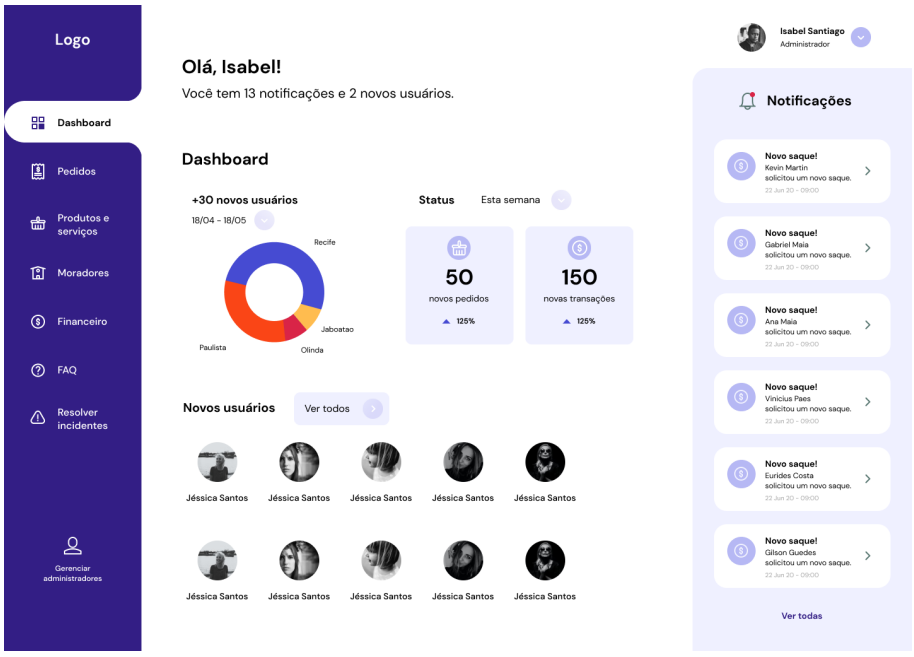

Figura 6. Tela inicial com usuário logado

6.3. Pedidos

Todos os pedidos feitos pelo aplicativo são listados no painel, são exibidas também informações do pedido como: nome do comprador, *item*, loja e valor do produto ou

serviço e ainda realizar ações de estorno e suspensão do pedido (Figura7). Após esta ação o administrador pode escolher enviar uma mensagem informando qual o motivo da suspensão para os envolvidos na compra, tanto o dono da loja ou prestador de serviço quanto o usuário, receberão a mensagem por *email*. Na tabela ainda, é possível filtrar por *status* do pedido, tipo de entrega e data. Também é possível fazer uma busca por nome do comprador e loja.

6.4. Notificações

A tela de *dashboard* exibe apenas um pequeno resumo das últimas notificações, mas a partir dela é possível chegar na tela geral de notificações. Todas as notificações vêm do aplicativo, e podem ser relacionadas a saque, denúncia e alteração de endereço. As notificações não são por *streaming*, como se tratar da primeira versão do projeto foi escolhida a opção mais simples, para posterior melhoria, então é necessário recarregar a página para exibir as novas notificações. Existe filtro por *status*, data e tipo, além de uma busca por nome do solicitante.

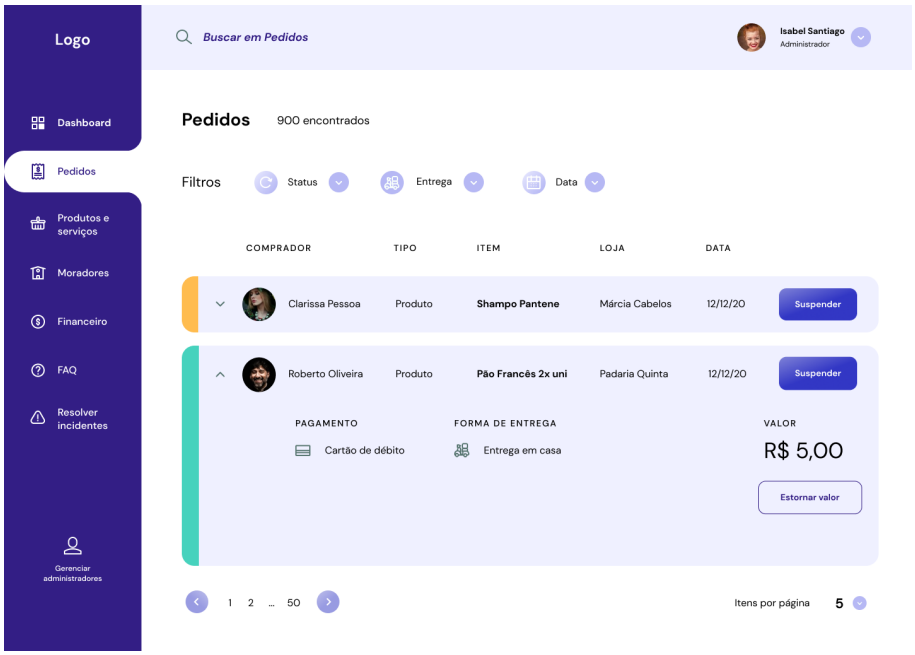

Figura 7. Tela de pedidos

6.5. Produtos e serviços

Esta seção não possui apenas a listagem dos produtos e serviços, mas também é possível adicionar os *banners* que serão mostrados na tela inicial aplicativo, criação de novas categorias do tipo produto ou serviço, edição de nome, imagem e remoção. No detalhe de cada pedido é informado a categoria, tipo, nome, loja e avaliações e é possível fazer a suspensão caso o administrador ache necessário (Figura8).

6.6. Moradores

Aqui são listados todos os usuários do aplicativo, além de ser possível adicionar um novo usuário. A partir da tabela podemos visualizar a tela de detalhes do usuário, aonde o administrador consegue ver a loja do morador, se ele possuir, a descrição, avaliação, quais itens

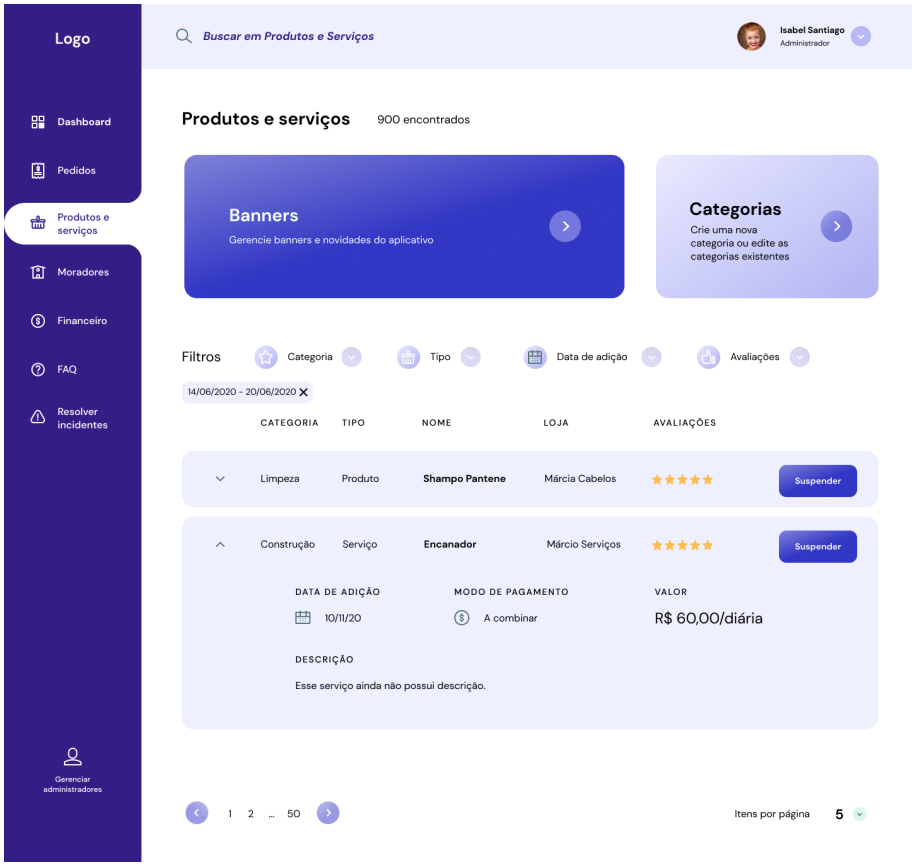

Figura 8. Tela de produtos e serviços

em estoque, além possibilitar a suspensão da loja e dados do próprio usuário (Figura9. No Sociedade Empreendedora(nome fictício) o usuário é tanto o comprador quanto o responsável pelo serviço ou loja.

6.7. Financeiro

É na seção de finanças, que ficam todas as transações, que podem ser cobranças, pagamentos ou saques, sendo importante ressaltar que todo o valor pago é repassado ao cliente uma vez por semana, na data estipulada por ele na fase desenvolvimento, por se tratar da primeira versão, algumas funcionalidades são feitas de forma manual. Nesta parte são listadas todas as transações do aplicativo e caso necessário pode ser feito o estorno da compra.

Os saques feitos pelas lojas tem que ser liberados no painel, então quando o dono da loja faz uma solicitação por meio do aplicativo cabe ao administrador liberar ou recusar. Esse processo é feito em uma tela dedicada ao saque e caso seja recusado pode-se adicionar uma mensagem informando o motivo para posteriormente ser exibido para o usuário no aplicativo. Nesta tela também é feita a importação do arquivo de retorno e a geração do arquivo de remessa.

As metas e seus valores são definidas nesta página, a ideia inicial do cliente é oferecer bonificações ou redução na taxa de serviço do aplicativo, mas isso é definido a critério do administrador.

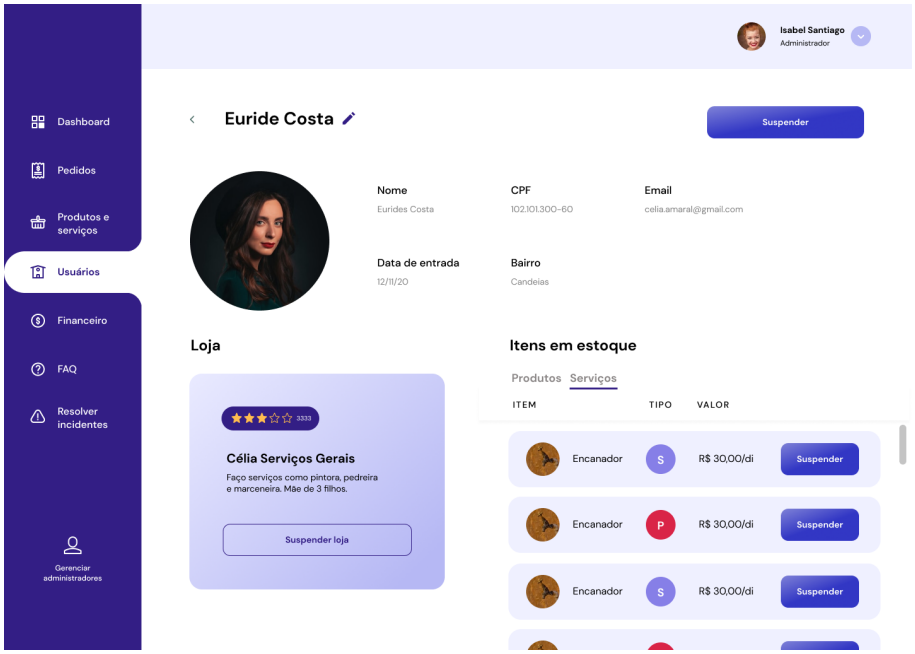

Figura 9. Detalhes do morador

6.8. Gerenciar administradores

Os usuários relacionados nesta seção são os administradores, as únicas ações nesta página são a listagem de administradores, remoção, busca por nome e novo administrador. Nesta primeira versão do painel não existem vários níveis de acesso então todos possuem as mesmas permissões.

6.9. FAQ

As perguntas do FAQ do aplicativo são criadas nesta seção, qualquer administrador pode criar, editar, remover ou pesquisar por palavras no título das perguntas.

6.10. Resolver incidente

O usuário pode reportar um problema através do aplicativo, e os mesmos são listados nesta página. A partir da listagem inicial o administrador consegue ir para uma página de detalhes com a mensagem relacionada ao problema, e escolher qual ação será aplicada para a resolução do problema, que pode ser realizada fora do painel ao entrar em contato por *email*, fazendo a suspensão do usuário ou com o estorno de compras.

7. Conclusão

O painel administrativo foi desenvolvido com objetivo de visualizar e manipular os dados, transações financeiras e gerenciar conteúdos oferecidos pelas lojas ou prestadores de serviço do sistema Sociedade Empreendedora(nome fictício). Dada a proposta inicial dos clientes, José(nome fictício) e Mário(nome fictício), o projeto foi concluído e está passando por um período de testes com os usuários finais, sendo disponibilizado para moradores de duas comunidades da Região Metropolitana do Recife, para posterior expansão as demais.

Tudo que foi acordado inicialmente foi entregue, mas esta é apenas uma versão inicial, em que algumas melhorias estão sendo avaliadas, como o serviço de notificação por *streaming*, no painel, que atualmente ocorre apenas na atualização da página. Em relação às tecnologias utilizadas todas cumpriram seu papel e não houve nenhuma limitação, sendo NuxtJs um bom *framework* para a criação de painéis administrativos.

Referências

Axios (2021). Axios . Disponível em: https://github.com/axios/axios. Acesso em: 02 de novembro de 2021.

Frost, B. (2013). Atomic Design . Disponível em: https://bradfrost.com/blog/post/atomic-web-design/. Acesso em: 15 de novembro de 2021.

Haan, K. and Bottorff, C. (2021). *The Best E-Commerce Platforms Of October 2021*. Disponível em: https://www.forbes.com/advisor/business/software/best-ecommerce-platform/. Acesso em: 12 de dezembro de 2021.

jornal do Brasil (2021). Brasileiros estão usando mais o celular. Disponível em: https://www.jb.com.br/ciencia_e_tec/2020/07/1024761-brasileiros-estao-usando-mais-o-celular.html. Acesso em: 12 de dezembro de 2021.

Mozilla (2021a). Mensagens HTTP . Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Messages. Acesso em: 02 de novembro de 2021.

Mozilla (2021b). *HTML: HyperText Markup Language*. Disponível em: https://www.w3.org/WAI/EO/wiki/Front-End_Developer_Responsibilities_Mapping#Front-End_Developer. Acesso em: 15 de dezembro de 2021.

Mozilla (2021c). *Javascript*. Disponível em: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Acesso em: 15 de dezembro de 2021.

NuxtJs (2021). *Installation* . Disponível em: https://nuxtjs.org/docs/get-started/installation. Acesso em: 02 de novembro de 2021.

Shopify (2021). *What is Ecommerce?* Disponível em: https://www.shopify.ca/encyclopedia/what-is-ecommerce. Acesso em: 12 de dezembro de 2021.

w3C (2021). *Front-End Developer Responsibilities Mapping*. Disponível em: https://www.w3.org/WAI/EO/wiki/Front-End_Developer_Responsibilities_Mapping#Front-End_Developer. Acesso em: 15 de dezembro de 2021.